git的基本操作

1、安装git

2、创建用户名和邮箱

git config --gloabal user.name "lau0400810121"

git config --gloabal user.email "lau0400810121@163.com"

3、建立版本仓库

git init

4、在仓库的工作区创建文件、添加到缓存区、提交到版本仓库

vim 1.php 创建

git add 1.php 添加到缓存区

git status 查看当前工作区的状态

git diff HEAD -- 1.php 查看当前修改和版本库中文件的差别

git commit -m '版本修改说明'

5、版本回退

git log 查看所有的版本信息

git reset --hard HEAD^ 回到上一版本

git reset --hard HEAD^^ 回到上上一版本

git reset --hard 版本ID

git reflog 查看版本操作命令（方便进行版本回到未来的操作）

6、撤销工作区内容的修改（在commit之前）

(已经add但没有提交)

git reset HEAD 1.php

(未提交之前的修改)

git checkout -- 1.php

会回退到最初的工作区的状态

但一旦提交到了版本仓库了之后，那就使用 git rest --hard 版本ID 来实现版本的回退，尽量不要托管到远程仓库再进行修改

总结：

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，参考版本回退一节，不过前提是没有推送到远程库。

7、删除文件

1）先在本地文件管理目录进行删除 git rm 1.php 然后提交到版本仓库 git commit -m 'delete 1.php' 实现本地和版本仓库的一致性

2）撤回删除 git checkout -- 1.php

8、远程仓库

第1步：生成秘钥文件

ssh-keygen -t rsa -C "lau0400810121@163.com"

一路下一步，最终在”C:\Users\hasee\.ssh“下看到2个文件

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：

第3步：团队开发，添加多个ssh key

9、添加远程仓库

在github上创建一个新的仓库，假设仓库名称为study

1、本地库与远程库建立联系

git remote add origin git@github.com:自己的github账号名/study.git

2、把本地仓库下的内容推送到GitHub仓库中

git push -u origin master 把本地master分支推送到远程仓库的master分支中

git push -u origin master:developer 把本地master分支推送到远程仓库的developer分支中,再在远程master中进行合并

3、以后只要本地做了提交到本地版本库，就可以使用下面命令推送到

git push origin master 推送到GitHub

假如远程仓库已经有了文件了，先得同步到本地

git pull --rebase origin master

或者

git rebase orgin master

从github拉取下来最新的和本地进行合并

git pull origin dev 从远程的dev分支上拉取到本地的master分支上

如果远程仓库走的是https协议，必须输入仓库的用户名和密码

如果是ssh协议，则不需要，前提是初始化时的用户名和邮箱必须和远程仓库的保持一致

10、从远程库中克隆

在github上创建一个新的版本仓库，命名为mygit

克隆远程仓库到本地形成一个本地仓库

git clone git@github.com:GitHub账号/mygit.git（既可以是git@github.com，也可以是https://github.com/GitHub账号/mygit.git）

然后在本地的mygit库中查看同步的文件

11、分支

创建并切换到新分支 git checkout -b dev

基于其他分支到新分支 git checkout -b dev 分支名

相当于： git branch dev(创建) git checkout dev(切换)

git branch 查看当前分支

分支合并:

首先切换到主分支上 git checkout master

在主分支上进行合并 git merge dev

分支删除：

git branch -d dev

12、冲突解决

1）当在一个分支上修改一个文件提交到分支版本库时，同时主分支也进行修改该文件并进行提交，

接着进行合并时就会出现冲突

2）直接在主分支上打开冲突文件（可以使用git status查看），并进行相应的修改，最后提交即可

3）删除分支

4）git log --graph --pretty=oneline --abbrev-commit 可以看到分支合并图

13、分支处理策略

master分支只作为发布版本，平时不在上面操作

dev分支用来做开发使用，每个人往dev分支上开发，最后进行合并

git merge --no-ff -m "merge with no-ff" dev

14、BUG分支

假定手头有分工作正在dev分支上操作，但是没有完成，但此时需要修复BUG

可以先隐藏当前的分支

git stash

通过git status 查看工作区

假设需要在主分支上修复BUG，此时切换到主分支上去

git checkout master

在主分支上创建BUG分支

git checkout -b bug_01

在BUG分支完成修改提交后，切换到主分支进行合并并删除掉BUG分支

接着重新跳到dev分支上工作

git checkout dev

重新返回到原工作区

git stash pop

15、强行删除（一个未合并的）分支

git branch -D feature-vulca

16、多人操作

查看远程仓库 git remote -v

推送分支 git push origin master

git push origin dev

master分支是主分支，因此要时刻与远程同步；

dev分支是开发分支，团队所有成员都需要在上面工作，所以也需要与远程同步；

bug分支只用于在本地修复bug，就没必要推到远程了，除非老板要看看你每周到底修复了几个bug；

feature分支是否推到远程，取决于你是否和你的小伙伴合作在上面开发。

17 多人协作开发

多人协作的工作模式通常是这样：

首先，可以试图用git push origin branch-name推送自己的修改；

如果推送失败，则因为远程分支比你的本地更新，需要先用git pull试图合并；

如果合并有冲突，则解决冲突，并在本地提交；

没有冲突或者解决掉冲突后，再用git push origin branch-name推送就能成功！

如果git pull提示“no tracking information”，则说明本地分支和远程分支的链接关系没有创建，用命令git branch --set-upstream branch-name origin/branch-name

18、打标签方面更好的查找，比版本ID更容易记

在指定分支上添加标签 git tag v1.0

查找所有的标签 git tag

查看所有的版本信息 git log --pretty=oneline --abbrev-commit

创建带有说明的标签 git tag -a v0.1 -m "version 0.1 released" 3628164

查看标签信息 git show v0.1

删除标签 git tag -d v0.1

推送标签到远程 git push origin v0.1

git push origin --tags

删除已经推送到远程的标签

先在本地删除 git tag -d v0.1

再删除远程 git push origin :refs/tags/v0.1

19、搭建git服务器

第一步，安装git

yum -y install git

第二步，创建git用户来管理git

adduser git

第三步，创建证书登陆

收集所有需要登录的用户的公钥，就是他们自己的id_rsa.pub文件，把所有公钥导入到/home/git/.ssh/authorized_keys文件里，一行一个。

第四步，初始化Git仓库：

先选定一个目录作为Git仓库，假定是/srv/sample.git，在/srv目录下输入命令

git init --bare sample.git

Git就会创建一个裸仓库，裸仓库没有工作区，因为服务器上的Git仓库纯粹是为了共享，所以不让用户直接登录到服务器上去改工作区，并且服务器上的Git仓库通常都以.git结尾。然后，把owner改为git：

chown -R git:git sample.git

第五步，禁用shell登录：

出于安全考虑，第二步创建的git用户不允许登录shell，这可以通过编辑/etc/passwd文件完成。找到类似下面的一行：

git:x:1001:1001:,,,:/home/git:/bin/bash

改为：

git:x:1001:1001:,,,:/home/git:/usr/bin/git-shell

第六步，克隆远程仓库：

git clone git@server:/srv/sample.git

指定一个分支克隆

git clone -b git@server:/srv/sample.git