Git Basic

# 概述

Git本质就是一个文件hash 索引tree。 基本的操作流程如下：

@git add -A 
Ocal File 
git reset 
Ogit commit -m "update to VI.I" 
git status 
git log 
Compare change 
View publish 
Tenp Area 
agit commit -m "update to v1.2" 
@git commit - 
m "VI.O" 
NULL 
repository 
Versionl.e 
Versionl .1 
@git reset 
@git reflog 
git reset —h rd HEADAA . 
OR git reset - hard HEADQ 
Version1.2 
-hard 34fadf4 
Ogit init 
update... not commit 
gitreset - -hard HEAD 
sit checkout •file>_._ 
—hard HEAP'V 
Sget reset - 
87fa803 
34fadf4 
87fa803 
HEADg(1y: 
HEADß{2}: 
HEADß{3} : 
reset ; 
commit : 
commi t 
commi t 
moving to headh 
VI. I 
(ini tial) : 

# Git Internal

Git 总共有4类对象

(1) blob 对象

这是真正的数据对象，只是保存数据。数据用libz做了压缩。

(2) tree 对象

blob对应文件的内容，tree对象可以理解为目录，它的树节点信息包含文件名，hash-key，文件类型、权限等等。这样就可以组织整个需要控制文件的结构。

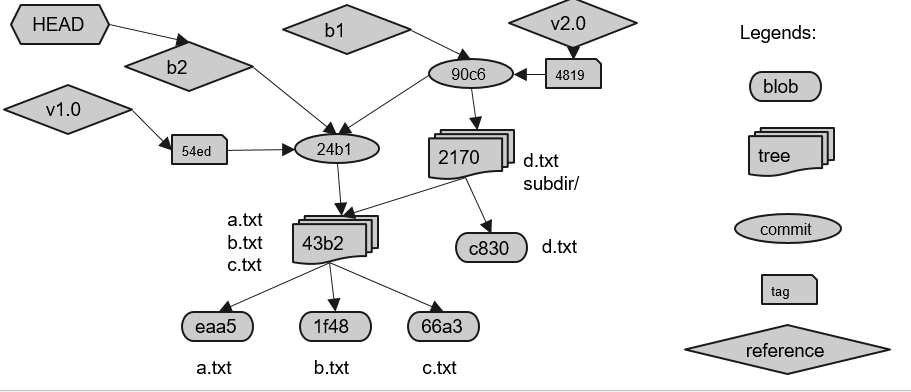
(3) commit 对象

介绍tree对象时，提到过commit,只有在commit的时候，才会根据index记录的内容生成tree对象，那么commit对象里只有两个类容:1.代表工作目录的tree对象的key，上一个commit的key。

(4) Tags 对象

指向某次commit hash 的一个引用。

示意图如下：



# 协同开发

一般clone之后，建一个本地分支，然后在本地分支上开发，当要递交到服务器时，流程如下：

方式一：

git commit –m "summary of changes to features or bugs."

git push origin HEAD:dev

如果此时git push 失败：提示本地修改落后于远程的仓库! 应使用pull同步远程仓库(注意：添加 --rebase 参数，如果代码有冲突，需先在本地解决冲突), 然后再次尝试推送即可。

git pull --rebase

如果不想每次都添加rebase参数，可配置git全局配置配置文件：

*git config --global pull.rebase true*

当然也可以采用下面列举的另外两种开发流程，如下：

方式二：

先stash 本地的修改，然后git pull(默认策略是merge)，然后将本地的修改应用到最新的代码之上（此时同样可能有冲突，请本地解决）。最后commit && push 即可。

git stash

git pull origin dev

// 如果有冲突本地解决冲突

git stash pop

git commit –m "commit msg!"

git push orgin localdev:dev

方式三：

采用主动merge的方式（feature -> dev）：

server 
local 
origin/master 
origin/dev 
clone 
dev 
Q branch 
feature_log 
origin/dev 
_@ git pull _ 
----------other developer modi-- 
dev 
S git merge - -squash feature_log 
M ay need to 
-O new featur 
resolve conflicts! 

## 工作中常用的命令

指定克隆远程的某一个分支 (同时递归更新所有子模块)

$ git clone -b dev --recurse <http://192.168.50.248/xielianghui/eddid_common.git>

把本地分支推送到服务器

$ git push origin LocalBranchName:RemoteBranchName

检出远程分支到本地

$ git checkout -b 本地分支名 origin/远程分支名

重写本地commit消息

$ git commit **--amend** -m "YOUR-NEW-COMMIT-MESSAGE"

设置别名

git config --global alias.st 'status'

## 分支合并

合并分支

缺省的 merge 会在 最后的commit记录之上,

自动生成一个不含任何修改的 merge commit记录；通过--no-commit 可以不生成该记录。

git merge **--no-commit** new-feature

压缩所有修改为一次修改

git merge **--squash** new-feature

摘取某一次递交

git cherry-pick commit-id

## 子模块

更新子模块到对应版本：

git submodule update --init

# 如何在项目中，升级子模块？

下面以eddid\_quote\_center项目的子模块eddid\_common为例：

操作流程：

1 到eddid\_common子模块的目录/src/eddidCommon下，把eddid\_common当作一个git仓库进行正常更新操作。这里的提交会推送到子模块的git仓库。

2 然后退出子模块目录，把/src/eddidCommon看作一个文件的修改 提交并推送到eddid\_quote\_center项目即可

git add src/eddidCommon

git commit -m "upgrade eddid\_common: 填写修改内容摘要"

git push origin HEAD:dev

<完>