#### 创建版本库

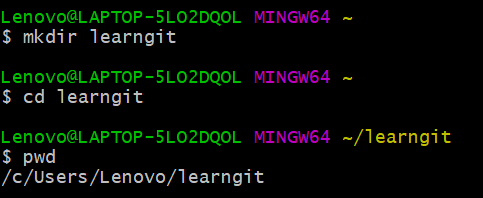
1、创建文件夹

输入：

$ mkdir learngit

$ cd learngit

$ pwd

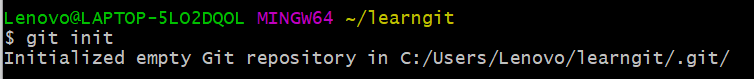
$pwd 显示当前learngit的路径在相应路径下可以找到learngit文件夹

1. 将目录创建为仓库

$git init 命令初始化仓库

$ git init

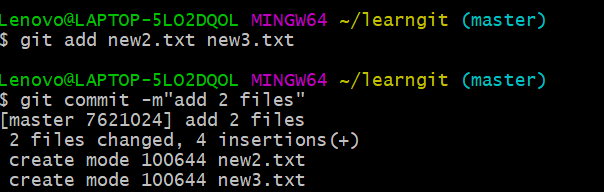
Initialized empty Git repository **in** /Users/michael/learngit/.git/



1. 添加文件

$git add file 添加文件可以添加多个

$git commit -m”” 完成



用Notepad++，写一个readme.txt的文件，吧它放到所创建的目录之下输入

$ git add readme.txt

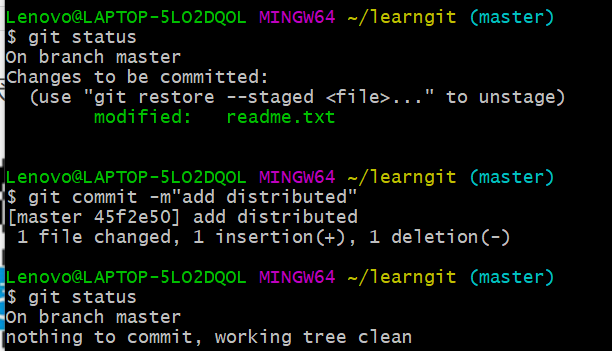
添加成功

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

把提交到仓库。

#### **修改**

1. git status 查看仓库当前状态

Git diff 查看修改内容

1. 版本退回：

HEAD 指向的版本就是当前的版本，使用命令$git reset --hard

Commit\_id 退回到相应版本

可以使用git log 查看提交历史，决定退回到那个版本

使用 git reflog 查看命令历史，确定要回到未来的那个版本

1. 工作区和暂存区

Git add 把要提交的所有修改放到缓存区，然后Git commit一次性把暂存区的所有修改提交到分支

1. 管理修改

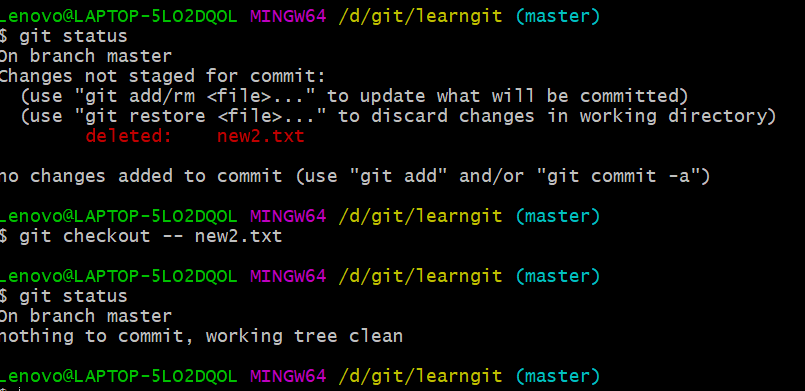
修改文件->git add->修改->git commit 第二次修改吗，没有被提交。因为第一次的修改后 git add把第一次修改放入暂存区，但第二次修改没有放入，提交时只提交暂存区的因此第二次的没有提交，没有记录

1. 撤销修改
2. 当改乱了工作区的某个文件内容时，想直接丢弃工作区的修改时 使用 git checkout -- file
3. 当将工作区的某个修改添加到暂存区时，想丢弃修改，使用git reset HEAD file

回到（1）

1. 提交了不合适的修改到版本库，可以使用版本回退
2. 删除文件

使用 git rm 删除一个文件,使用 $git checkout -- filename 可以使用版本库里的版本替换工作区的版本，把误删的文件恢复到最新版本。



#### 远程仓库

1. 添加远程仓库

关联一个远程库，使用git remote add origin [git@server-name:path/repo-name.git](mailto:git@server-name:path/repo-name.git)

使用git push -u origin master第一次推送master分支的所有内容

每次本地提交后就可以使用 git push origin master 推送最新修改

1. 从远程克隆

在GitHub上建立一个仓库，再使用

$ git clone git@github.com:ttnh899/gitskills.git

可以将GitHub上的仓库克隆到电脑中

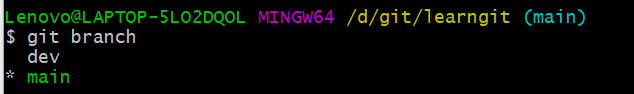
还可以使用https://github.com/michaelliao/gitskills.git

Git支持多种协议但ssh最快

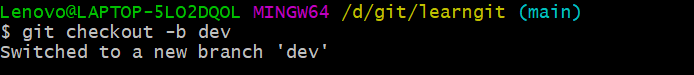
#### 分支管理

1、创建与合并

（1）查看分支：git branch



（2）创建分支git branch <name>



（3）切换分支git checkout <name>或git switch <name>

（4）创建+切换分支git checkout -b <name>或者git switch -c <name>

（5）合并某分支到当前分支git merge <name>

（6）删除分支git branch -d <name>

1. 解决冲突

当Git无法自动合并分支时，就必须首先解决冲突。解决冲突后，再提交，合并完成。

解决冲突就是把Git合并失败的文件手动编辑为我们希望的内容，再提交。

使用git log --graph 可以看到分支合并图

1. 分支管理策略

在实际开发中master 分支非常稳定，在dev分支上干活，最后再合并到master上

加上“--no--ff”就是普通模式合并

1. bug分支

修复bug时，我们可以通过创建新的bug分支进行修复，然后合并，最后删除

使用$ git stash把当前工作现场存储起来$ git checkout -b issue-101 建立临时分支，使用git stash list查看原本的工作位置$ git stash list

stash@{0}: WIP on dev: f52c633 add merge

使用git stash apply或git stash drop删除临时分支git stash pop删除分支和内容

cherry-pick 复制一个特定的提交到当前分支$ git cherry-pick 4c805e2

6、feature 分支

如果要丢弃一个没有被合并过的分支，通过git branch -D <name>强行删除

7、多人协作

查看远程库的信息git remote git remote -v显示更详细的信息

推送分支：

如：$ git push origin dev 将dev分支推送到远程库对应的远程分支上

抓取分支：

另一个人使用$ git checkout -b dev origin/dev创建远程origin的dev分支到本地，现在，他就可以在dev分支上继续修改，然后不断把dev分支使用$ git push origin dev 把dev分支发送到远程，而两人正在对同样的文件修改，他人的最新提交和你试图推送的提交有冲突推送失败。解决：

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，再推送没有指定本地dev 分支与远程origin/dev 分支连接，设置dev和origin/dev的连接

$ git branch --set-upstream-to=origin/dev dev

Branch 'dev' set up to track remote branch 'dev' from 'origin'.

再pull

8、Rebase

使用$git rebase ba 把本地未push的分叉提交历史整理成直线

#### 标签管理

1. 创建标签
2. 切换到需要打标签的分支上
3. 使用git tag <name>打一个新标签

使用 git tag 查看所有标签，可以使用git tag 标签名 commitID 对以前的提交打标签

创建带说明的标签：

$ git tag -a v0.1 -m "version 0.1 released" 1094adb

-a 可以指定标签名，-m 指定说明文字

使用 git show tagname 查看说明文字

1. 操作标签

删除标签：$ git tag -d v0.1

将标签推送到远程：git push origin <tagname>

一次性将所有标签推送到远程$ git push origin --tags

删除远程标签：先从本地删除$ git tag -d v0.9

再从远程删除$ git push origin :refs/tags/v0.9

1. 自定义git

1、.gitignore

在git工作区的根目录下创建一个特殊的.gitognore文件，然后把要忽略的文件名填进去，git就会自动忽略这些文件

2、配置别名

1. $ git config --global alias.st status 使用st代替status