Jun 27, 2021

Git知识点

> *The source comes from:*
> *https://www.liaoxuefeng.com/wiki/896043488029600*

Category

1.创建版本库

- 初始化一个Git仓库，使用git init命令。
- 添加文件到Git仓库，分两步：

1. 使用命令git add ，注意，可反复多次使用，添加多个文件；
2. 使用命令git commit -m ，完成。

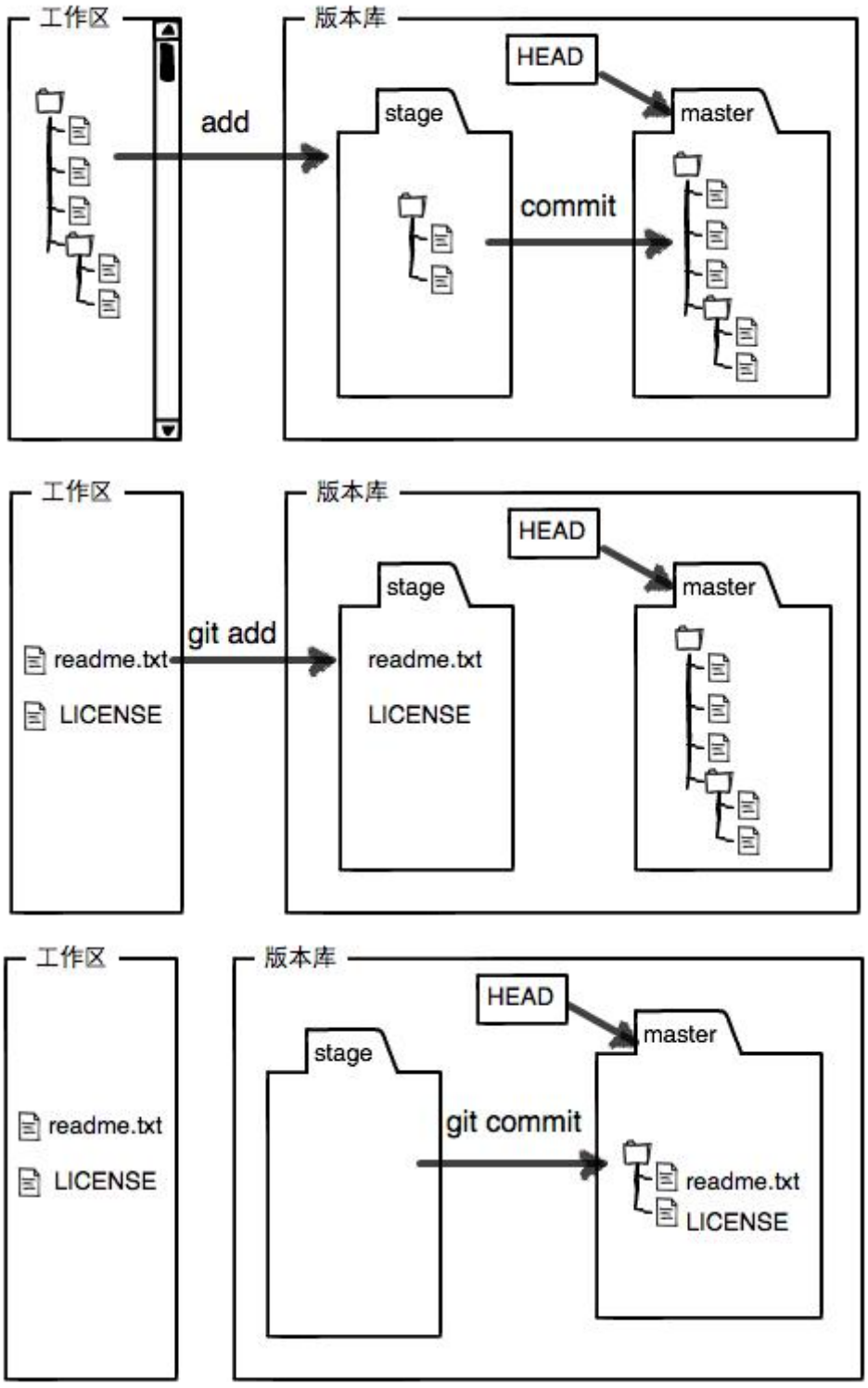
2.时光机穿梭

- 要随时掌握工作区的状态，使用git status命令。
- 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。

2.1 版本回退

- 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。
- 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。
- 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。

2.2 工作区和暂存区



- git add命令实际上就是把要提交的所有修改放到暂存区（Stage），然后，执行git commit就可以一次性把暂存区的所有修改提交到分支

2.3 管理和撤销修改

- 每次修改，如果不用git add到暂存区，那就不会加入到commit中。
- 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：

1. 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；
2. 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。

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

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

2.4 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，使用git checkout – file可以追回。
- 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。

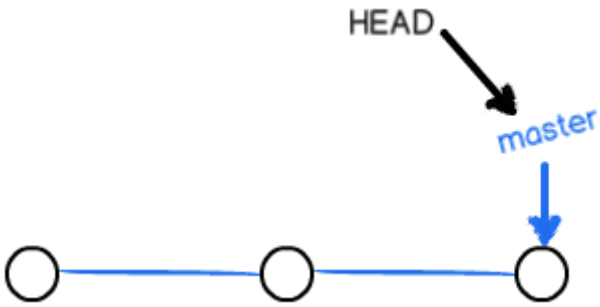
3.远程仓库

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

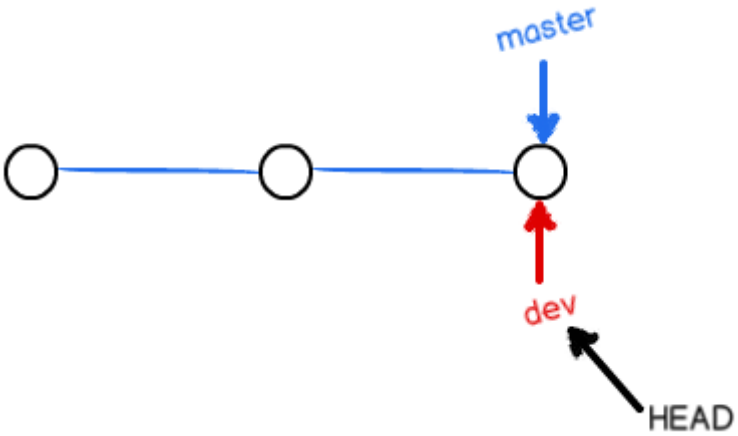
4.分支管理

Git鼓励大量使用分支，主分支即为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。

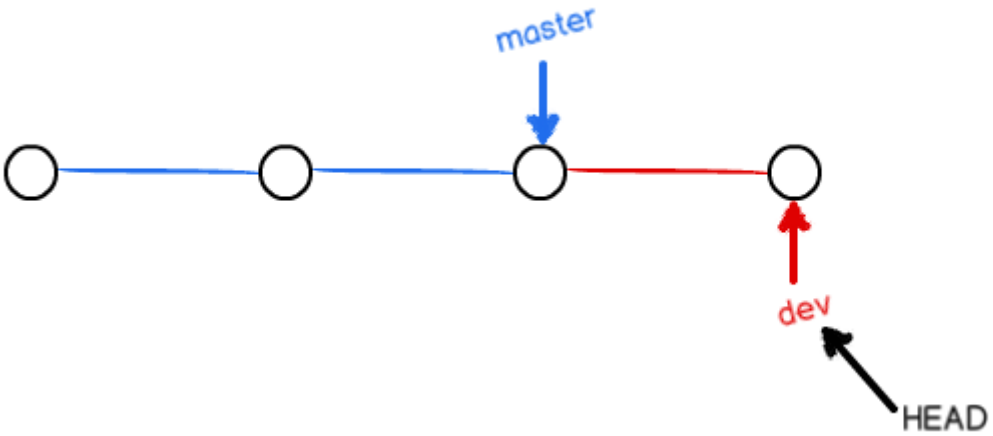
- 仅存在主分支时：



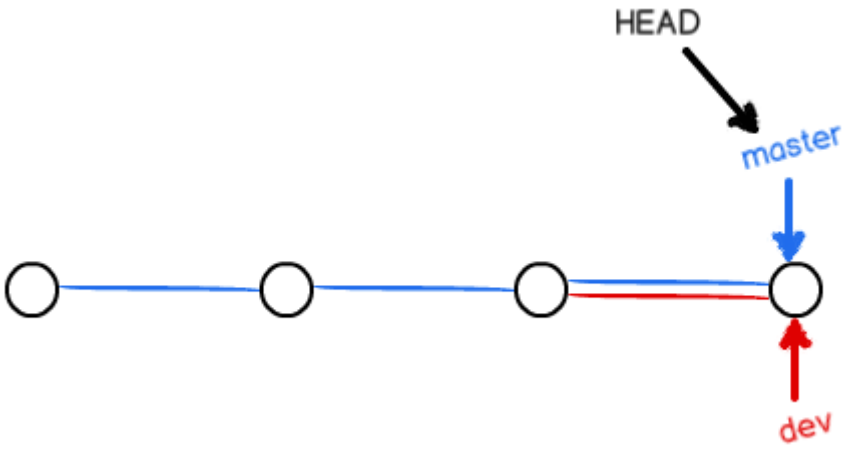
- 创建dev分支时：（等于创建dev指针+使HEAD指向该指针）



- 在dev分支上修改和提交时：（仅针对分支起作用，主分支不变）



- 合并分支时：



相关命令汇总：

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

4.1 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。
- 用git log -graph命令可以看到分支合并图。

4.2 分支管理策略

- 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。

4.3 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；
- 在master分支上修复的bug，想要合并到当前dev分支，可以用git cherry-pick 命令，把bug提交的修改“复制”到当前分支，避免重复劳动。

4.4 Feature分支（新功能）

- 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 强行删除。

4.5 多人合作

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

4.6 Rebase（变基）

- 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。

5.标签管理

- 命令git tag 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 -m “blablabla…“可以指定标签信息；
- 命令git tag可以查看所有标签。
- 命令git push origin 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 可以删除一个本地标签；
- 命令git push origin :refs/tags/可以删除一个远程标签。

6.自定义Git

- 忽略文件的原则是：

1. 忽略操作系统自动生成的文件，比如缩略图等；
2. 忽略编译生成的中间文件、可执行文件等，也就是如果一个文件是通过另一个文件自动生成的，那自动生成的文件就没必要放进版本库，比如Java编译产生的.class文件；
3. 忽略你自己的带有敏感信息的配置文件，比如存放口令的配置文件。

如果你确实想添加该文件，可以用-f强制添加到Git：

```
$ git add -f App.class
```

虽然可以用git add -f强制添加进去，但如果还是希望不要破坏.gitignore规则，可以添加两条例外规则：

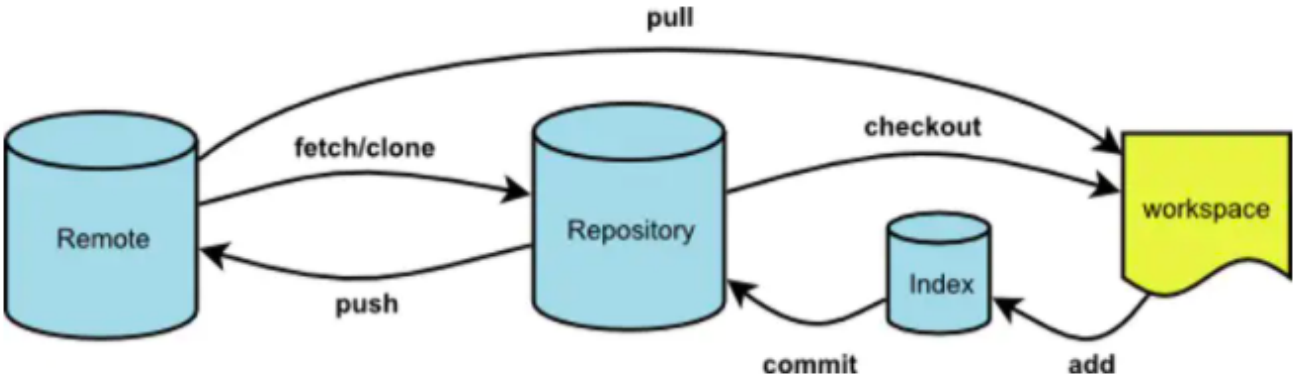
```
# 排除所有.开头的隐藏文件:
.*
# 排除所有.class文件:
*.class

# 不排除.gitignore和App.class:
!.gitignore
!App.class
```

6.1 配置别名

$ git config -global alias.st status $ git config -global alias.co checkout $ git config -global alias.ci commit $ git config -global alias.br branch

7.git fetch vs git pull

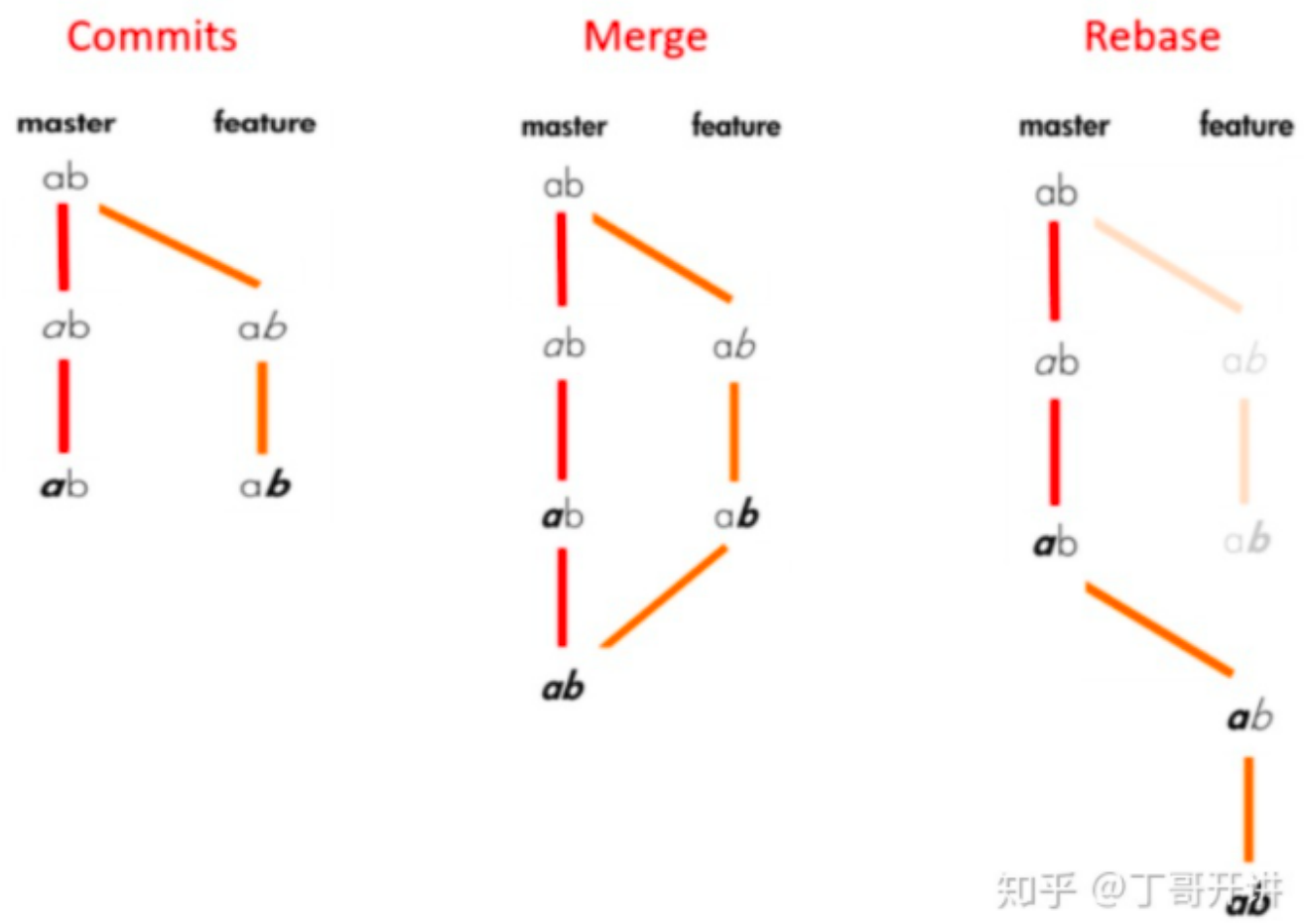
来源



- git fetch是将远程主机的最新内容拉到本地，用户在检查了以后决定是否合并到工作本机分支中。
- git pull 则是将远程主机的最新内容拉下来后直接合并，即：git pull = git fetch + git merge，这样可能会产生冲突，需要手动解决。

8.git merge vs git rebase

来源1



merge 特点：自动创建一个新的commit 来源2

- 如果合并的时候遇到冲突，仅需要修改后重新commit
- 优点：记录了真实的commit情况，包括每个分支的详情
- 缺点：因为每次merge会自动产生一个merge commit，所以在使用一些git 的GUI tools，特别是commit比较频繁时，看到分支很杂乱。

rebase 特点：会合并之前的commit历史

- 优点：得到更简洁的项目历史，去掉了merge commit
- 缺点：如果合并出现代码问题不容易定位，因为re-write了history

合并时如果出现冲突需要按照如下步骤解决

修改冲突部分
git add
git rebase --continue
（如果第三步无效可以执行 git rebase --skip）

不要在git add 之后习惯性的执行 git commit命令

9.子模块

来源

- 添加： git submodule add url为子模块的路径，path为该子模块存储的目录路径。
- 使用： git submodule update -init -recursive
- 更新： 子模块的维护者提交了更新后，使用子模块的项目必须手动更新才能包含最新的提交。 在项目中，进入到子模块目录下，执行 git pull更新，查看git log查看相应提交。 完成后返回到项目目录，可以看到子模块有待提交的更新，使用git add，提交即可。
- 删除：

1. rm -rf 子模块目录 删除子模块目录及源码
2. vi .gitmodules 删除项目目录下.gitmodules文件中子模块相关条目
3. vi .git/config 删除配置项中子模块相关条目
4. rm .git/module/* 删除模块下的子模块目录，每个子模块对应一个目录，注意只删除对应的子模块目录即可

10.Cheatsheet

Git Cheatsheet

#Note #Git
3386 Words
2021-06-27 16:11 +0800

OLDER →
PaperWeekly-2: nlp中的基础知识点整理(partI)