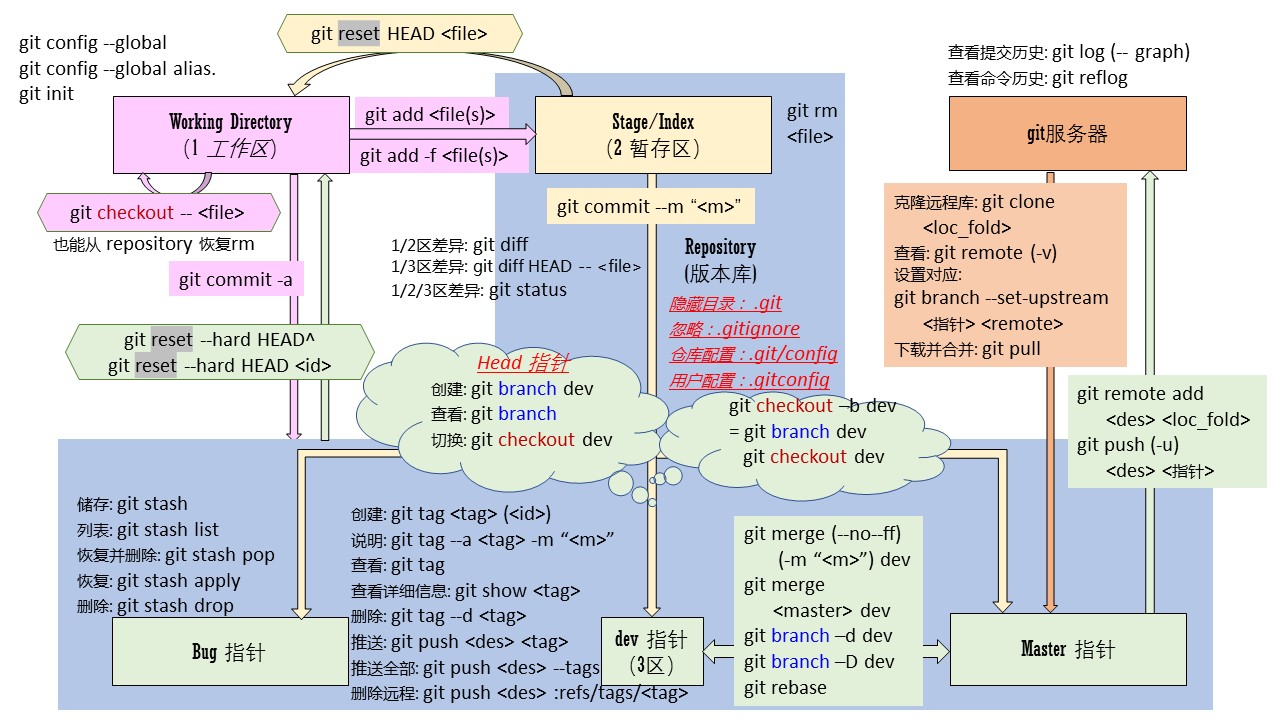
**Git的一些介绍和使用**

git是一个分布式版本控制软件



**一、创建版本库，创建一个仓库，并且如何添加文件到仓库中：**

1、初始化一个Git仓库，使用git init命令。

2、添加文件到Git仓库，分两步：

a.使用命令git add <file>，注意，可反复多次使用，添加多个文件；

b.使用命令git commit -m <message>，完成。

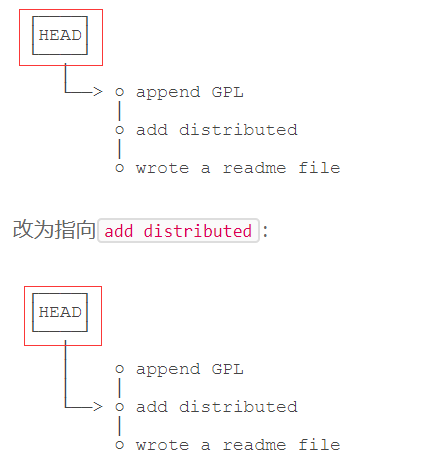
**二、时光机穿梭，Git的一些常用的命令：**

1、要随时掌握工作区的状态，使用git status命令。

2、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。

**三、版本回退：**

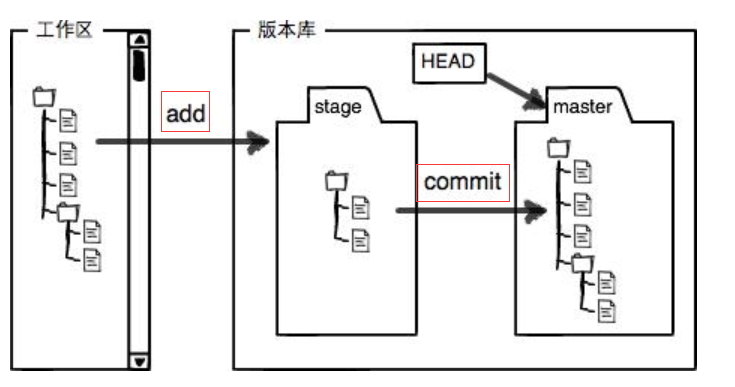
1、HEAD指向的版本就是当前版本，因此，Git允许我们在版本的历史之间穿梭，使用命令git reset --hard commit\_id。



2、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。

3、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。

**四、工作区和暂存区**



**五、管理修改：Git管理的是修改，而不是文件**

现在，你又理解了Git是如何跟踪修改的，每次修改，如果不用git add到暂存区，那就不会加入到commit中。

**六、撤销修改**

1、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。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”。

2、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。

3、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。

**七、删除文件**

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。

**八、添加远程库**

1、要关联一个远程库，使用命令git remote add origin git@server-name:path/repo-name.git；

例子：git remote add origin [git@github.com:JennyHJN/learngit](mailto:git@github.com:JennyHJN/learngit)

2、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；

3、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；

会出现的问题：git push错误failed to push some refs to的解决【解决博客地址：<https://blog.csdn.net/MBuger/article/details/70197532】>这个问题是因为远程库与本地库不一致造成的，那么我们把远程库同步到本地库就可以了。

使用指令：git pull --rebase origin master

这条指令的意思是把远程库中的更新合并到本地库中，–rebase的作用是取消掉本地库中刚刚的commit，并把他们接到更新后的版本库之中。

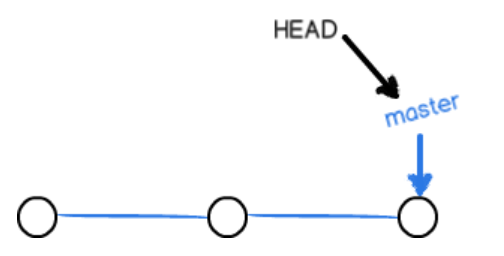
**九、从远程库克隆**

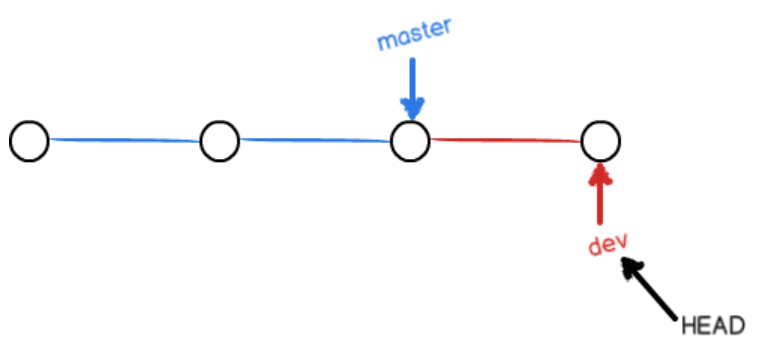
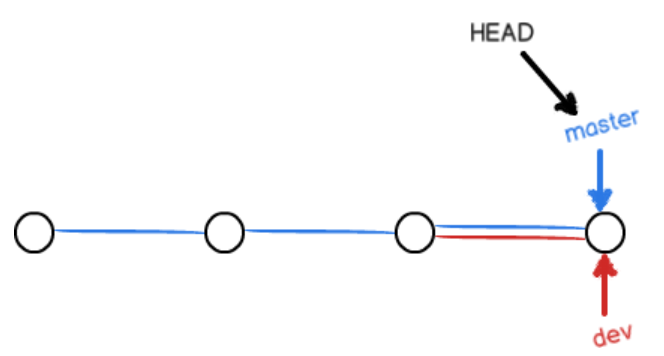
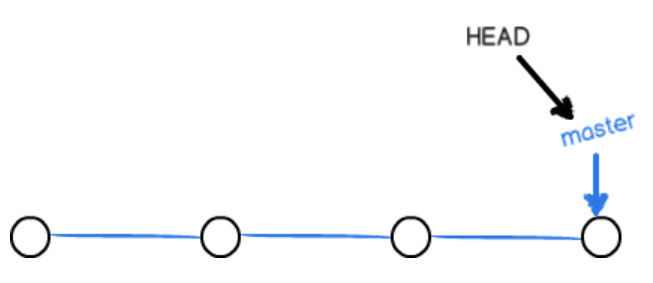
要克隆一个仓库，首先必须知道仓库的地址，然后使用git clone命令克隆。

命令：git clone git@github.com:JennyHJN/gitskills.git

Git支持多种协议，包括https，但通过ssh支持的原生git协议速度最快。

**十、创建与合并分支**

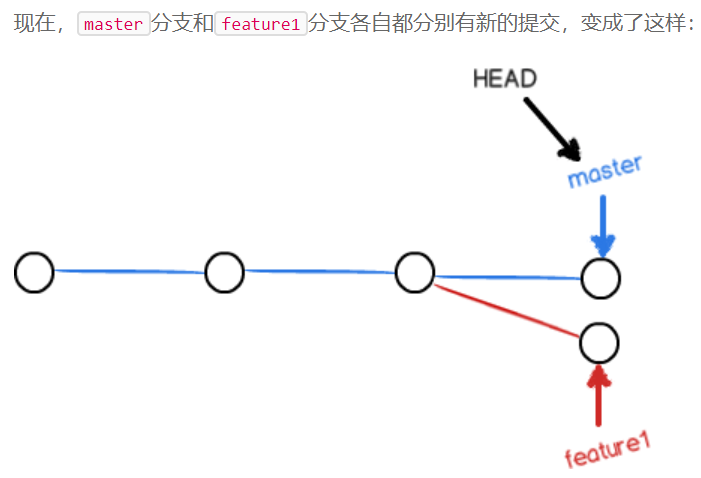


Git鼓励大量使用分支：

1. 查看分支：git branch
2. 创建分支：git branch <name>
3. 切换分支：git checkout <name>
4. 创建+切换分支：git checkout -b <name>
5. 合并某分支到当前分支：git merge <name>
6. 删除分支：git branch -d <name>

**十一、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。

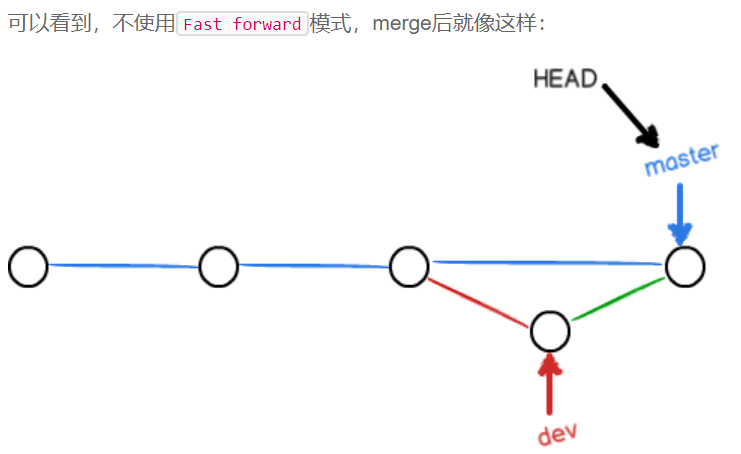
解决冲突就是把Git合并失败的文件手动编辑为我们希望的内容，再提交。

用git log --graph命令可以看到分支合并图。

命令：git log --graph --pretty=oneline --abbrev-commit

**十二、分支管理策略**

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



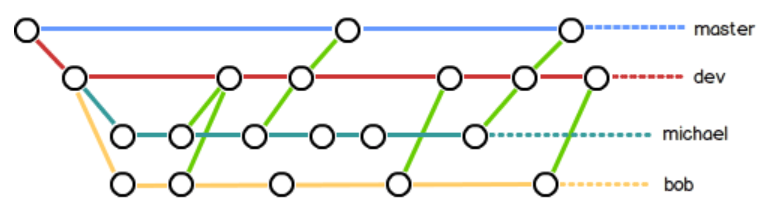
在实际开发中，我们应该按照几个基本原则进行分支管理：

首先，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；

那在哪干活呢？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。

所以，团队合作的分支看起来就像这样：



Git分支十分强大，在团队开发中应该充分应用。

合并分支时，加上--no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并。

**十三、Bug分支**

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。

**十四、Feature分支**

开发一个新feature，最好新建一个分支；

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

**十五、多人协作**

因此，多人协作的工作模式通常是这样：

1. 首先，可以试图用git push origin <branch-name>推送自己的修改；
2. 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；
3. 如果合并有冲突，则解决冲突，并在本地提交；
4. 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！
5. 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-to <branch-name> origin/<branch-name>。

小结：

1. 查看远程库信息，使用git remote -v；
2. 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；
3. 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；
4. 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；
5. 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；
6. 从远程抓取分支，使用git pull，如果有冲突，要先处理冲突。

**十六、Rebase**

这就是rebase操作的特点：把分叉的提交历史“整理”成一条直线，看上去更直观。缺点是本地的分叉提交已经被修改过了。

rebase操作可以把本地未push的分叉提交历史整理成直线；

rebase的目的是使得我们在查看历史提交的变化时更容易，因为分叉的提交需要三方对比。

**十七、创建标签**

注意：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。

1. 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：

1. 命令git tag -a <tagname> -m "blablabla..."可以指定标签信息；
2. 命令git tag可以查看所有标签。

**十八、操作标签**

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