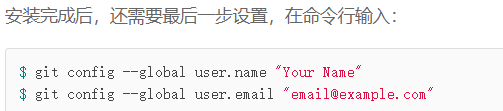
Git教程

1. Git简介

Git是目前世界上最先进的分布式版本控制系统。

在windows上***安装***git

直接下载安装程序，一步步安装。在开始菜单里找到“Git”->“Git Bash”，蹦出一个类似命令行窗口的东西，就说明Git安装成功！



自报家门，这是设置本电脑为git机器的凭证

***创建本地版本库***（repository）：

1. 找一个空的文件夹设置为repository
2. 通过git init命令把这个目录变成Git可以管理的仓库

这就创建了一个空的版本库，会出现一个.git的目录，用于追踪管理版本库

***把文件添加到版本库***：

首先这里再明确一下，所有的版本控制系统，其实只能跟踪文本文件的改动，比如TXT文件，网页，所有的程序代码等等。假设repository里面首先编写了一个readme.txt文件将其添加到版本库。

1. 第一步，用命令git add*+文件名* （或者git add *.* 表示添加所有刚才修改的文件）告诉Git，把文件添加到仓库
2. 第二步，用命令git commit -m “说明”告诉Git，把文件提交到仓库，简单解释一下git commit命令，-m后面输入的是本次提交的说明

总结：



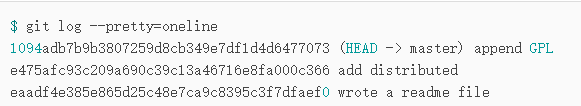
1. 时光机穿梭

如果修改了文件内容，可以通过git status 查看修改的文件。如果需要看文件修改了哪些东西可以通过git diff+文件名 来查看修改情况。然后将修改后的文件提交到仓库（git add . 和 git commit -m “修改的注释”）



***版本回退：***

git log命令显示从最近到最远的提交日志，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：



看到的一大串类似1094adb...的是commit id（版本号）

Git必须知道当前版本是哪个版本，在Git中，用HEAD表示当前版本，也就是最新的提交1094adb...（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

***如何回到之前版本：***

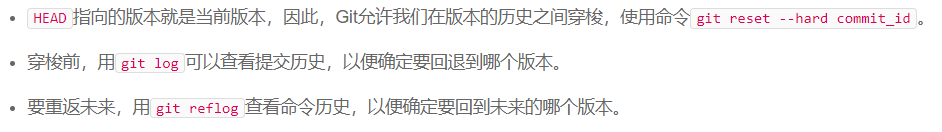
git reset --hard HEAD^ 则表示回退到上一个版本

git reset --hard HEAD^^ 则表**示**回退到上上一个版本，以此类推。

***如何回到未来版本：***

git reset --hard + commit id Git提供了一个命令git reflog用来记录你的每一次命令从而找到版本号。

小结：



***工作区和暂存区：***

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。



前面说的将文件提交到版本库分两步执行：

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；

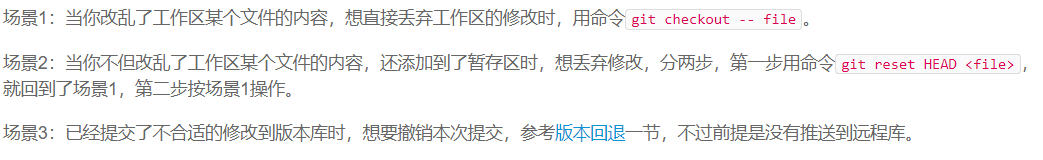
第二步是用git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支。

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。

***二次修改最后提交***，也就是可以先多次add后，最后commit

第一次修改 -> git add -> 第二次修改 -> git add -> git commit

***撤销修改***：



***删除文件：***

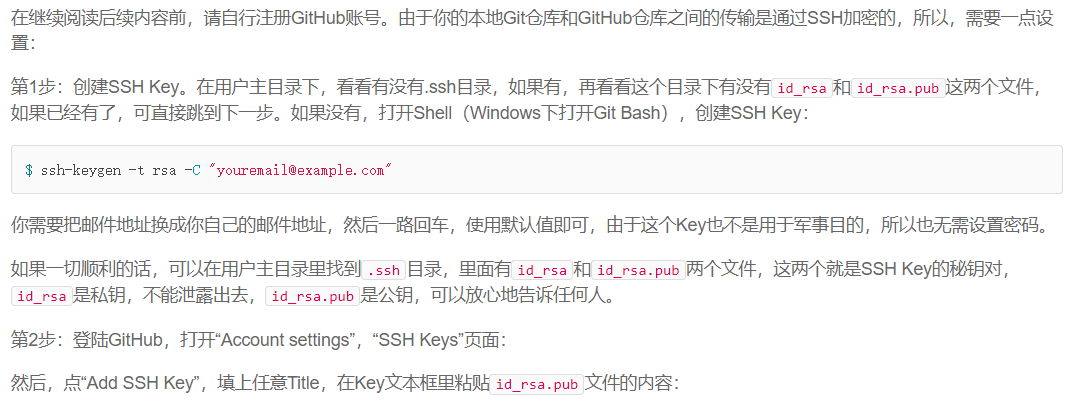
如果你在工作区删除了某个文件，而版本库和工作区就会不一致，通过git status命令会立刻告诉你哪些文件被删除了。

现在你有两个选择，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：现在，文件就从版本库中被删除了

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+ file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”。

1. 远程仓库

GitHub需要识别出你推送的提交确实是你推送的，所以要设置SSH Key



***创建远程GitHub库：***

1. 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
2. 在Repository name填入本地库的文件夹库名，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：

***要关联一个远程库***，使用命令git remote add origin git@server-name:path/repo-name.git；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，比如关联GitHub可以起名github，关联Gitee可以起名gitee等。

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；

***克隆远程库：***

上次我们讲了先有本地库，后有远程库的时候，如何关联远程库。

那么最好的方式是先创建远程库，然后，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ssh协议速度最快。

1. 分支管理

***创建与合并分支***

每次提交，Git都把它们串成一条时间线，这条时间线就是一个分支。

截止到目前，只有一条时间线，在Git里，这个分支叫主分支，即master分支。

HEAD严格来说不是指向提交，而是指向master，master才是指向提交的，所以，HEAD指向的就是当前分支。



当我们创建新的分支，例如dev时，Git新建了一个指针叫dev，指向master相同的提交，再把HEAD指向dev，就表示当前分支在dev上：



从现在开始，对工作区的修改和提交就是针对dev分支了，比如新提交一次后，dev指针往前移动一步，而master指针不变



Git怎么合并呢？最简单的方法，就是直接把master指向dev的当前提交，就完成了合并：



合并完分支后，甚至可以删除dev分支。删除dev分支就是把dev指针给删掉，删掉后，我们就剩下了一条master分支：



实战：

首先，我们创建dev分支，然后切换到dev分支：git checkout -b 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和git checkout dev

git branch命令会列出所有分支，当前分支前面会标一个\*号。

dev分支的工作完成，我们就可以切换回master分支：git checkout master

我们把dev分支的工作成果合并到master分支上：git merge dev。//git merge命令用于合并指定分支到当前分支。

合并完成后，就可以放心地删除dev分支了：git branch -d dev

注意到切换分支使用git checkout <branch>，而前面讲过的撤销修改则是git checkout -- <file>最新版本的Git提供了新的git switch命令来切换分支



***解决冲突：***

准备新的feature1分支（git switch -c feature1），继续我们的新分支开发，修改readme.txt最后一行，改为..然后在feature1分支上提交：

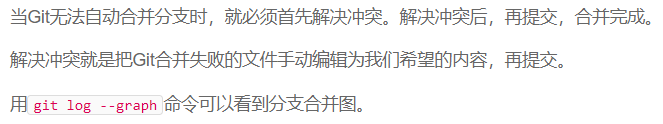
切换到master分支：在master分支上把readme.txt文件的最后一行改为..然后提交

现在，master分支和feature1分支各自都分别有新的提交，变成了这样：



果然冲突了！Git告诉我们，readme.txt文件存在冲突，必须手动解决冲突后再提交。git status也可以告诉我们冲突的文件：

小结：



***分支管理策略：***

合并分支时，如果可能，Git会用Fast forward模式，但这种模式下，删除分支后，会丢掉分支信息

强制禁用Fast forward模式，Git就会在merge时生成一个新的commit，这样，从分支历史上就可以看出分支信息。

实战一下--no-ff方式的git merge：

1. 首先，仍然创建并切换dev分支git switch -c dev
2. 修改readme.txt文件，并提交一个新的commit：
3. 现在，我们切换回master：准备合并dev分支，请注意--no-ff参数，表示禁用Fast forward：git merge --no-ff -m "merge with no-ff" dev因为本次合并要创建一个新的commit，所以加上-m参数，把commit描述写进去。合并后，我们用git log看看分支历史



在实际开发中，master分支应该是非常稳定的，也就是仅用来发布新版本，平时不能在上面干活

干活都在dev分支上，也就是说，dev分支是不稳定的，到某个时候，比如1.0版本发布时，再把dev分支合并到master上，在master分支发布1.0版本；

你和你的小伙伴们每个人都在dev分支上干活，每个人都有自己的分支，时不时地往dev分支上合并就可以了。



***Bug分支***

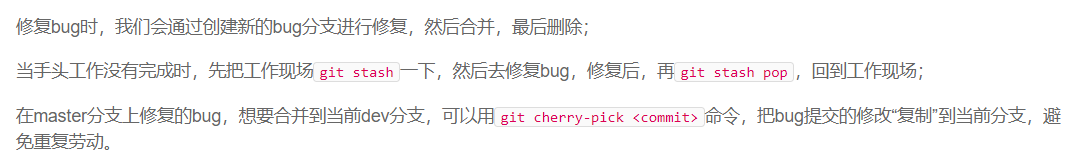
工作只进行到一半，还没法提交，预计完成还需1天时间。但是，必须在两个小时内修复该bug，怎么办？

Git还提供了一个stash功能git stash，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

现在，用git status查看工作区，就是干净的，因此可以放心地创建分支来修复bug。

1. 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：git checkout master，git checkout -b issue-101
2. 现在修复bug，git add readme.txt，git commit -m "fix bug 101"
3. 修复完成后，切换到master分支，并完成合并，最后删除issue-101分支：git switch master，git merge --no-ff -m "merged bug fix 101" issue-101
4. 现在，是时候接着回到dev分支干活了，git switch dev，用git stash list命令看看，工作现场还在，Git把stash内容存在某个地方了，但是需要恢复一下，有两个办法：一是用git stash apply或git stash apply+指定工作区恢复，但是恢复后，stash内容并不删除，你需要用git stash drop来删除；另一种方式是用git stash pop，恢复的同时把stash内容也删了
5. 对于其他分支也有同样bug，可以我们只需要把4c805e2 fix bug 101这个提交所做的修改“复制”到dev分支，Git专门提供了一个cherry-pick命令，让我们能复制一个特定的提交到当前分支：git cherry-pick 4c805e2

小结：



***Feature分支***

添加一个新功能时，你肯定不希望因为一些实验性质的代码，把主分支搞乱了，所以，每添加一个新功能，最好新建一个feature分支，在上面开发，完成后，合并，最后，删除该feature分支。

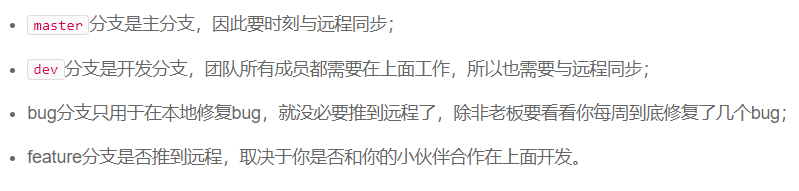
1. 接到了一个新任务：开发代号为Vulcan的新功能，git switch -c feature-vulcan，然后开发，add和commit后切回dev分支准备合并。
2. 一切顺利的话，feature分支和bug分支是类似的，合并，然后删除。但是这个包含机密资料的分支还是必须就地销毁git branch -D feature-vulcan如果要强行删除，需要使用大写的-D参数。

***多人合作或一人多机器***

当你从远程仓库克隆时，实际上Git自动把本地的master分支和远程的master分支对应起来了，并且，远程仓库的默认名称是origin。

要查看远程库的信息，用git remote -v

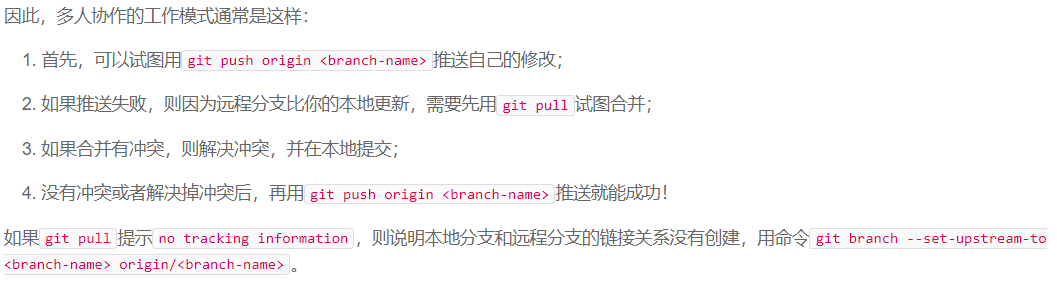
推送分支，就是把该分支上的所有本地提交推送到远程库。推送时，要指定本地分支，这样，Git就会把该分支推送到远程库对应的远程分支上：



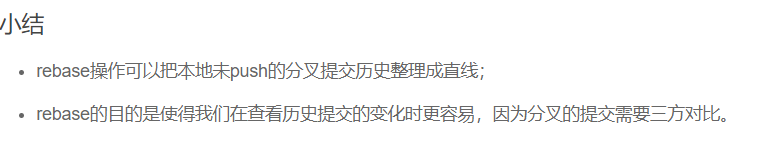
当你的小伙伴从远程库clone时，默认情况下，你的小伙伴只能看到本地的master分支。

你的小伙伴要在dev分支上开发，就必须创建远程origin的dev分支到本地，于是他用这个命令创建本地dev分支：git checkout -b dev origin/dev

在，他就可以在dev上继续修改，然后，时不时地把dev分支push到远程：







1. 标签管理

标签总是和某个commit挂钩。如果这个commit既出现在master分支，又出现在dev分支，那么在这两个分支上都可以看到这个标签。

### 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。
* 用命令git show <tagname>可以看到说明文字：

### 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。

***关联多个远程库：***

仍然以learngit本地库为例，我们先删除已关联的名为origin的远程库：git remote rm origin

然后，先关联GitHub的远程库：git remote add github git@github.com:michaelliao/learngit.git

接着，再关联Gitee的远程库：git remote add gitee git@gitee.com:liaoxuefeng/learngit.git

现在，我们用git remote -v查看远程库信息，可以看到两个远程库：

git remote -v

gitee git@gitee.com:liaoxuefeng/learngit.git (fetch)

gitee git@gitee.com:liaoxuefeng/learngit.git (push)

github git@github.com:michaelliao/learngit.git (fetch)

github git@github.com:michaelliao/learngit.git (push)

如果要推送到GitHub，使用命令：git push github master

如果要推送到Gitee，使用命令：git push gitee master

### Git中从远程的分支获取最新的版本到本地有这样2个命令：



