#### 01 安装Git

$ git

The program 'git' is currently not installed. You can install it by typing:

su**do** apt-**get** install git

如果你碰巧用Debian或Ubuntu Linux，通过一条sudo apt-get install git就可以直接完成Git的安装，非常简单。

安装完成后，还需要最后一步设置，在命令行输入：

$ git config --global user.name "Your Name"

$ git config --global user.email [email@example.com](mailto:email@example.com)

#### 02 创建版本库

版本库又名仓库，英文名**repository**，你可以简单理解成一个目录，这个目录里面的所有文件都可以被Git管理起来，每个文件的修改、删除，Git都能跟踪，以便任何时刻都

可以追踪历史，或者在将来某个时刻可以“还原”。所以，创建一个版本库非常简单:

(1) 首先，选择一个合适的地方，创建一个空目录：

$ mkdir learngit

$ cd learngit

$ pwd

/Users/michael/learngit

(2) 第二步，通过git init命令把这个目录变成Git可以管理的仓库：

$ git initInitialized empty Git repository **in** /Users/michael/learngit/.git/

#### 03 把文件添加到版本库

在 learngit目录下新建 readme.txt文件，把readme.txt文件添加到 learngit目录：

（1）第一步，用命令git add告诉Git，把文件添加到仓库：

$ git add readme.txt

（2）第二步，用命令git commit告诉Git，把文件提交到仓库：

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

[master (root-**commit**) eaadf4e] wrote a readme file

1 file changed, 2 insertions(+)

**create** mode 100644 readme.txt

# git commit命令，-m后面输入的是本次提交的说明，可以输入任意内容，最好是有意义的，这样你就能从历史记录里方便地找到改动记录。

# git commit命令执行成功后会告诉你，1 file changed：1个文件被改动（我们新添加的readme.txt文件）；2 insertions：插入了两行内容（readme.txt有两行内容）。

# commit可以一次提交很多文件，所以你可以多次add不同的文件，比如：

# $ git add file1.txt

# $ git add file2.txt file3.txt

# $ git commit -m "add 3 files."

#### 04 随时掌握工作区的状态

* 要随时掌握工作区的状态，使用git status命令。
* 如果git status告诉你有文件被修改过，用git diff可以查看修改内容。

如，修改readme.txt文件为：

Git **is** a distributed version control system.

Git **is** free software.

运行git status命令看看结果：

$ 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: readme.txt

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

# 上面的命令输出告诉我们，readme.txt被修改过了，但还没有准备提交的修改。

用git diff命令查看具体修改了什么内容：

$ git diff readme.txt

diff --git a/readme.txt b/readme.txt

index 46d49bf..9247db6 100644

--- a/readme.txt

+++ b/readme.txt

@@ -1,2 +1,2 @@

-Git **is** a version control system.

+Git **is** a distributed version control system.

Git **is** free software.

#### 05 版本回退

用git log命令查看历史记录：

$ git log

commit 1094adb7b9b3807259d8cb349e7df1d4d6477073 (HEAD -> master)

Author: Michael Liao <askxuefeng@gmail.com>

Date: Fri May 18 21:06:15 2018 +0800

append GPL

commit e475afc93c209a690c39c13a46716e8fa000c366

Author: Michael Liao <askxuefeng@gmail.com>

Date: Fri May 18 21:03:36 2018 +0800

add distributed

commit eaadf4e385e865d25c48e7ca9c8395c3f7dfaef0

Author: Michael Liao <askxuefeng@gmail.com>

Date: Fri May 18 20:59:18 2018 +0800

wrote a readme file

git log命令显示从最近到最远的提交日志，我们可以看到3次提交，最近的一次是append GPL，上一次是add distributed，最早的一次是wrote a readme file。

如果嫌输出信息太多，看得眼花缭乱的，可以试试加上--pretty=oneline参数：

$ git log --pretty=oneline

1094adb7b9b3807259d8cb349e7df1d4d6477073 (HEAD -> master) append GPL

e475afc93c209a690c39c13a46716e8fa000c366 add distributed

eaadf4e385e865d25c48e7ca9c8395c3f7dfaef0 wrote a readme file

类似1094adb...的是commit id（版本号）

每提交一个新版本，实际上Git就会把它们自动串成一条时间线。如果使用可视化工具查看Git历史，就可以更清楚地看到提交历史的时间线。

在Git中，用HEAD表示当前版本，也就是最新的提交1094adb...，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。

使用git reset命令回退版本：

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

HEAD **is** now **at** e475afc **add** distributed

# 这样就退回到上一个版本了

#也可以用git re**set** –hard+版本号（版本号只需要前面几位就行）

用git reflog查看命令历史

#### 06 工作区和暂存区

工作区（Working Directory）: 就是你在电脑里能看到的目录

版本库（Repository）: 工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。

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。

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。

# 每次修改，如果不用git add到暂存区，那就不会加入到commit中。

#### 07 撤销修改

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](https://www.liaoxuefeng.com/wiki/896043488029600/897013573512192)一节，不过前提是没有推送到远程库。

#### 08 删除文件

$ git add test.txt #添加text.txt到暂存区

$ git **commit** -m "add test.txt" #把暂存区的内容提交到当前分支

$ rm test.txt # 在工作区（即当前文件夹）删除text.txt文件

# 工作区和版本库就不一致了，工作区没有text.txt，而版本库里还有这个文件

# 现在有两个选择：

# （1）从版本库中删除该文件，那就用命令git rm删掉，并且git commit：

# （2）另一种情况是删错了，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”。

#### 09 添加SSH Key

先将本地Git仓库和GitHub仓库之间的用SSH加密传输：

（1）第1步：创建SSH Key。在用户主目录下，看看有没有.ssh目录，如果有，再看看这个目录下有没有id\_rsa和id\_rsa.pub这两个文件，如果已经有了，可直接跳到下一步。如果没有，打开Shell（Windows下打开Git Bash），创建SSH Key：

$ ssh-keygen -t rsa -C [songchongyong@gmail.com](mailto:songchongyong@gmail.com)

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。

（2）第2步：登陆GitHub，打开“Account settings”，“SSH Keys”页面：

然后，点“Add SSH Key”，填上任意Title，在Key文本框里粘贴id\_rsa.pub文件的内容，点“Add Key”， SSH Key添加完成。

#### 010 添加远程库

已经在本地创建了一个Git仓库后，又想在GitHub创建一个Git仓库，并且让这两个仓库进行远程同步。

首先，登陆GitHub，然后，在右上角找到“Create a new Repository”按钮，创建一个新的仓库，填写Repository name，其他保持默认设置，点击“Create repository”按钮，就成功地创建了一个新的Git仓库。

目前，在GitHub上的这个learngit仓库还是空的，GitHub告诉我们，可以从这个仓库克隆出新的仓库，也可以把一个已有的本地仓库与之关联，然后，把本地仓库的内容推送到GitHub仓库。

现在，我们根据GitHub的提示，在本地的learngit仓库下运行命令：

$ git remote add origin git@github.com:songchongyong/learngit.git

下一步，就可以把本地库的所有内容推送到远程库上：

$ git push -u origin master

由于远程库是空的，我们第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令。

从现在起，只要本地作了提交，就可以通过命令：

$ git push origin master

### 小结

要关联一个远程库，使用命令git remote add origin git@server-name:path/repo-name.git；

关联后，使用命令git push -u origin master第一次推送master分支的所有内容；

此后，每次本地提交后，只要有必要，就可以使用命令git push origin master推送最新修改；

#### 011 从远程库克隆

要克隆一个仓库，首先必须知道仓库的地址，然后使用git clone命令克隆。

Git支持多种协议，包括https，但通过ssh支持的原生git协议速度最快。