重要概念：工作区，暂存区，远程仓库（默认名称origin），本地仓库

1、git的诞生

如果不是当年BitMover公司威胁Linux社区，可能现在我们就没有免费而超级好用的Git了。

2、集中式vs分布式

集中式典型：svn；

分布式典型：git

3、git安装

git安装完成后，命令行中键入：

git config --global user.name "your name"

git config --global user.email "your email"

注意git config命令的**--global参数**，用了这个参数，表示**你这台机器上所有的Git仓库**都会使用这个配置，当然也可以对某个仓库指定不同的用户名和Email地址。

4、创建版本库

创建一个空目录：mkdir yourrepositoryname

pwd指令：显示当前目录

把某个目录变为git可管理的仓库：**git init**

文件**添加**到版本库：**git add** 文件名(如readme.txt)

文件**提交**至仓库：**git commit -m** blablabla（blabla记录的是你提交理由，如新增登录接口等）

查看**仓库当前**的状态：**git status**

查看**文件**做了哪些修改：**git diff** 文件名

5、版本回退（本地仓库中的版本回退）

显示**提交日志**：**git log (--pretty=oneline)** 可以获取commit id

版本回退至上一个版本：git reset --hard HEAD^(HEAD指向的是当前版本)

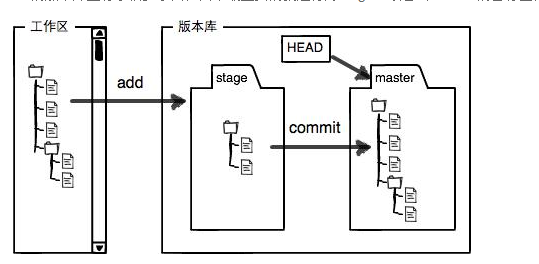
版本回退至上上个版本：git reset --hard HEAD^^

版本回退至上100个版本：git reset --hard HEAD~100

版本回退至某个版本：**git reset --hard 版本号**

查看历史git指令：**git reflog**

6、工作区和暂存区（又叫stage或index）



7、git管理的是修改而非文件

查看工作区和版本库里面最新版本的区别：**git diff HEAD --文件名**

8、撤销修改

查看文件中的内容：cat 文件名

**丢弃工作区中所有的修改**：**git checkout --文件名**

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：

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；

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。

总之，就是让这个文件回到最近一次git commit或git add时的状态

**撤销已经添加至暂存区的文件的修改**：①重新放回工作区：**git reset HEAD 文件名，**

②撤销工作区的修改：**git checkout --文件名**

**已经提交至版本库，但未提交至远程仓库**：使用**回退**操作撤销修改

9、删除文件

从版本库中删除文件：删掉本地文件，执行**git rm 文件名，然后git commit，这样仓库里的文件也被删掉了**

若文件之前已被提交至版本库，且本地文件被误删，恢复：执行**git checkout --文件名即可恢复文件至仓库中的最新版本**（git checkout其实就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）

10、建立远程仓库（git和svn的优势就在于分布式）

github上注册账户并建立仓库

由于你的本地Git仓库和GitHub仓库之间的传输是通过SSH加密的

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）$ ssh-keygen -t rsa -C "youremail@example.com"），id\_rsa和id\_rsa.pub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

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（GitHub需要SSH Key呢？因为GitHub需要识别出你推送的提交确实是你推送的，而不是别人冒充的，而Git支持SSH协议，所以，GitHub只要知道了你的公钥，就可以确认只有你自己才能推送，当然，GitHub允许你添加多个Key。假定你有若干电脑，你一会儿在公司提交，一会儿在家里提交，只要把每台电脑的Key都添加到GitHub，就可以在每台电脑上往GitHub推送了）

在本地仓库文件夹下，使本地仓库关联远程库的命令：**git remote add origin git@github.com:自己的github账户名/自己建立的github仓库名.git**

关联远程仓库后，第一次推送master分支所有内容：**git push -u origin master**(-u会将本地的master分支和远程的master分支关联起来)

以后推送最新修改：**git push origin master**

11、从远程仓库克隆

从远程仓库克隆至本地仓库：**git clone git@github.com: 自己的github用户名/github远程仓库名.git**

12、分支和合并

查看分支：**git branch**

创建分支：**git checkout -b dev**（-b参数表示**创建并切换**分支，相当于以下两条指令：

**git branch 分支名 和 git checkout 分支名**）

然后修改文件执行git add 文件名，git commit -u 理由

分支工作完成，切换至某个分支：**git checkout master**

分支的成果合并到当前分支，分支合并：**git merge dev（快速合并方式Fast forward）**

分支合并完成后，删除分支：**git branch -d dev**

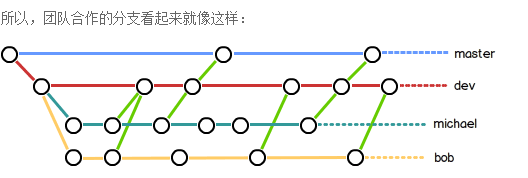
13、解决冲突

手动解决，然后提交

14、分支管理策略

合并分支时，git使用Fast forward模式，但这种模式下，删除分支时会丢掉分支信息，**若禁用fast forward 模式，git在merge时会生成一个新的commit**，这样从分支历史上就可以看出分支信息；

禁用fast forward模式的合并：**git merge --no--ff -m 提交理由 分支名**（本次合并需要创建一个新的commit，所以需要加上-m参数）



15、bug分支

当接到一个代号101的bug，若创建一个bug-101的分支来修复它，但是在分支dev上的工作还没有提交

git提供stash功能将当前工作现场"存储"起来：**git stash**

bug修复过程中，创建分支，提交，合并分支，删除分支，修复完之后，调出刚刚存储的工作现场：

①**git stash apply**，恢复后stash内容并不删除，需要用git stash drop来删除

②**git stash pop**，恢复同事stash内容也被删除了

再用**git stash list**查看就看不到任何stash内容了

先用stash list查看stash 列表，然后恢复指定的stash：**git stash apply stash@{0}**

16、feature分支

丢弃一个已经执行add/commit但没有被合并过的分支：**git branch -D 分支名**

即强制删除

查看远程仓库信息：git remote或者**git remote -v**显示更详细的信息

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：**git push origin 分支名**

git bash界面，使用git log之后无法终止这个命令，按下q即可

linux系统下vi编辑器：

按i进行inser模式，可以编辑文件；

按esc退出insert模式，按：wq！保存修改，按：q！放弃修改，然后退出vi编辑器