## 安装

linux下，apt-get install，注意还需要设置user.name & user.email

git config –global user.name “xx”

git config –global user.email “xx”

linux生成公钥：

ssh-keygen –t rsa –C “email”

在linux下，在～/.ssh/目录下配置config文件

|  |
| --- |
| Host github.com  User [xx@xx.com](mailto:xx@xx.com)（github登陆邮箱）  Hostname ssh.github.com  PreferredAuthentications publickey  IdentityFile ~/.ssh/id\_rsa  Port 443 |

## 原理

**工作区**：文件夹（clone或者init的文件夹）

**版本库**：工作区下的.git文件夹

**暂存区**：包含于版本库，执行add命令之后，修改存在于暂存区

**分支**：包含于版本库，默认有一个master分支，执行commit之后，修改存在于master分支。

分支

暂存区

版本库

工作区

## 查看状态

git status

git diff

## 回退版本、撤销修改

版本回退

|  |  |
| --- | --- |
| git log | 查看版本 |
| git log –graph | 图形化显示版本线 |
| git reflog | 查看历史操作 |
| git reset --hard HEAD^|HEAD^^|HEAD~100 | 回退前1|2|100版本 |

撤销修改

|  |  |
| --- | --- |
| git checkout – file | 将修改撤销到上一次add或者commit时的状态 |
| git reset HEAD file | 将暂存区的修改撤销 |

## 分支

|  |  |
| --- | --- |
| git branch | 查看本地分支 |
| git branch –a | 查看远程分支 |
| git checkout xxx | 切换分支 |
| git branch xxx | 创建分支 |
| git checkout –b xxx | 创建分支并切换到该分支 |
| git branch –D xxx | 强制删除分支 |

## 远程（优先使用clone）

|  |  |
| --- | --- |
| git remote -v | 查看fetch和push实际地址 |
| git push origin master | 向origin默认分支推送本地maser分支 |
| ~~git branch –b 本地分支 origin/远程分支~~  git branch –set-upstream 本地分支 origin/远程分支 | 关联本地分支和远程分支  优先使用这个：   1. 在gihub网站create 2. git remote add origin url 3. git push -u origin master（第一次加上-u，把本地master关联远程master，以后就不用   -u）  重要：创建之后一定先pull，如果先在本地更改，再push会出问题。 |

## 合并分支

|  |  |
| --- | --- |
| git merge xxx | 使用快进模式合并xxx到当前分支 |
| git merge –no-ff xxx | 不使用快进方式合并分支 |

区别:

假设将dev分支合并到master分支。

快进方式合并分支，将master指针移动到dev指针的位置。使用git log –graph看不出合并操作。

普通合并方式，将master和dev都移动到一个汇合点，可以看出合并操作。

## 中断

|  |  |
| --- | --- |
| git stash | 保存当前分支现场 |
| git stash list | 查看中断前保存的现场 |
| git stash apply xxx | 恢复现场xxx  需要手动删除这个保存下的现场  git stash drop xxx |
| git stash pop | 恢复栈顶 |

## 原理介绍

<https://git-scm.com/book/zh/v1>

### git对象

blob、tree、commit、ref

git add之后会在.git/objects下存储tree对象或者blob对象，当只有一个文件时对象类型为blob，文件结构复杂时存储为tree对象，commit对象一定是tree类型。

tree{

blob；

tree\*

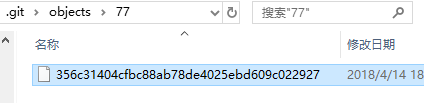
}

### git add

在.git/objects下生成文件夹（对于git来说是一种对象），具体过程是：

“git hash-object -w 文件内容”生成40字符的sha-1校验符，该校验符的前两位作为文件夹名称，后面的38位作为文件名称。





使用cat-file -p查看对象内容：



使用cat-file -t查看对象类型



### **git commit**

会在.git/objects下存储tree对象，该对象还包括一个指向tree对象的指针，指向上一个版本的commit，并更新.git/refs/heads/branch-name文件的内容。

### git log

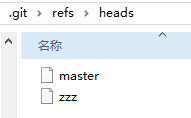
查看.git/HEAD文件（ref类型对象）：



其内容引用了refs文件夹下的内容：

表示当前HEAD变量引用refs/heads/zzz，zzz表示当前branch名称。

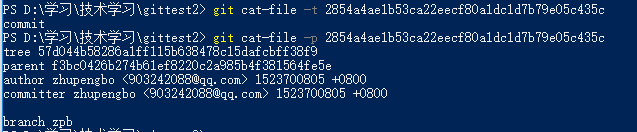
refs/heads下有文件夹：

表示当前有两个分支master和zzz。

查看zzz内容：

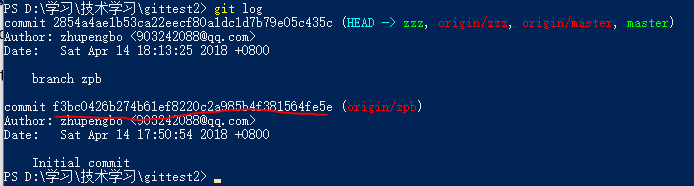
是一个sha-1哈希值，代表objects下的一个对象，git commit时更新。

查看该对象：



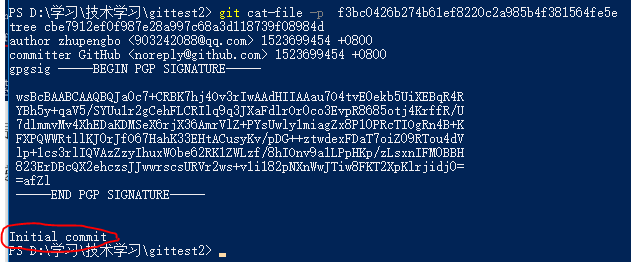
（使用cat-file -t看出其类型为commit，使用cat-file -p查看其内容，parent即就是指向上一次commit对象。）

使用git log查看：



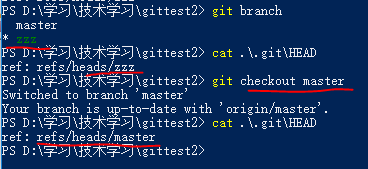
确实是当前commit对象的parent。

然后递归的去查找parent，直到：



### git checkout

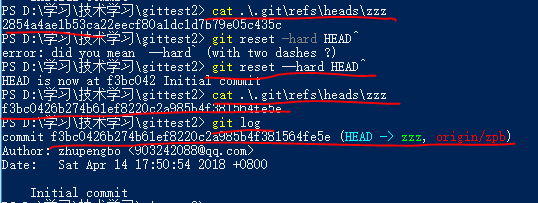
改变HEAD文件内容：

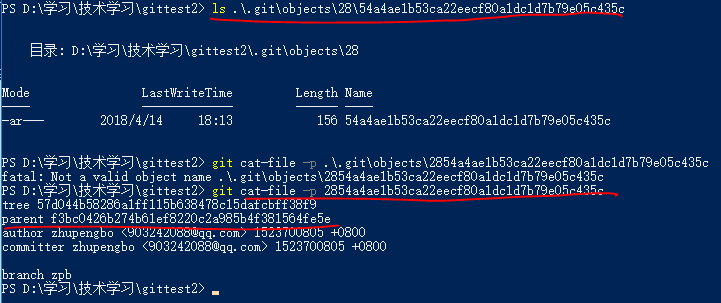


### git reset

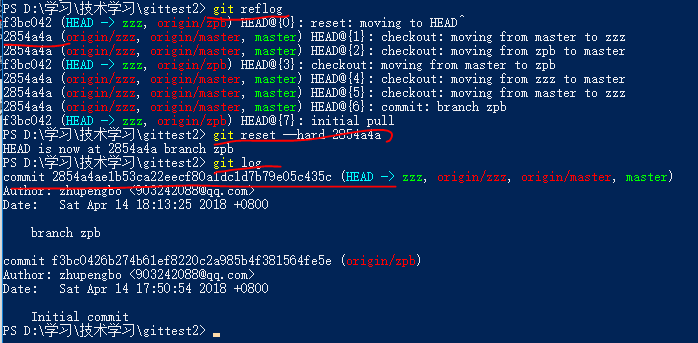
用于回退版本。

仅仅改变.git/refs/heads/branch-name内容，但是.git/objects下的对象不改变，commit对象之间的关系仍在。



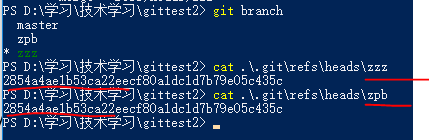


额外说一下，如果reset之后想要回到最先前的版本，即就是2854xxx那个，可以通过git reflog查看之前的操作，可以得到reset之前的版本号，然后reset就可以了：



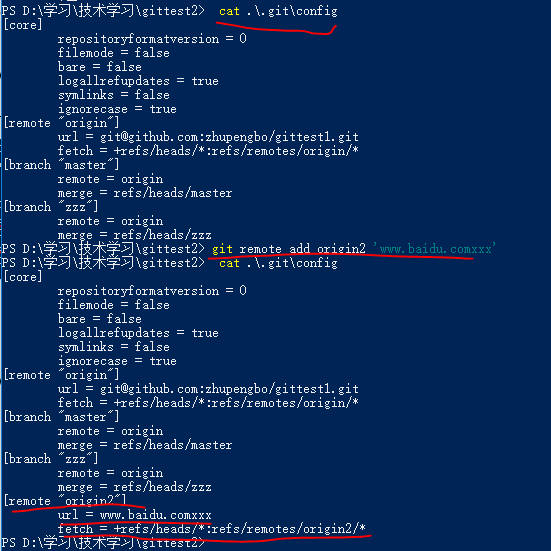
### git branch xxx

在.git/refs/heads下新建文件，文件名为xxx，并设置内容为新建branch时的HEAD引用的值



### git remote add xxx url

在.git/config文件添加内容。



### git push -u origin branch-name

关联当前分支到远程分支，第一次使用这么复杂，之后直接git push。添加内容到.git/config。

