

git

(1)什么是git

》git是一个"分布式"的版本控制工具

》git的作者是Linux之父: Linus Benedict Torvalds,当初开发git仅仅是为了辅助Linux内核的开发(管理源代码)

[Linus Benedict Torvalds\\_百度百科](#)



姓名: 李纳斯

生日: 1969年12月28日 成就: Linux操作系统内核

简介: 李纳斯, 被誉为Linux之父, 芬兰人, 当他还是大学生..

[个人简历](#) [Linux系统简介](#) [主要成就](#) [评价](#) [人物经历](#)

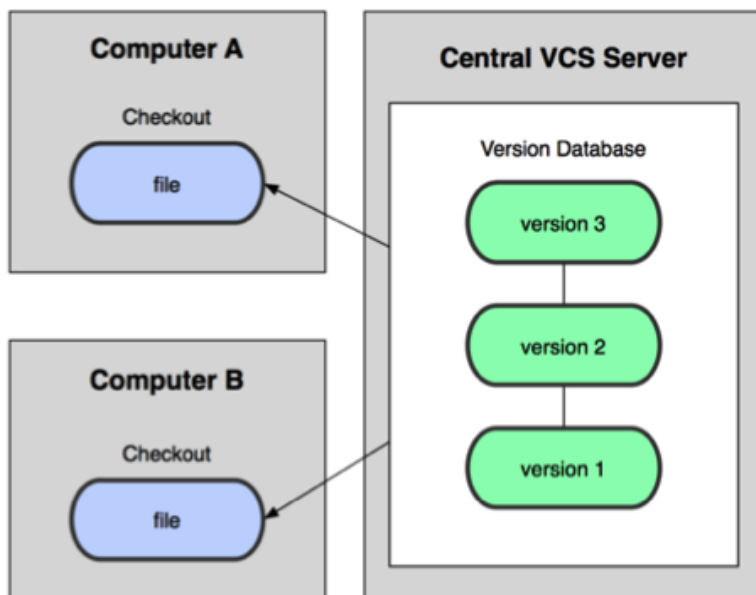
[baike.baidu.com/](#) 2013-05-26 ▾

》git 在国外已经很普及, 在国内已经慢慢普及了。

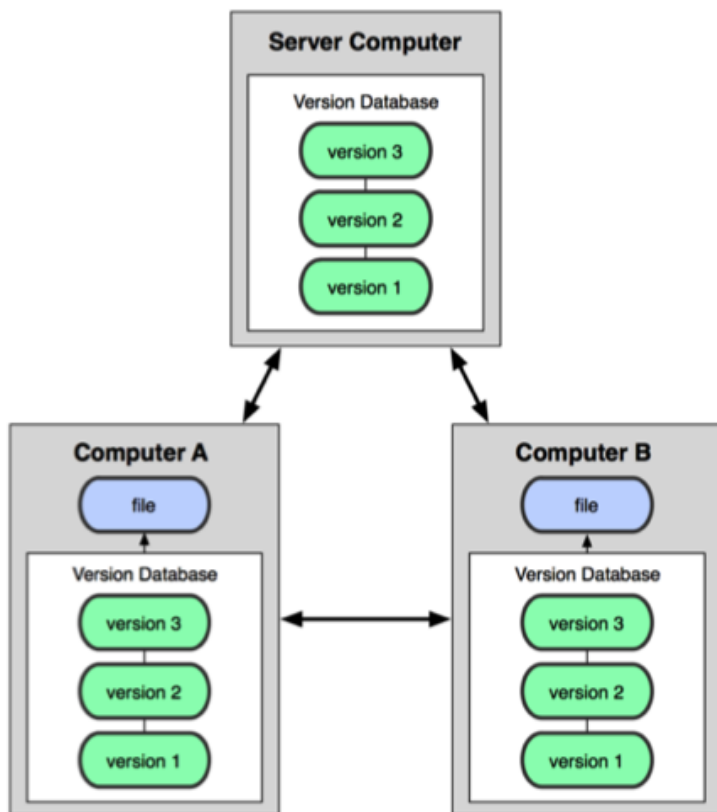
(2)git与svn对比

①结构:svn是"集中式"的版本控制, git是"分布式"版本控制  
"集中式"与"分布式"版本控制的区别

集中式



分布式



- ②速度:多数情况下git的速度比svn快
- ③分支:svn的分支比较笨拙,git可以轻松的建立无限个分支
- ④旧版本的svn会在每个文件夹下建立一个.svn,git只会有根目录下拥有一个.git

### (3)svn与git的工作流程

》svn

- ①从服务器获取最新版本的代码
- ②写了一天的代码后,下班前把代码上传到服务器
  - /\*
  - \*上传时要先从服务器下载最新代码,因为有可能其它事件对代码进行了修改
  - \*然后与自己的代码整合,运行无问题再上传你的代码
  - \*/
- ③第二天上班前,再次下载代码,有可能别人加班,很晚才提交代码
- ④接着开始今天的代码开发,重复①②③④

》git

- ①从服务器获取代码(clone),我们称为克隆
- ②修改代码后提交到本地
- ③当有需要时再提交到服务器

### (4)git的命令实战

#### 1)帮助命令

- 如果对一个命令不清楚,可以输入 'git help'来了解

#### 2)初始化一个仓库

- 仓库用来存放各个版本的文件信息
- 建立一个空的文件夹,命名为Weibo,假设用来存放微博项目的
- 在终端切换到Weibo目录下,输入 'git init'命令
- 在终端输入 'defaults write com.apple.finder AppleShowAllFiles Yes && killall Finder' 看到Weibo下有个.git的隐藏文件夹,这个文件非常重要,没了就没有版本控制了。

#### 3)提交版本信息

- 在Weibo下添加一个main.m文件,终端输入 'touch main.m' 即可,然后添加一行文字
- 在终端输入下面命令,目的是将main.m文件,提交到版本库git中
  - git commit -m '第一次初始化项目'

- 输入上面的命令后，会有如下图的红色提示，这是因为提交前要把.m文件纳入版本控制(暂缓区)

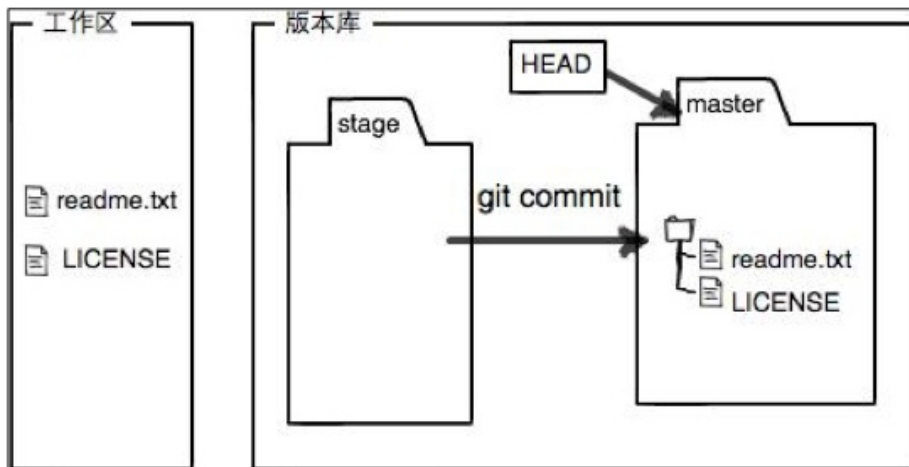
```
yongdembp:WeiBo Fung$ touch main.m
yongdembp:WeiBo Fung$ git commit -m '第一次初始化项目'
On branch master

Initial commit

Untracked files:
  main.m

nothing added to commit but untracked files present
yongdembp:WeiBo Fung$
```

理解 git 的工作原理,stage 就是暂缓区



- 输入 '`git add .`' 把当前目录下的所有文件纳入版本控制,然后再输入上面的 `git commit` 命令，这个我们就看到版本提交成功了，如图

所有新添加的文件都要添加纳入到版本控制后才能提交

```
yongdembp:WeiBo Fung$ git add .
yongdembp:WeiBo Fung$ git commit -m '第一次初始化项目'
[master (root-commit) 1c8723e] 第一次初始化项目
1 file changed, 2 insertions(+)
create mode 100644 main.m
yongdembp:WeiBo Fung$
```

#### 5) 查看文件的状态

- 在 `main.m` 文件中再添加一行文字
- 输入 '`git status`', 我们会看到一个红色的文件，这代表这个文件是被修改过的

```
yongdembp:WeiBo Fung$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   main.m

no changes added to commit (use "git add" and/or "git commit -a")
```

- 修改过的文件要提交前，也要把修改的文件纳入版本控制后才可以使⤵用 `commit` 命令

```

yongdembp:WeiBo Fung$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   main.m

no changes added to commit (use "git add" and/or "git commit -a")
yongdembp:WeiBo Fung$ git commit -m '添加第二行文字' 1.修改文件后, 第一次commit不成功
On branch master
Changes not staged for commit:
        modified:   main.m

no changes added to commit
yongdembp:WeiBo Fung$ git add . 2.把修改的文件纳入版本控制
yongdembp:WeiBo Fung$ git status 3.纳入版本控制器, 发现modified的变录了, 这里可以提交版本
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   main.m

yongdembp:WeiBo Fung$ git commit -m '添加第二行文字' 3.这次提交版本就成功了
[master 7ff5fc0] 添加第二行文字
1 file changed, 3 insertions(+), 1 deletion(-)

```

#### 6) 配置 git 用户

- 配置用户名  
`git config "user.name" zhangsan` 用于查看谁修改了文件
- 配置邮箱  
`git config "user.email" zhangsan@itcast.cn` 用于多人开发, 邮件通知
- 查看当前的配置

```
cat .git/config
```

```

yongdembp:WeiBo Fung$ git config "user.name" zhangsan
yongdembp:WeiBo Fung$ git config "user.email" zhangsan@itcast.cn
yongdembp:WeiBo Fung$ cat .git/config
[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
    ignorecase = true
    precomposeunicode = true
[user]
    name = zhangsan
    email = zhangsan@itcast.cn

```

当前的git帐号和邮箱, 只对此项目有效  
意思是局部

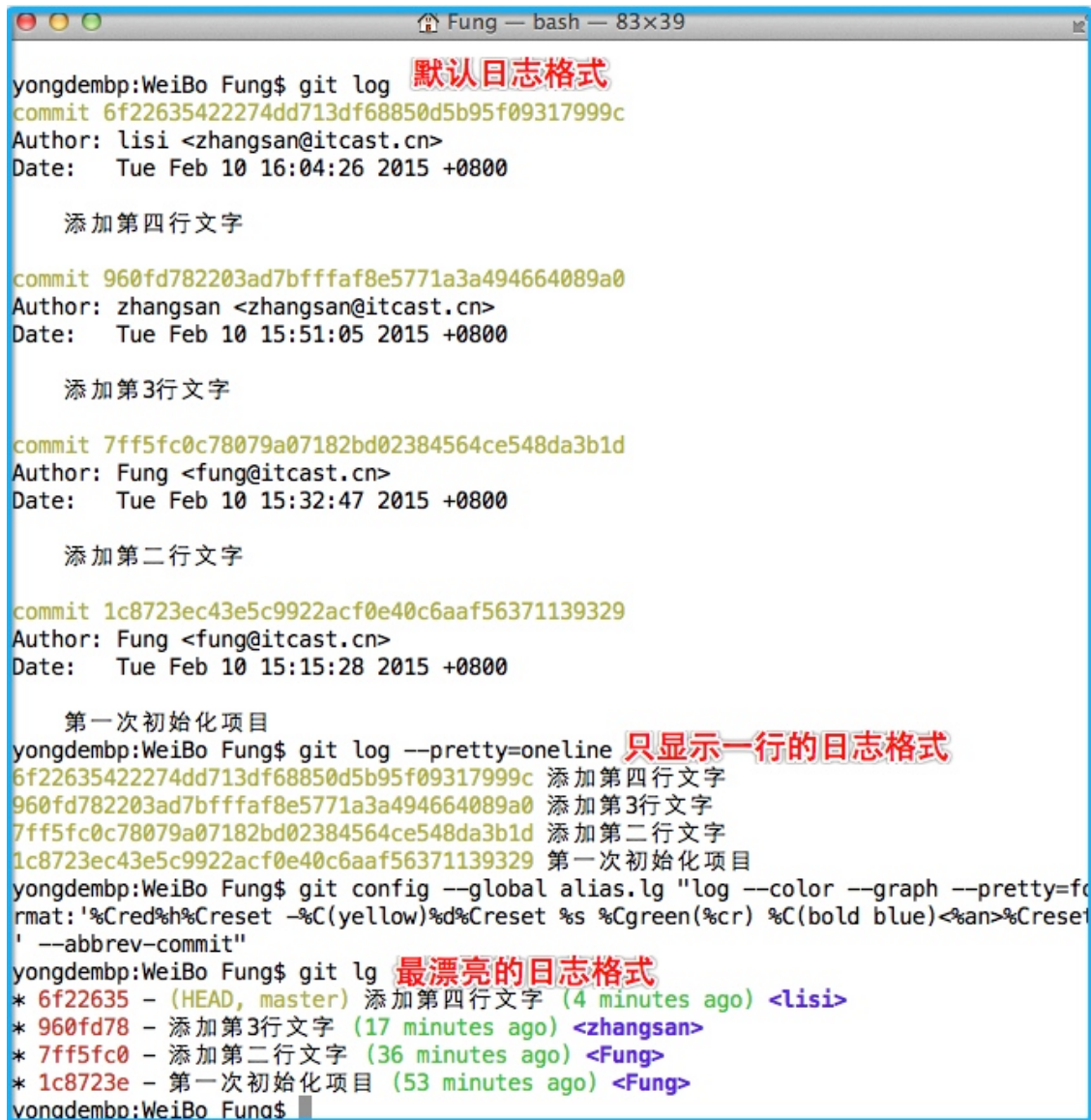
- 再次修改文件提交后, 显示修改的用户将是当前配置的

#### 7) 给 config, status, commit 命令 配置别名

- `git config -l` 查看配置
- `git config alias.cfg config` 给 config 取别名
- `git config alias.st status` 给 status 取别名
- `git config alias.ct "commit -m"` 给 "commit -m" 取别名

## 8) 配置日志格式

- `git log` // 默认日志
- `git log --pretty=oneline` // 显示一行日志
- `git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"` // 最漂亮的日志



```
Fung — bash — 83x39
yongdembp:WeiBo Fung$ git log 默认日志格式
commit 6f22635422274dd713df68850d5b95f09317999c
Author: lisi <zhangsan@itcast.cn>
Date: Tue Feb 10 16:04:26 2015 +0800

    添加第四行文字

commit 960fd782203ad7bfffaf8e5771a3a494664089a0
Author: zhangsan <zhangsan@itcast.cn>
Date: Tue Feb 10 15:51:05 2015 +0800

    添加第3行文字

commit 7ff5fc0c78079a07182bd02384564ce548da3b1d
Author: Fung <fung@itcast.cn>
Date: Tue Feb 10 15:32:47 2015 +0800

    添加第二行文字

commit 1c8723ec43e5c9922acf0e40c6aaf56371139329
Author: Fung <fung@itcast.cn>
Date: Tue Feb 10 15:15:28 2015 +0800

    第一次初始化项目
yongdembp:WeiBo Fung$ git log --pretty=oneline 只显示一行的日志格式
6f22635422274dd713df68850d5b95f09317999c 添加第四行文字
960fd782203ad7bfffaf8e5771a3a494664089a0 添加第3行文字
7ff5fc0c78079a07182bd02384564ce548da3b1d 添加第二行文字
1c8723ec43e5c9922acf0e40c6aaf56371139329 第一次初始化项目
yongdembp:WeiBo Fung$ git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"
yongdembp:WeiBo Fung$ git lg 最漂亮的日志格式
* 6f22635 - (HEAD, master) 添加第四行文字 (4 minutes ago) <lisi>
* 960fd78 - 添加第3行文字 (17 minutes ago) <zhangsan>
* 7ff5fc0 - 添加第二行文字 (36 minutes ago) <Fung>
* 1c8723e - 第一次初始化项目 (53 minutes ago) <Fung>
vonadembp:WeiBo Fung$
```

## 9) 版本回退

- 恢复前一个版本 `git reset --hard HEAD^`
- 恢复前两个版本 `git reset --hard HEAD^^`
- 恢复前N版本 `git reset --hard HEAD~N` // N是数字
- 恢复指定版本(`git lg`) `git reset --hard 版本号`
- 查看指令使用记录 `git reflog`

## 10) 查看文件的不同

- 修改文件
- 输入 `git diff` 文件名

## 11) 删除文件

- `git rm` 文件名

总结，不管是添加、修改、删除文件，都要选把文件纳入到版本控制中后，才可以提交，最为安全

(5)git 的工作原理

- ① 工作与暂缓区
- ②Head 当前版本号
- ③主分支(当前开发的版本)

(6)创建分支

- ① 查看所有分支 `git branch`
- ② 从当前版本创建分支 `git branch v1` `//branch` 后面不带参数, 代表查看所有分支
- ③ 切换分支 `git checkout v1`
- ④ 合并分支 `git merge v1`  
合并过程中有冲突, 要手动解决
- ⑤ 删除分支 `git branch -d v1`

(7)xcode 创建 git 项目 以下操作要在 xcode 中熟练使用

- ①xcode 中 git 的提交
- ②xcode 中的历史查看
- ③xcode 中文件名后的 M/A 代理修改和添加
- ④xcode 中文件修改后与之前版本间的不同之处查看
- ⑤xcode 创建分支
- ⑥xcode 合并分支, 解决冲突
- ⑦xcode 删除分支

总结:以上的所有版本控制信息都在本地