GIT使用手册

*------分布式版本控制系统入门级使用手册*

作者：高勇

日期：2017年1月24日星期二

邮箱：gaoyong@my-soft.net.cn

版本管理无疑是非常重要的事情，特别是要保障变化的团队能够持续协同工作，对于知识库的形成和管理，即使是对个人也有很大的意义。对程序员而言，这是必须掌握的技能。如果有更好的比对和合并工具支撑，相信对任何电子文档的创作者，都是一个利器……

目录

[1. 版本控制系统简介 1](#_Toc531636867)

[1.1. 历史 1](#_Toc531636868)

[1.2. VSS 1](#_Toc531636869)

[1.3. CVS 1](#_Toc531636870)

[1.4. SVN 1](#_Toc531636871)

[1.5. CC 1](#_Toc531636872)

[2. GIT原理 2](#_Toc531636873)

[2.1. 工作原理 2](#_Toc531636874)

[2.2. 使用场景 2](#_Toc531636875)

[3. 单机 2](#_Toc531636876)

[3.1. 版本库 3](#_Toc531636877)

[3.2. 暂存区 3](#_Toc531636878)

[3.3. 工作区 4](#_Toc531636879)

[3.4. 临时储存 4](#_Toc531636880)

[3.5. 对象 4](#_Toc531636881)

[4. 局域网 5](#_Toc531636882)

[4.1. 环境准备 5](#_Toc531636883)

[4.1.1. 服务器 5](#_Toc531636884)

[4.1.2. 客户端 6](#_Toc531636885)

[4.2. 创建仓库 7](#_Toc531636886)

[4.3. 常用操作 7](#_Toc531636887)

[4.3.1. 建立工作副本 7](#_Toc531636888)

[4.3.2. 获取最新内容 8](#_Toc531636889)

[4.3.3. 查看变更文件列表 8](#_Toc531636890)

[4.3.4. 添加变动文件 8](#_Toc531636891)

[4.3.5. 提交变动 9](#_Toc531636892)

[4.3.6. 恢复工作副本文件 9](#_Toc531636893)

[4.3.7. 从仓库恢复工作副本 9](#_Toc531636894)

[4.3.8. 推送到仓库 9](#_Toc531636895)

[4.3.9. 分支 9](#_Toc531636896)

[4.3.10. 获取历史版本 10](#_Toc531636897)

[4.3.11. 服务端 10](#_Toc531636898)

[5. 互联网知名代码库 10](#_Toc531636899)

[5.1. 环境准备 10](#_Toc531636900)

[5.2. 仓库 10](#_Toc531636901)

[5.3. 工作目录 10](#_Toc531636902)

[5.4. 常用操作 10](#_Toc531636903)

[6. 附录 11](#_Toc531636904)

[6.1. 常用指令 11](#_Toc531636905)

[6.1.1. init 11](#_Toc531636906)

[6.1.2. clone 11](#_Toc531636907)

[6.1.3. add 11](#_Toc531636908)

[6.1.4. commit 11](#_Toc531636909)

[6.1.5. status 11](#_Toc531636910)

[6.1.6. push 12](#_Toc531636911)

[6.1.7. remote 12](#_Toc531636912)

[6.1.8. checkout 12](#_Toc531636913)

[6.1.9. branch 13](#_Toc531636914)

[6.1.10. ls-tree 13](#_Toc531636915)

[6.1.11. pull 13](#_Toc531636916)

[6.1.12. merge 13](#_Toc531636917)

[6.1.13. diff 14](#_Toc531636918)

[6.1.14. tag 14](#_Toc531636919)

[6.1.15. reflog 14](#_Toc531636920)

[6.1.16. log 15](#_Toc531636921)

[6.1.17. fetch 15](#_Toc531636922)

[6.1.18. reset 15](#_Toc531636923)

[6.1.19. config 15](#_Toc531636924)

[1.1.1. rm 16](#_Toc531636925)

[1.1.2. mv 17](#_Toc531636926)

[1.1.3. stash 17](#_Toc531636927)

[1.1.4. rev-parse 17](#_Toc531636928)

[1.1.5. … 18](#_Toc531636929)

[1.2. 图形界面工具 18](#_Toc531636930)

[1.3. 相关资源 18](#_Toc531636931)

# 版本控制系统简介

## 历史

好记性不如烂笔头，最好是将变更记录下来，便于事后了解来龙去脉。

软件的开发更是如此，时间持续长，多人协作，人员更迭，还需要回复和回滚。从2000年开始，我第一次接触的是Microsoft Visual Studio 6 系列的开发工具，其中Visual Source Safe让人记忆犹新。后来随着平台的变迁，从windows到linux，以及网络环境的变更，从局域网到互联网，各种版本控制系统软件都一一体验。目前可以说，研发管理的重要一环就是版本控制软件，因为他确保了研发资产的延续……

## VSS

非常优秀的软件，我的启蒙老师。我一直觉得他是称职的，大多数问题都可以解决。

的确看到有一些开发人员，在linux下编辑，再通过ftp上传到windows的VSS服务器，配置专门的版本管理员合代码。

## CVS

仓库越来越大，差不多几百人都使用同一个版本控制系统服务，要求每个开发人员自己合入代码。CVS对跨平台的支持和性能，还是非常不错的。

## SVN

用得最久的一个，分支处理的非常好。一些开源的软件也使用他来管理分布在世界各地的开发人员。

## CC

昙花一现，IBM应该靠这个挣不到钱了。

开源软件大大推进了软件行业的发展，必需支持社区。

# GIT原理

## 工作原理

git push …

git commit…

git pull …

图 1 GIT的工作原理

## 使用场景

GIT的几种使用模式，单机使用，局域网应用，互联网知名代码仓库……

# 单机

GIT的日常工作就在本地进行，当要同步到远程仓库时，才需要使用push/pull/fetch指令去协同。如果只在本地工作，原理图应该是这样。本地库则直接通过init创建。从其他本地库获取工作副本，则使用命令clone。

add

checkout -- <file>

commit

reset HEAD

图 2 本地工作原理

当然还有一个objects仓库，专门存相关的文件。一般在.git/objects/中

## 版本库

从暂存区写到版本库

git commit

显示版本库目录树

git ls-tree -l HEAD

## 暂存区

从工作区增加到暂存区

git add file\_in\_workspace

从版本库替换暂存区目录树

git reset HEAD

git reset HEAD <file>

从暂存区删除文件

git rm –cached <file>

从版本库替换暂存区和工作区的文件

git checkout HEAD -- file\_in\_index

git checkout HEAD .

显示暂存区的目录树

git ls-files -s

如果想用ls-tree看与版本库一样的结构,需要先将暂存区目录树写入对象库

git write-tree

这里会输出一个tree id

git ls-tree -l $id

如果输出的是766cbc1b3205891668715dace11cf7b257a643e8

echo 766cbc1b3205891668715dace11cf7b257a643e8 |xargs git ls-tree -l -t -r

比较暂存区与版本库

git diff –cached

## 工作区

从暂存区撤销工作区改动

git checkout -- file\_in\_index

git checkout .

清除工作区未加入到版本库的文件和目录

git clean -fd

比较工作区与暂存区

git diff

比较工作区与版本库

git diff HEAD

## 临时储存

临时储存可让你保存工作区，并切换到其他分支，当需要时，可以从储存区恢复出来。

查看文件状态

git status

储存当前工作

git stash

查看储存的工作列表

git stash list

恢复最近的储存

git stash apply

恢复指定的储存

git stash apply stash@{1}

删除最近的储存

git stash drop

## 对象

执行命令，会看到多个SHA1的对象ID

git log -1 –pretty=raw

显示对象ID的类型

git cat-file -t $SHA1\_ID

用git内部命令显示对象ID的内容

git cat-file -p $SHA1\_ID

显示objects的文件列表，可以看到所有对象的实际文件存放方式

ls -R $prj\_root/.git/objects

查看commit\_id的信息

git log -1 –pretty=raw –graph $COMMIT\_ID

简单罗列文件状态，并给出分支

git status -s -b

显示指定分支对应的信息,如下三个指向相对的信息

git log -1 HEAD

git log -1 master

git log -1 refs/heads/master

在.git目录中查找，可获得如下信息

$ find .git -name HEAD -o -name master

.git/HEAD

.git/logs/HEAD

.git/logs/refs/heads/master

.git/logs/refs/remotes/origin/master

.git/refs/heads/master

.git/refs/remotes/origin/master

分别查看相关文件内容，可以查看到分支指向的是一个commitID

$ cat .git/HEAD

ref: refs/heads/master

$ cat .git/refs/heads/master

31bba8fa9f47b107f6330ad7f1e3bf664d21e239

$ cat .git/refs/remotes/origin/master

31bba8fa9f47b107f6330ad7f1e3bf664d21e239

$ git cat-file -t 31bba8fa9f47b107f6330ad7f1e3bf664d21e239

commit

$ git cat-file -p 31bba8fa9f47b107f6330ad7f1e3bf664d21e239

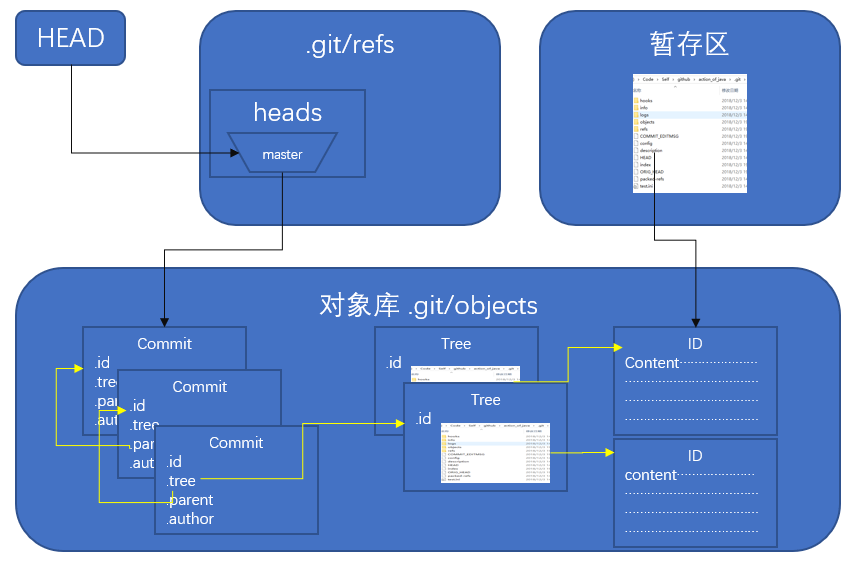
tree 1856c0fd23f992f2f7f18951e5075d156700c7c4

parent 51ace77af7a14f80941e6c78e42cf98955dbd122

author gaoy <admin@wucar.com.cn> 1543844423 +0800

committer gaoy <admin@wucar.com.cn> 1543844423 +0800

dist



以上是git对象的关系图。可以用如下命令看引用分支对应的提交ID。

$ git rev-parse master

31bba8fa9f47b107f6330ad7f1e3bf664d21e239

$ git rev-parse refs/heads/master

31bba8fa9f47b107f6330ad7f1e3bf664d21e239

$ git rev-parse HEAD

31bba8fa9f47b107f6330ad7f1e3bf664d21e239

查看COMMITid的hash的组成：

$ git cat-file commit HEAD

tree 1856c0fd23f992f2f7f18951e5075d156700c7c4

parent 51ace77af7a14f80941e6c78e42cf98955dbd122

author gaoy <admin@wucar.com.cn> 1543844423 +0800

committer gaoy <admin@wucar.com.cn> 1543844423 +0800

dist

$ git cat-file commit HEAD |wc -c

203

$ git rev-parse HEAD

31bba8fa9f47b107f6330ad7f1e3bf664d21e239

$ (printf "commit 203\000";git cat-file commit HEAD)|sha1sum

31bba8fa9f47b107f6330ad7f1e3bf664d21e239 \*-

查看文件ID的hash组成

$ git cat-file blob HEAD:readme.txt

https://github.com/wapuboy/ha\_how\_to.git

$ git cat-file blob HEAD:readme.txt |wc -c

41

$ git rev-parse HEAD:readme.txt

50c9a0cc957f87d6ef78635fb92c59e196d45e94

$ (printf "blob 41\000";git cat-file blob HEAD:readme.txt)|sha1sum

50c9a0cc957f87d6ef78635fb92c59e196d45e94 \*-

查看树ID的hash组成

$ git cat-file tree HEAD^{tree}|wc -c

287

$ (printf "tree 287\000";git cat-file tree HEAD^{tree})|sha1sum

1856c0fd23f992f2f7f18951e5075d156700c7c4 \*-

$ git rev-parse HEAD^{tree}

1856c0fd23f992f2f7f18951e5075d156700c7c4

### 一些约定

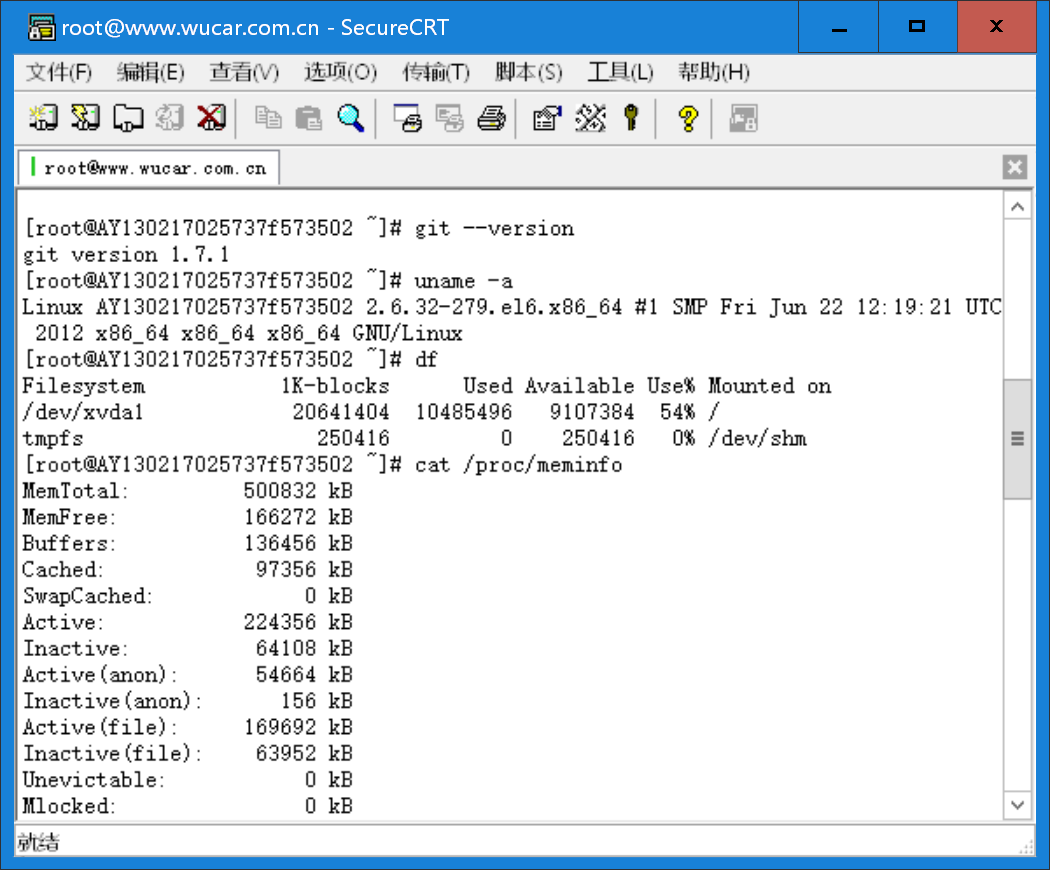
|  |  |
| --- | --- |
| 规则 | 示例 |
| SHA1的部分值可用来代替提交ID | git cat-file -p e695 |
| 别名master------ refs/heads/master |  |
| 别名HEAD代表最近一次提交 | git rev-parse HEAD |
| 符号^可用来代表上一次或者父节点 | HEAD^ 代表上一次提交  HEAD^^ 代表上两次提交 |
| 如果有多个上一次或父节点，可指定序号 | HEAD^1代表上一次提交的第一个  HEAD^2 代表上一次提交的第二个 |
| 符号~可以用来替代祖先提交 | 如e695~5 等价于e695^^^^^ |
| 提交对应的树对象^{tree}来表示 | git rev-parse HEAD e695^{tree}  git rev-parse HEAD e695^^a{tree} |
| 某次提交的文件对象 | e695:path/to/file |
| 暂存区中的文件对象 | :path/to/file |

# 局域网

## 环境准备

准备一台可以通过ssh访问的服务器，我用的是阿里的centos云主机。一台办公电脑作为客户端。

### 服务器



执行如下命令安装GIT软件，以及配置GIT用户。

yum intall git

mkdir /home/git

groupadd git

useradd git -g git -d /home/git

passwd git

#modify the passwd profile to “git:x:502:502::/home/git:/usr/bin/git-shell”

vi /etc/passwd

为了简化客户端每次输入口令，服务端可以将客户端公钥纳入受信任的列表中。

cd /home/git

mkdir .ssh

cd .ssh

#公钥文件来自于客户端

cat id\_rsa.pub\@asus\_notebook >> authorized\_keys

cat authorized\_keys

### 客户端

我的环境是Windows 10，使用qq软件管家安装了Git 2.12.0。



在客户端生成RSA密钥，并将公钥提供给服务端。

#生成密钥对,并将公钥文件提供给服务端

ssh-keygen

scp /c/Users/gaoyo/.ssh/id\_rsa.pub root@www.wucar.com.cn:/home/git/.ssh/id\_rsa.pub@asus\_notebook

## 创建仓库

在服务器上执行如下命令：

cd /home/git

mkdir iot.git

cd iot.git

git init –bare

git init

#version 6+

git init iot

## 常用操作

日常主要是客户端的操作，从服务端获取分支建立本地仓库，在本地仓库增加文件，提交修改，并最后更新到服务端仓库。

### 建立工作副本

在客户端上执行如下命令

mkdir /d/git/www.wucar.com.cn

cd /d/git/www.wucar.com.cn

git clone git@www.wucar.com.cn:/home/git/iot

### 获取最新内容

获取仓库最新的内容

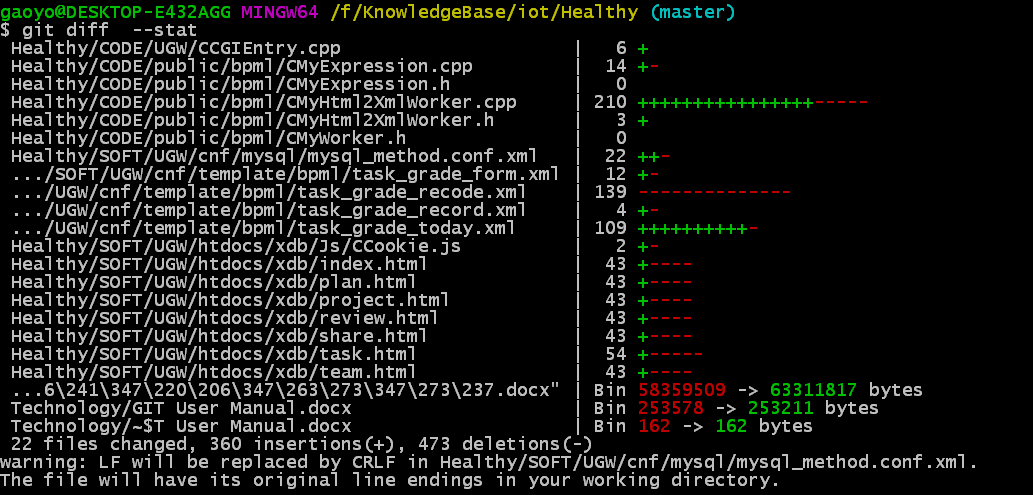
cd /d/git/www.wucar.com.cn/iot

git pull

### 查看变更文件列表

比较工作区与待提交区

git diff –stat



比较工作区与已经提交区

git diff –stat HEAD

比较HEAD与本地仓库master分支

git diff --stat master

比较master与远程服务仓库

git diff --stat master origin

### 添加变动文件

Git add $filename

### 提交变动

Git commit -m”message”

### 恢复工作副本文件

git checkout -- $filename

当工作副本文件被你弄坏，可以使用此命令从最后提交的变动中恢复此文件。

### 从仓库恢复工作副本

放弃工作副本做的修改，从仓库获取最新内容

git fetch origin

git reset --hard origin/master

### 推送到仓库

把本地已经提交的内容发送到仓库的特定分支master。

git push origin master

### 分支

分支用来隔离开发相对独立的特性，master分支是默认的主分支，当创建仓库时就自动生成了。我们可以使用其他的分支开发特性，当完成这个特性后，再把他合并到主分支。

建立分支iot\_test\_20170307

git checkout -b iot\_test\_20170307

如果要与其他人共享分支

git push origin iot\_test\_20170307

合并master到当前分支

git merge master

切换到指定分支 master

git checkout master

比较分支的异同

git diff $source\_branch\_name $target\_branch\_name

删除分支iot\_test\_20170307

git branch -d iot\_test\_20170307

### 获取历史版本

git log # fetch the sha-1 key $1

git checkout $sha-1key

### 服务端

# 互联网知名代码库

## 环境准备

git config –global user.name “wapuboy”

git config –global user.email [wapuboy@wucar.com.cn](mailto:wapuboy@wucar.com.cn)

## 仓库

## 工作目录

## 常用操作

获取指定路径的代码仓库

$ git clone https://github.com/chartjs/Chart.js.git

Cloning into 'Chart.js'...

remote: Counting objects: 12238, done.

remote: Compressing objects: 100% (7/7), done.

Receiving objects: 82% (10057/12238), 6.93 MiB | 5.00 KiB/s

# 附录

## 常用指令

### init

在当前目录下创建git仓库。

### clone

从远程库中克隆仓库

git clone git@www.wucar.com.cn:iot.git

git clone https://github.com/tugenhua0707/testgit

### add

将指定文件添加到git仓库。

### commit

提交文件到仓库

git commit -am“reason”

可以将之前本地删除或修改的文件一律提交，按照当前工作目录组织

#--amend 对刚提交的信息进行修补

#--reset-author 会将作业的id同步修改，不仅仅修改提交者

git commit --amend --reset-author

允许空提交

--allow-empty

### status

查看文件状态。

### push

推送分支到远程仓库

git push origin $branch

### remote

查看远程库的详细信息

Git remote -v

git remote show origin

查看远程库的信息

Git remote

添加远程仓库

git remote add origin <server>

删除

git remote rm origin

修改远程仓库的地址

git remote set-url origin <server>

vi .git/config # 修改remote的数据

### checkout

创建分支

git checkout -b branch\_name

对特定文件，放弃本地修改

git checkout -- readme.txt

切换分支

git checkout branch\_name

### branch

查看分支

git branch

创建分支

git branch branch\_name

切换分支

git checkout branch\_name

删除分支

git branch -d branch\_name

### ls-tree

$ git branch

\* master

$ git ls-tree master

040000 tree 6a96a1ac2a78c1c532f25692a694251c1c9dbe29 Asset

$ git ls-tree -r -t master

040000 tree 6a96a1ac2a78c1c532f25692a694251c1c9dbe29 Asset

100644 blob 70b7fa53c6d0f069b73ad370cdff1f241f02b030 Asset/KeyNote.xlsm

### pull

获取最新的版本

git pull

### merge

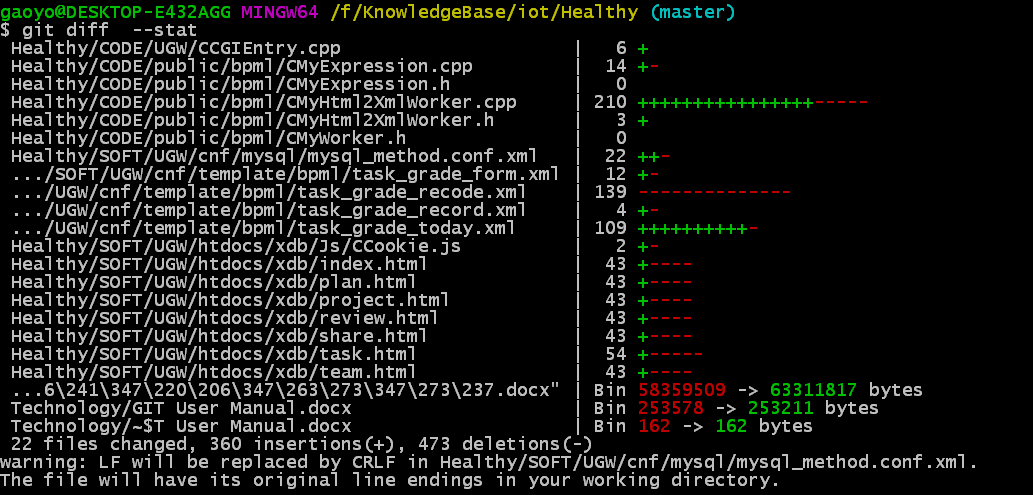
合并指定分支到当前分支

git merge branch\_name

### diff

比较工作区与待提交区

git diff –stat



比较工作区与已经提交区

git diff –stat HEAD

比较HEAD与本地仓库master分支

git diff --stat master

比较master与远程服务仓库

git diff --stat master origin

### tag

打标签

git tag $version\_such\_as\_1.0.0.1 $commit\_id\_which\_can\_use\_reflog\_find\_it

### reflog

查看版本号以及简易日志

### log

查看详细日志

git log

只看指定用户提交的

--author=”gaoy”

所有的都在一行,默认是fuller

--pretty=oneline

图形化界面

--graph --oneline --decorate –all

--name-status

--help

### fetch

### reset

版本回退

回退到上一个版本

git reset -hard HEAD^

回退到前100个版本

git reset -hard HEAD~100

回退到指定版本

git reset -hard bfcfc89

### config

git的配置分为，版本库、git全局、系统三个可见区域。

清除配置的选项

--unset

#git清除账户密码

git config --system --unset credential.helper

#版本库配置文件，优先级最高、作用域最小；

#如我的在/f/Code/Self/github/action\_of\_java/.git/config

git config -e

#全局配置文件，优先级其次、作用域居中；一般路径是用户主目录下的.gitconfig文件，

#如我的在这里：C:\Users\gaoy

git config -e --global

#系统配置文件，优先级最低、作用域最大；一般路径在安装目录，

#如C:\Program Files\Git\mingw64\etc\ gitconfig

git config -e --system

$ git config --global user.name "gauss"

$ git config --global user.email 7457222@qq.com

重置鉴权信息

git config --system --unset credential.helper

设置GIT子命令别名：

git config --global alias.ci commit

git config --global alias.st status

git config --global alias.co checkout

git config --global alias.br branch

git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset %C(bold blue)<%an>%Creset ---%C(yellow)%d%Creset %s %Cgreen(%cr)' --abbrev-commit"

在GIT命令输出中开启颜色显示：

git config --global color.ui true

config可以对任何INI文件进行读写。

$ GIT\_CONFIG=test.ini git config a.b.c.d "hello world ini."

$ ls

test.ini

$ cat test.ini

[a "b.c"]

d = hello world ini.

$ GIT\_CONFIG=test.ini git config a.b.c.d

hello world ini.

### rm

删除文件

git rm file\_path

### mv

git mv -f oldfolder newfolder

git add -u newfolder (-u选项会更新已经追踪的文件和文件夹)

git commit -m "changed the foldername whaddup"

git mv foldername tempname && git mv tempname folderName (在大小写不敏感的系统中，如windows，重命名文件的大小写,使用临时文件名)

git mv -n foldername folderName (显示重命名会发生的改变，不进行重命名操作)

### stash

把当前的工作隐藏起来，等待以后恢复现场后继续工作

Git stash

查看所有被隐藏的文件列表

Git stash list

恢复被隐藏的文件，但内容不删除

Git stash apply

删除文件

Git stash drop

恢复文件的同时，也删除文件

Git stash pop

### rev-parse

显示.git目录所在的位置：

git rev-parse --git-dir

显示工作区根目录：

git rev-parse --show-toplevel

显示当前目录，相对于工作区根目录的相对目录：

git rev-parse --show-prefix

显示从当前目录到工作区根目录的深度：

git rev-parse --show-cdup

显示引用对应的提交ID

git rev-parse refs/heads/master

git rev-parse master

git rev-parse HEAD

### …

## 图形界面工具

## 相关资源

最简单的GIT教程 <http://www.bootcss.com/p/git-guide/>

GIT入门 <https://githowto.com/setup>

安装GIT <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

GIT 使用资源 <https://www.kernel.org/pub/software/scm/git/docs/howto-index.html>

GIT <http://freeplane.sourceforge.net/wiki/index.php/Git_howto>

GIT软件windows <https://git-for-windows.github.io/>

<https://git-scm.com/download/win>

<http://blog.jobbole.com/25775/>

GIT中文教程

<http://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000>