## 20190718学习记录：

学习git，Linux基础命令；

Git命令之：

$ git init 这里将当前的目录变成git可以管理的仓库

$ git add <path> 这里的<path>可以是文件也可以是目录。

$ git commit -m <massage> 提交到仓库,<massage>指备注信息例如”this is an update!”

$ pwd 此命令用于显示当前目录

$git init 此命令是将当前目录变成git可以管理的仓库。

$ls -ah 此命令是用于查看当前路径下的文件（可以显示隐藏文件）

$git status 用于查看所在目录仓库是否有修改而还没提交的

$git log 用于查看当前仓库的修改记录和备注（倒叙，最新的在前）**加上--graph显示各指针之间的图联系（git log graph）**

**$****git log** **--graph --pretty=oneline --abbrev-commit** 可以简化历史提交信息（只包含图，开头七位的版本id和提交的备注信息）

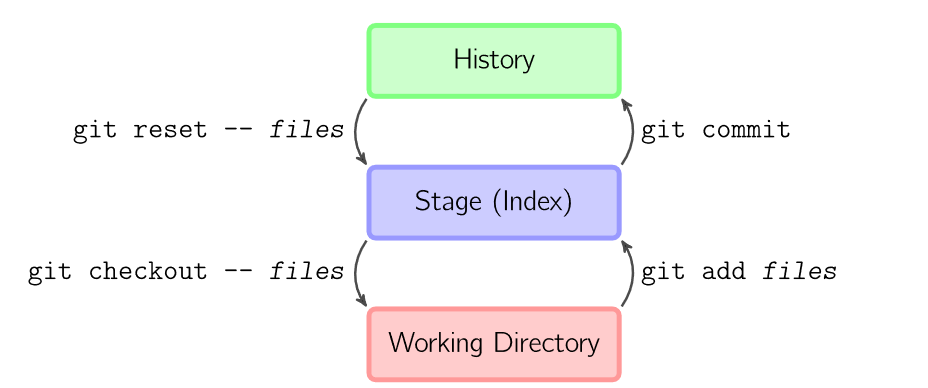
$ cat <file> 可以用于查看当前仓库的文件内容信息

**$git cherry-pick commitid 用于拉取git库中的某一条commit记录到分支**

$git reset –hard HEAD^ 可以用于回退上一个版本 （HEAD^是指上一个版本，HEAD^^是指上上一个版本，依此类推，当然也可以用HEAD~100来表示往上第100个版本）$git reset <commit id >还是可以恢复到指定id的仓库记录（此情况是用于命令行未关闭可以找到commit id 的情况下。）如果已经关闭了命令行窗口，则需要先使用git reflog命令来找到所有的commit id从而达到恢复。

$git checkout --<filename>意思是指把工作区的修改撤销，其中包括两种情况：一种是文件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；一种是文件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。



**问题：git commit 时发现index lock ，初步判断是在上个提交时出现了死锁，无法释放导致，但是没有找到根原因和预防方案，解决方案是：在仓库目录中删除indexlock文件即可。**

$ reset 用于清空git bash命令窗口

## 20190719学习记录：

### 添加远程库和push本地仓库到远程库

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（master为分支名称，本地测试master为默认主分支）；

### 分支的学习和操作

$ git checkout 命令加上-b 表示创建并且切换分支，（例：$git checkout -b dev）

它相当于两条命令：$ git branch dev $git checkout dev

$ git branch 会列出（查看）所有分支，$git branch <branch name>指创建分支

$git checkout <branch name>指切换对应分支

$git branch -d <branch name>删除对应分支

$git merge <branch name>指把此分支合并到当前分支

$git merge –no—ff -m “massage” <branch name>将此分支合并到当前分支

（--no-ff参数，表示禁用Fast forward）；

使用—no-ff参数合并分支时，会要生成一个新的分支，所有要加-m将描述写入，

以上两种合并前者是使用fast forward 模式；后者则是不使用该模式的合并；

**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。**

对于未合并的分支，使用git branch -d <branch name>会失败，并且提示未合并（因为这是为了避免分支数据的丢失），如果该分支确实需要删除，则可以使用git branch -D <branch name>,这样强制删除的话 分支数据将会直接销毁；

**Bug分支的处理（未理解）：**

当前任务不足以提交，但是又突然来了一个紧急的bug需要及时修改时，则需要新增一个分支来对bug进行修改，然后通过git stash来存储当前工作环境，再修改结束bug时，再通过恢复（两种方法，）继续回到之前的任务环境；详情操作如下：

先保存当前的工作现场使用 git stash;

新增分支，处理相应的bug后，删除新增分支

还原现场：1：使用git stash apply ,但是恢复后，stash内容不删除，之后需要使用git stash drop 删除。（在使用stash时可以多次使用，然后用git stash list查看，然后使用git stash apply stash@{index}恢复指定stash！！！）

2：使用git stash pop,这种情况恢复的同时直接把stash的内容也删除了

**多人协作的分支开发：**

$ git checkout -b dev origin/dev 指生成远程origin的dev分支到本地，然后就可以在本地的dev修改并且可以随时push到远程；

如果你的小伙伴已经push了dev分支的远程，并且你也完成了本地dev的修改，并且打算push到远程，则会出现报错，此时我们应该需要pull一下代码，并且进行本地合，可能会出现冲突，解决再提交进行推送（可能存在git pull失败这种情况则需要指定本地dev和远程origin/dev的分支链接：$ git branch --set-upstream-to=origin/dev dev）;

**git clone后, 在自己的git目录下执行上面的配置项, 可以修改提交时的author信息, 如果有submodule还需进入submodule目录 , 执行相同操作.**

**1：**git config user.name zhangjiahui

**2：**git config user.email zhangjh@car-ocean.com

**3：git config –-list //查看作者以及邮箱信息**

**小结抄送：**

因此，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-to <branch-name> origin/<branch-name>。

这就是多人协作的工作模式，一旦熟悉了，就非常简单。

**标签的学习：**

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；(如：git tag v1.0 <commit id>),可以通过git log 查取对应的commit id，再做操作；

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。

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。