在Windows上安装Git

1 在Windows上使用Git，可以从Git官网直接[下载安装程序](https://git-scm.com/downloads" \t "_blank)，然后按默认选项安装即可。

2 设置git账号

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

$ git config --global user.email "email@example.com"

3 创建版本库

$ mkdir learngit

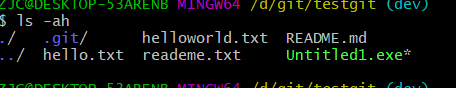
$ cd learngit

$ pwd

/Users/michael/learngit

4 把他变成Git 可以管理的仓库

$ git init



5 第一步，用命令git add告诉Git，把文件添加到仓库：

$ git add readme.txt

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

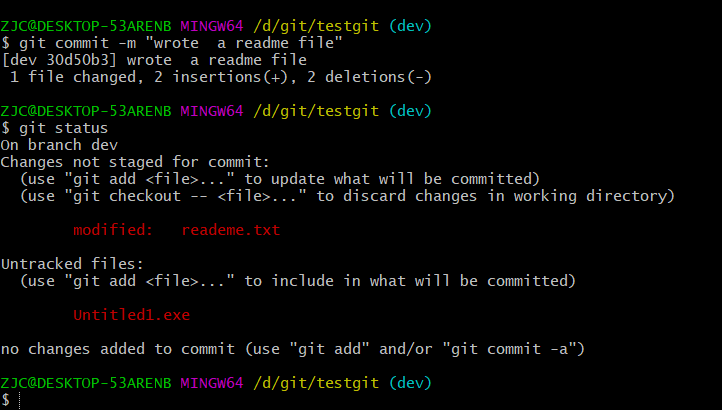
[master (root-**commit**) eaadf4e] wrote a readme file

1 file changed, 2 insertions(+)

**create** mode 100644 readme.txt

6 修改文件

运行git status命令看看结果：



查看修改

$ 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.

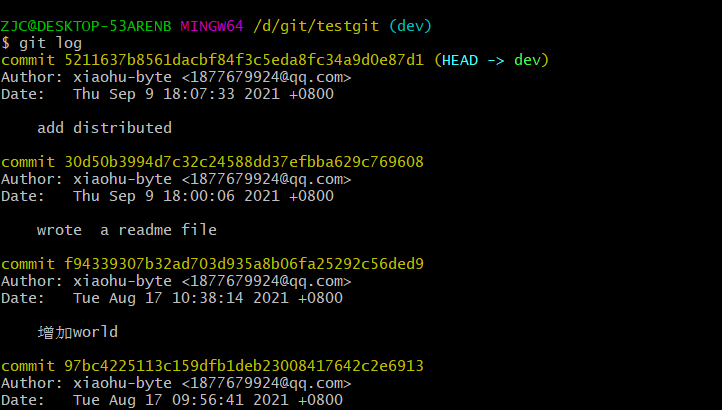
+Git **is** a distributed version control system.

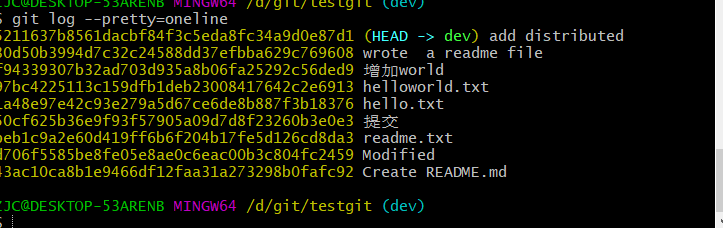
Git **is** free software.

### 小结

* 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。

版本控制系统肯定有某个命令可以告诉我们历史记录，在Git中，我们用git log命令查看：





好了，现在我们启动时光穿梭机，准备把readme.txt回退到上一个版本，也就是add distributed的那个版本，怎么做呢？

第一步还原回退到上个版本

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

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

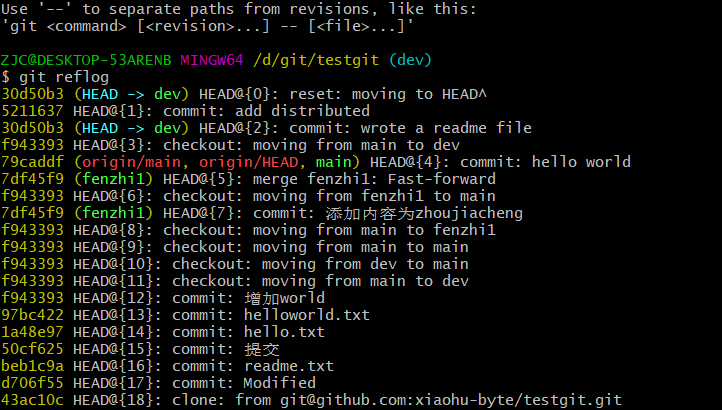
第二步查看

$ cat readme.txt

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

Git **is** free software

Git提供了一个命令git reflog用来记录你的每一次命令：



* HEAD指向的版本就是当前版本，因此，Git允许我们在版本的历史之间穿梭，使用命令git reset --hard commit\_id。
* 穿梭前，用git log可以查看提交历史，以便确定要回退到哪个版本。
* 要重返未来，用git reflog查看命令历史，以便确定要回到未来的哪个版本。

git status:

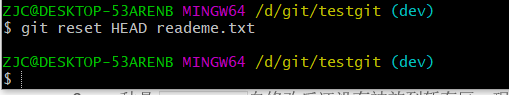
作两次对比，工作区 vs 暂存区 以及 暂存区 vs 仓库，并将两次对比的结果显示在输出中。哪个对比没差别，就不显示。两个对比都没差别，就显示working tree clean。

git diff：

上面已经总结了。

1. git diff：是查看working tree与index的差别的。
2. git diff --cached：是查看index与repository的差别的。
3. git diff HEAD：是查看working tree和repository的差别的。其中：HEAD代表的是最近的一次commit的信息。
4. $ git checkout -- readme.txt
5. 命令git checkout -- readme.txt意思就是，把readme.txt文件在工作区的修改全部撤销，这里有两种情况：
6. 一种是readme.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；
7. 一种是readme.txt已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。
8. 总之，就是让这个文件回到最近一次git commit或git add时的状态。
9. 现在，看看readme.txt的文件内容：

git reset命令既可以回退版本，也可以把暂存区的修改回退到工作区。当我们用HEAD时，表示最新的版本。



### 小结

又到了小结时间。

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。

1 删除文件

$ git add test.txt

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

[master b84166e] **add** test.txt

1 file changed, 1 insertion(+)

**create** mode 100644 test.

2 查看

$ git status

On branch master

Changes not staged for commit:

(use "git add/rm <file>..." to **update** what will be committed)

(use "git checkout -- <file>..." **to** discard changes **in** working directory)

deleted: test.txt

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

现在你有两个选择，一是确实要从版本库中删除该文件，那就用命令git rm删掉，并且git commit

$ git rm test.txt

rm 'test.txt'

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

[master d46f35e] remove test.txt

1 file changed, 1 deletion(-)

**delete** mode 100644 test.txt

$ git checkout -- test.txt

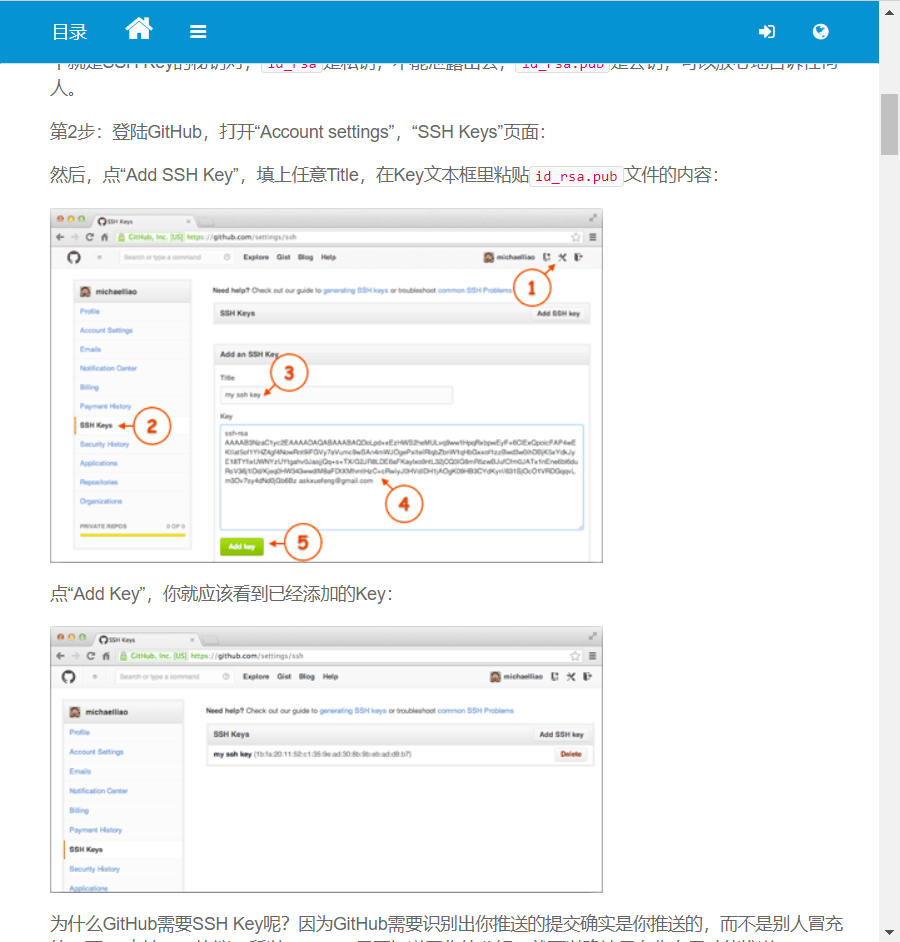
git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。

命令git rm用于删除一个文件。如果一个文件已经被提交到版本库，那么你永远不用担心误删，但是要小心，你只能恢复文件到最新版本，你会丢失**最近一次提交后你修改的内容**

#### 添加远程库

创建ssh key

$ ssh-keygen -t rsa -C "youremail@example.com"



首先，登陆GitHub，然后，在右上角找到“Create a new repo”按钮，创建一个新的仓库：



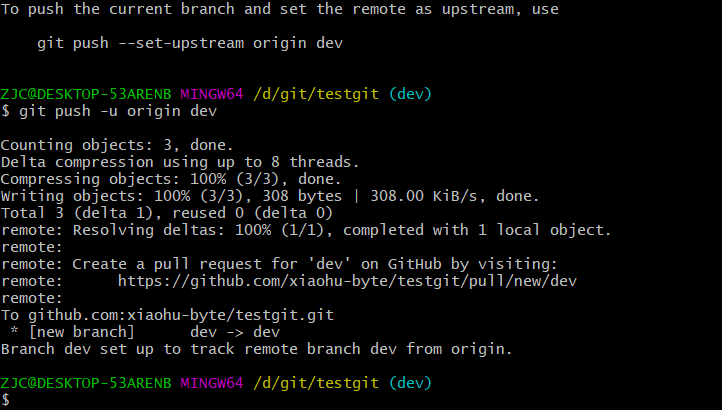
在Repository name填入learngit，其他保持默认设置，点击“Create repository”按钮，就成功地创建了一个新的Git仓库：



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

上面的michaelliao替换成你自己的GitHub账户名

推送到远程库



**删除远程库**

**首先查看信息**

$ git remote -v

origin git@github.com:michaelliao/learn-git.git (fetch)

origin git@github.com:michaelliao/learn-git.git (push)

然后，根据名字删除，比如删除origin：

$ git remote rm origin

**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；

关联一个远程库时必须给远程库指定一个名字，origin是默认习惯命名；

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；

分布式版本系统的最大好处之一是在本地工作完全不需要考虑远程库的存在，也就是有没有联网都可以正常工作，而SVN在没有联网的时候是拒绝干活的！当有网络的时候，再把本地提交推送一下就完成了同步，真是太方便了！

1克隆

$ git clone git@github.com:michaelliao/gitskills.git

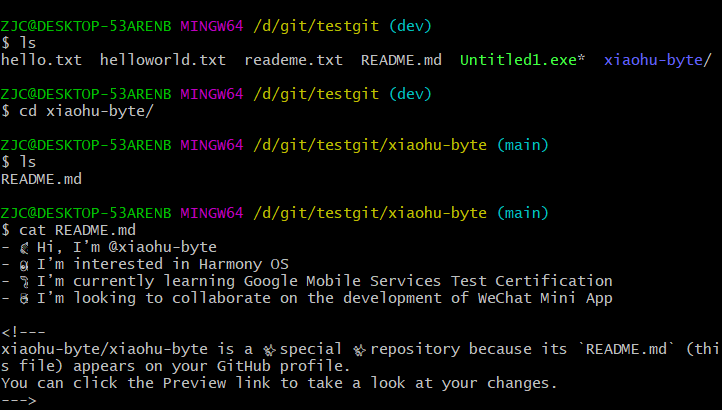
Cloning into 'gitskills'...

remote: Counting objects: 3, done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 3

Receiving objects: 100% (3/3), done.

**2 进入目录并查看**



### 小结

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ssh协议速度最快。

**创建和管理分支**

$ git checkout -b dev

Switched to a new branch 'dev'

**=**

$ git branch dev

$ git checkout dev

Switched to branch 'dev'

git branch命令查看当前分支

$ git branch

\* dev

Master

提交

$ git add readme.txt

$ git commit -m "branch test"

[dev b17d20e] branch test

1 file changed, 1 insertion(+)

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'

**合并**

$ git merge dev

Updating d46f35e..b17d20e

Fast-forward

readme.txt | 1 +

1 file changed, 1 insertion(+)

**删除**

$ git branch -d dev

Deleted branch dev (was b17d20e).

**查看**

$ git branch

\* master

**切换**

小结

Git鼓励大量使用分支：

查看分支：git branch

创建分支：git branch <name>

切换分支：git checkout <name>或者git switch <name>

创建+切换分支：git checkout -b <name>或者git switch -c <name>

合并某分支到当前分支：git merge <name>

删除分支：git branch -d <name>

**分支管理策略**

**1 储藏**

$ git stash

Saved working directory **and** index state WIP on dev: f52c633 add merge

首先确定要在哪个分支上修复bug，假定需要在master分支上修复，就从master创建临时分支：

1

$ git checkout master

Switched to branch 'master'

Your branch is ahead of 'origin/master' by 6 commits.

(use "git push" to publish your local commits)

$ git checkout -b issue-101

Switched to a new branch 'issue-101

### 小结

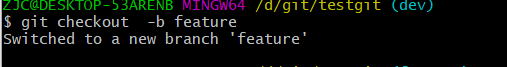
修复bug时，我们会通过创建新的bug分支进行修复，然后合并，最后删除；

当手头工作没有完成时，先把工作现场git stash一下，然后去修复bug，修复后，再git stash pop，回到工作现场；

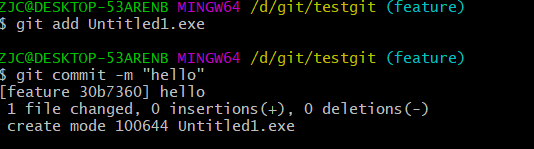
在master分支上修复的bug，想要合并到当前dev分支，可以用git cherry-pick <commit>命令，把bug提交的修改“复制”到当前分支，避免重复劳动。

开发代号为Vulcan的新功能，该功能计划用于下一代星际飞船。

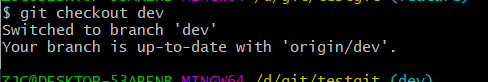
1 新建分支



**2 上传**



**3 切换回原分支**



**4 删除刚才分支**



要查看远程库的信息，用git remote：

$ git remote

Origin



推送分支

$ git push origin master

或

$ git push origin dev

现在，模拟一个你的小伙伴，可以在另一台电脑（注意要把SSH Key添加到GitHub）或者同一台电脑的另一个目录下克隆：

$ git clone git@github.com:michaelliao/learngit.git

Cloning into 'learngit'...

remote: Counting objects: 40, done.

remote: Compressing objects: 100% (21/21), done.

remote: Total 40 (delta 14), reused 40 (delta 14), pack-reused 0

Receiving objects: 100% (40/40), done.

Resolving deltas: 100% (14/14), done.

2 当你的小伙伴从远程库clone时，默认情况下，你的小伙伴只能看到本地的master分支。不信可以用git branch命令看看

$ git branch

\* master

创建远程分支

$ git checkout -b dev origin/dev

伙伴推送

$ git add env.txt

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

[dev 7a5e5dd] **add** env

1 file changed, 1 insertion(+)

**create** mode 100644 env.txt

$ git push origin dev

Counting objects: 3, done.

Delta compression **using** up **to** 4 threads.

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 308 bytes | 308.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0)

**To** github.com:michaelliao/learngit.git

f52c633..7a5e5dd dev -> dev

解决冲突

推送失败，因为你的小伙伴的最新提交和你试图推送的提交有冲突，解决办法也很简单，Git已经提示我们，先用git pull把最新的提交从origin/dev抓下来，然后，在本地合并，解决冲突，再推送

$ git pull

There is no tracking information **for** the current branch.

Please specify which branch you want to merge **with**.

See git-pull(1) **for** details.

git pull <remote> <branch>

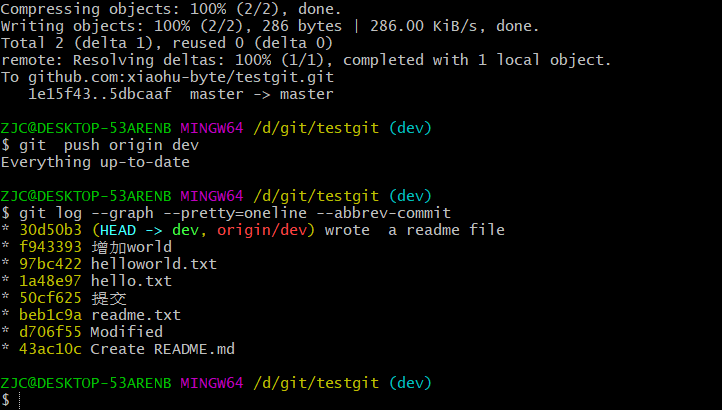
If you wish to set tracking information for this branch you can do so with:

git branch --set-upstream-to=origin/<branch> dev

### 小结

* 查看远程库信息，使用git remote -v；
* 本地新建的分支如果不推送到远程，对其他人就是不可见的；
* 从本地推送分支，使用git push origin branch-name，如果推送失败，先用git pull抓取远程的新提交；
* 在本地创建和远程分支对应的分支，使用git checkout -b branch-name origin/branch-name，本地和远程分支的名称最好一致；
* 建立本地分支和远程分支的关联，使用git branch --set-upstream branch-name origin/branch-name；
* 从远程抓取分支，使用git pull，如果有冲突，要先处理冲突

查看历史记录



然后，敲命令git tag <name>就可以打一个新标签：

$ git tag v1.0

可以用命令git tag查看所有标签：

$ git tag

v1.0

add merge这次提交打标签，它对应的commit id是f52c633，敲入命令：

$ git tag v0.9 f52c63

### 小结

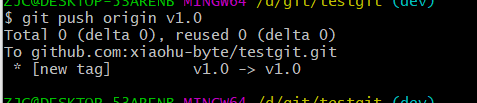
* 命令git tag <tagname>用于新建一个标签，默认为HEAD，也可以指定一个commit id；
* 命令git tag -a <tagname> -m "blablabla..."可以指定标签信息；
* 命令git tag可以查看所有标签。

1 如果标签打错了，也可以删除：

$ git tag -d v0.1

Deleted tag 'v0.1' (was f15b0dd)

如果要推送某个标签到远程，使用命令git push origin <tagname>：



一次性推送全部尚未推送到远程的本地标签：

$ git push origin --tags

Total 0 (delta 0), reused 0 (delta 0)

To github.com:michaelliao/learngit.git

\* [new tag] v0.9 -> v0.9

删除标签

1 本地删除

$ git tag -d v0.9

Deleted tag 'v0.9' (was f52c633)

2 远程删除

$ git push origin :refs/tags/v0.9

To github.com:michaelliao/learngit.git

- [deleted] v0.9

### 小结

* 命令git push origin <tagname>可以推送一个本地标签；
* 命令git push origin --tags可以推送全部未推送过的本地标签；
* 命令git tag -d <tagname>可以删除一个本地标签；
* 命令git push origin :refs/tags/<tagname>可以删除一个远程标签