[1. 安装git 2](#_Toc914884)

[2. 创建版本库 2](#_Toc914885)

[工作区和暂存区 3](#_Toc914886)

[3. 时光穿梭机 3](#_Toc914887)

[版本回退 3](#_Toc914888)

[撤销修改 4](#_Toc914889)

[删除文件 4](#_Toc914890)

[4. 远程仓库 4](#_Toc914891)

[添加远程仓库 5](#_Toc914892)

[从远程仓库克隆 5](#_Toc914893)

[5. 分支管理 5](#_Toc914894)

[创建和合并分支 5](#_Toc914895)

[解决冲突 6](#_Toc914896)

[分支策略 6](#_Toc914897)

[Bug分支 6](#_Toc914898)

[Featuer 分支 7](#_Toc914899)

[多人协作 7](#_Toc914900)

[推送分支 7](#_Toc914901)

[抓取分支 7](#_Toc914902)

[6. 标签管理 8](#_Toc914903)

[创建标签 8](#_Toc914904)

[操作标签 8](#_Toc914905)

[7. 配置 9](#_Toc914906)

[配置别名 9](#_Toc914907)

[配置文件 9](#_Toc914908)

# 安装git

Sudo apt-get install git

Git config –global user.name yourname

Git config –global user.email youremail

# 创建版本库

1. 创建一个空目录 learngit

￥mkdir learngit

￥cd learngit

1. 通过init命令把这个目录变成git可以管理的目录

$git init

1. 把文件添加到仓库

$git add readme.txt 把文件添加到仓库

$git commit –m “wrote a readme file” 把文件提交到仓库。 -m后面是提交的说明。输入有意义的内容。这样可以从历史记录里找到改动记录。可以多次add不同额文件，一次提交。

## 工作区和暂存区

工作区 workingdirectory ： 这是你在电脑上看到的目录，比如前面的learngit目录

暂存区 repository ： 工作区隐藏目录.git 。这是git的版本库，不是工作区。

Git版本可以存放了很多东西，其中最重要的成为stage（或者index）的暂存区，还有git为我们自动创建的第一个master，以及指向master的一个指针head。把文件往git版本库添加的时候，分为两步

1. 用git add 把文件加进去，实际上是吧文件添加到暂存区。
2. Git commit 提交，实际上是把暂存区的内容提交到当前分支。当我们创建git版本库时，git自动穿件了master分支，git commit 就是往master分支提交
3. 随时可以用git status查看状态

# 时光穿梭机

$git diss filename 查看对文件做了什么修改

$git diff HEAD –filename 查看工作区和版本库里面最新版本的区别

$git status 随时掌握工作区的状态。如果git status告诉你有文件被修改过时，可以用git diff查看修改的内容

## 版本回退

把当前版本回退到上一个版本，可以使用git reset

$git reset –hard HEAD ^

把当前版本回退到上个版本，也就是HEAD^^,往上100个版本，写成HEAD~100，也可以指定回退到指定版本。

$git reset --hard commit\_id

$git log 查看提交的历史记录

$git log --pretty=oneline 可以减少输出

$git reflog 如果想恢复到新版本，git 提供了一个可以用来记录你的一次命令。

小结： HEAD指向的版本库就是当前的版本，因此git允许我们在历史穿梭，使用git reset --hard commit-id。 穿梭之前可以用 git log查看提交历史，以便确定要回到哪个版本。

要重返未来，用git reflog 查看命令历史，以便确定回到未来哪个版本。

## 撤销修改

$git checkout readme.txt 把readme.txt在工作区的修改全部撤销

$git reset HEAD readme.txt 把readme.txt 暂存区的修改撤销，放回工作区

## 删除文件

一般情况下，你通常直接在文件管理器中吧文件给删了，或者rm命令给删了。这个时候，git知道你删了文件，因此工作区和版本库就不一致了。Git status 命令会立刻告诉你哪些文件被删除了。现在你有两个选择

1. 确实要从版本库中删除该文件

$git rm test.txt

$git commit –m “remove test.txt”

1. 另一种情况是删错了，回到最新版本

$git checkout test.txt

# 远程仓库

由于你本地的git仓库跟github之间的传输是通过ssh加密的，所以需要一点设置。

1. 创建ssh key 在用户主目录下

$sshkeygen 一路回车，在用户家目录下面有两个文件，id.rsa id.rsa.pub 这是密钥对。 Id.rsa是私钥，不能泄露. Id.rsa.pub是公钥。

1. 打开github，打开account setting SSH key 点add key 填写任意title，在key文本框里填入id.rsa.pub里面的内容即可。

## 添加远程仓库

1. 登录github，然后在右上角找到create new repo 按钮，创建一个新仓库
2. 在repo name 中输入learngit，其他保持默认
3. 关联远程仓库，在本地learngit目录下执行$git remote add origin [git@github.com:/XXX.git](mailto:git@github.com:/XXX.git)
4. 把本地仓库所有内容推送到远程仓库上

第一次推送： ￥git push -u origin master

之后推送： $git push origin master

## 从远程仓库克隆

$git clone [git@github.com:/xxx.git](mailto:git@github.com:/xxx.git)

当前目录下会生成仓库所有的目录

# 分支管理

## 创建和合并分支

$git branch 查看分支

创建分支 $git branch <name>

切换分支$ git checkout <name>

创建并切换分支 $git checkout –b <name>

合并某分支到当前分支 $git merge <name>

删除分支 $git branch -d <name>

## 解决冲突

用带参数的git log也可以看到分支合并的情况

$git log --graph --pretty=oneline --abbrev-commit 当git无法自动合并分支时，就必须解决冲突再合并。 用git log --graph可以看到分支合并图

## 分支策略

通常合兵分支时，如果可能，git会使用Fastforward模式。在这种模式下，删除分支后，会丢到分支信息

如果要禁用fastforward模式，git就会在merge时生成一个新的commit，这样就可以从历史上看到分支的信息了

$git merge --no-ff -m “merge with no ff” dev 合并dev，采用noff模式

分支策略

在实际开发中，我们的基本原则：

1. Master 应该非常稳定，也就是用来发布新版本，平时不能再上面干活
2. 干活都在dev分支上，也就是说dev分支时不稳定的。在某个时候，比如01版本发布，再把dev分支合并到master上。在master发布01版本
3. 每个人都有自己的分支，是不是地往dev分支上合并
4. 分支合并时，加上—no-ff参数就可以用普通模式合并，合并后能看出来曾经做过的合并。

## Bug分支

Git还提供了一个stash功能，可以把当前工作现场存储起来，等以后恢复现场继续使用:

$git stash

查看保存的内容：

$git stash list

恢复现场工作：

$git stash apply 但是恢复后 git stash内容并不删除，你需要使用git stash drop来删除，或者 git stash pop

## Featuer 分支

开发一个新的featuer，最好新建一个分支

如果要丢弃一个没有被用过的分支，可以通过 $git branch -D<name> 强行删除

## 多人协作

要查看远程仓库的信息，用git remote -V 显示更详细的信息

$git remote -V

上面显示了可以抓取和推送的origin的地址，如果没有推送权限，就看到push地址

### 推送分支

推送时要指定分支，这样，git就会把该分支推送到对应的远程分支上

$git push origin master

如果要推送到其他分支上，比如dev

$git push origin dev

### 抓取分支

多人协作的工作模式，同城是这样的

1. 可以试图使用git push origin branch-name 推送自己的修改
2. 如果推送失败，则因为远程分支比你本地的新。首先需要 git pull试图合并
3. 如果合并有冲突，则解决冲突后，再提交本地
4. 如果没有冲突，或解决冲突后，再用git push origin branch-anme 来推送
5. 如果git pull 提示“no tracking information”则说明本地分支跟远程分支的链接没有创建，用命令 $git branch --set-upstream branch-name origin/branch-name

# 标签管理

## 创建标签

命令 git tag <name> 用于新建一个标签，默认为HEAD，也可以指向一个commit id

$git tag -a <tagname> -m “blabla…” 可以指定标签信息

$git tag -s <tagname> -m “blabla” 可以用PGP签名标签

$git tag 查看所有标签

$git show <tagname> 查看标签信息

## 操作标签

$git push origin <tagname> 推送一个本地标签

$git push origin –tags 推送所有本地标签

$git tag -d <tagname> 删除一个本地标签

$git push origin :refs/tags/<tagname> 删除一个远程标签

# 配置

## 配置别名

我们只需要一条命令，告诉git以后st就表示status

$git config --global alias.st status

$git config --global alias.co checkout

配置一个git last 让其显示最后一次提交内容

$git config --global alias.last ‘log -l’ 这样，$git last 就能显示最后一次提交了

## 配置文件

配置git的时候，加上 –global 是针对当前用起作用的，如果不加，则只对当前仓库起作用

每个仓库文件的git配置文件都放在.git/config文件中

[core]

Repisitoryformatversion=0

Filemode=true

Bare=false

Logallrefupdates=true

[alias]

St=status

Last=log -l