# Git使用

### Git最全教程：http://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000

### 创建版本库

选择一个合适的地方，创建一个空目录：

$ mkdir learngit

$ cd learngit

通过git init命令把这个目录变成Git可以管理的仓库：

$ git init

### 把文件添加至版本库

用命令git add告诉Git，把文件添加到仓库：

$ git add readme.txt

用命令git commit告诉Git，把文件提交到仓库：

$ git **commit** -m "wrote a readme file"

### 查看仓库当前状态

$ git status

### 查看仓库文件的difference

$ git diff readme.txt

### 提交修改后的文件

提交修改和提交新文件是一样的两步，第一步是git add

$ git add readme.txt

然后git commit

$ git **commit** -m "add distributed"

### 查看历史记录

$ git log

### 查看每一次提交的commit id

$ git reflog

### 回退到历史版本

在Git中，用HEAD表示当前版本，也就是最新的提交3628164...882e1e0（注意我的提交ID和你的肯定不一样），上一个版本就是HEAD^，上上一个版本就是HEAD^^，当然往上100个版本写100个^比较容易数不过来，所以写成HEAD~100。

$ git re**set** --hard HEAD^

### 再回到未来版本

$ git re**set** --hard [commit id]

Eg：

$ git re**set** --hard 3628164

### 关于SSH KEY

创建SSH Key：

$ ssh-keygen -t rsa -C ["youremail@example.com"](mailto:\"youremail@example.com\")

如果一切顺利的话，可以在用户主目录里找到.ssh目录，里面有id\_rsa和id\_rsa.pub两个文件，这两个就是SSH Key的秘钥对，id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥，可以放心地告诉任何人。

公钥用于设置到git服务器，比如GitHub，私钥用于提交/clone代码时检测安全性，因为git使用非对称加密方式，本地私钥和公钥对应的情况下才能进行上传和clone。

### Git暂存区

Git的版本库里存了很多东西，其中最重要的就是称为stage（或者叫index）的暂存区，还有Git为我们自动创建的第一个分支master，以及指向master的一个指针叫HEAD。



前面讲了我们把文件往Git版本库里添加的时候，是分两步执行的：

第一步是用git add把文件添加进去，实际上就是把文件修改添加到暂存区；

第二步是用git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支。

因为我们创建Git版本库时，Git自动为我们创建了唯一一个master分支，所以，现在，git commit就是往master分支上提交更改。

你可以简单理解为，需要提交的文件修改通通放到暂存区，然后，一次性提交暂存区的所有修改。

### 撤销修改

$ git checkout -- readme.txt

场景1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令git checkout -- file。

场景2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令git reset HEAD file，就回到了场景1，第二步按场景1操作。

场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考[版本回退](http://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/0013744142037508cf42e51debf49668810645e02887691000)一节，不过前提是没有推送到远程库。

### 删除文件

$ git rm test.txt

rm 'test.txt'

$ git **commit** -m "remove test.txt"

如果删错了，因为版本库里还有呢，所以可以很轻松地把误删的文件恢复到最新版本：

$ git checkout -- test.txt

git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。

### 从远程库克隆clone

现在，远程库已经准备好了，下一步是用命令git clone克隆一个本地库：

dh@dh-PC MINGW64 ~

$ cd d:

dh@dh-PC MINGW64 /d

$ cd AndroidStudioProject

dh@dh-PC MINGW64 /d/AndroidStudioProject

$ git clone git@github.com:DangGui-2dfire/DfireDemo.git

Cloning into 'DfireDemo'...

remote: Counting objects: 183, done.

remote: Compressing objects: 100% (64/64), done.

remote: Total 183 (delta 37), reused 171 (delta 34), pack-reused 0

Receiving objects: 100% (183/183), 93.99 KiB | 116.00 KiB/s, done.

Resolving deltas: 100% (37/37), done.

### 创建本地分支

首先，我们创建dev分支，然后切换到dev分支：

$ git checkout -b dev

Switched to a new branch 'dev'

git checkout命令加上-b参数表示创建并切换，相当于以下两条命令：

$ git branch dev

$ git checkout dev

Switched to branch 'dev'

然后，用git branch命令查看当前分支：

$ git branch

\* dev

master

git branch命令会列出所有分支，当前分支前面会标一个\*号。



然后，我们就可以在dev分支上正常提交，比如对readme.txt做个修改，加上一行：

Creating a **new** branch is quick.

然后提交：

$ git add readme.txt

$ git commit -m "branch test"

[dev fec145a] branch test

1 file changed, 1 insertion(+)

### 创建远程分支

$ git push --set-upstream origin dev

### 删除远程分支

$ git push origin :dev

To github.com:DangGui-2dfire/DfireDemo.git

- [deleted] dev

### 切换分支

现在，dev分支的工作完成，我们就可以切换回master分支：

$ git checkout master

Switched to branch 'master'

切换回master分支后，再查看一个readme.txt文件，刚才添加的内容不见了！因为那个提交是在dev分支上，而master分支此刻的提交点并没有变。

### IMG_256

### 删除分支

删除dev分支：

$ git branch -d dev

Deleted branch dev (was fec145a).

### 合并分支

把dev分支的工作成果合并到master分支上：

$ git merge dev

Updating d17efd8..fec145aFast-forward

readme.txt | 1 +

1 file changed, 1 insertion(+)

git merge命令用于合并指定分支到当前分支。合并后，再查看readme.txt的内容，就可以看到，和dev分支的最新提交是完全一样的。

### “储藏”分支现场

当你接到一个修复一个代号101的bug的任务时，很自然地，你想创建一个分支issue-101来修复它，但是，等等，当前正在dev上进行的工作还没有提交，

幸好，Git还提供了一个stash功能，可以把当前工作现场“储藏”起来，等以后恢复现场后继续工作：

$ git stash

Saved working directory **and** index state WIP on dev: 6224937 add merge

HEAD is now at 6224937 add merge

现在，用git status查看工作区，就是干净的（除非有没有被Git管理的文件），因此可以放心地创建分支来修复bug。