**1、本地初始化名为learngit的仓库，并配置gitignore文件，忽略掉所有 .class, .log, .jar, .war文件以及.mtj.tmp文件夹下的所有内容。将该本地仓库推送到个人github中。**

1. **仅执行 clone learngit 仓库的 dev 分支到本地。分别使用 switch和checkout命令在本地创建 feature-1 分支并对应到远程该分支。**
2. **当不在同一个地方的两个开发人员需要pair着完成同一个故事卡内容的开发，如何交替使用每个人的电脑进行协同开发？请写出这个过程中所涉及的 git 命令。例如：在 tdd 的过程中，一个人先在自己的机子上写了测试，然后另一个人需要在自己机子上依据测试来写实现。(提示：可以新建一个协同开发分支，分别提交然后拉取的方式，进行协调工作)**

Git branch dev创建新分支，先分别拉取，写测试和实现，最后分别提交

1. **将本地仓库版本回退到两次提交之前的版本后，突然发现回退错了，想将回退后的仓库版本还原到执行回退操作之前的版本。**

找到回退之前的版本的id，使用git reset --hard 版本id来进行还原，找之前版本id可用“git reflog”来实现。

1. **将本地 feature-1 分支内容推到 github 上 git@github.com:cheny/learngit.git 仓库中的 master 分支上。**

先将本地feature-1分支merge到本地master分支，然后git push

1. **版本发布后，需要给代码库打 tag，tag 名为v0.1，描述信息为version 0.1 released。并将该 tag 推到远程分支。以及如何删除本地和远程仓库名为 v0.9的 tag。**

git tag -a v0.1 -m "version 0.1 released"

git tag -d v0.1

1. **开始了一天的工作，先从 feature-1 分支拉取了最新的代码，然后开始对故事卡的内容进行开发。开发到一半的时候，接到 QA 通知，需要在 dev分支上紧急修复一个 bug。当你在dev分支上修复完成并提交后，又有其他人在 dev分支上提交了新的内容。接着收到通知，需要仅仅将修复该 bug 的提交内容合并到 master 分支上，该修复内容之后的提交不能一起合并到 master 分支上。全部操作完成后，回到 feature-1分支继续进行卡的开发工作。（提示： stash 、 cherry-pick）**

使用git stash将当前分支上的开发内容进行暂存，之后转到dev执行其他代码

在master分支将dev分支上修复bug的提交用git cherry-pickgit cherry-pick <commitHash>提交到master分支。。

回到feature-1分支后，使用git stash pop弹出暂存的内容并应用到当前分支。

1. **显示提交人为 cheny的从 2020.04.02到 2020.08.06的所有提交记录，只显示提交的 message 即可，不需要显示详细内容。**

git log --since=2017-05-21 --until=2017-06-20 --author="author-name"--pretty=format:"%h - %an, %ar : %s"

1. **将远程代码库中的提交回滚到 id 为 01e31c的提交。**

先将本地代码执行git reset -版本id回退到该版本，然后git push到远程。

1. **如何修改倒数第 5 次（HEAD~5）提交的 message 信息。如果要将此次修改同步到远程仓库，如何操作？应该注意什么，如果不注意会造成什么后果**

git rebase -i HEAD~5，进入编辑器将对应提交信息前的pick改成edit，保存，执行git commit --amend后生成的commit是一个全新的commit,之前老的commit会从项目历史中被删除 ，执行 git rebase –continue 提示 Successfully rebased and updated，并git log 查看是否恢复到当前初始状态。

最后执行git push操作

1. **简述 merge 和 rebase的区别。**

生成的代码树不同，merge操作会合并两个分支并生成一个新的提交。git rebase提取操作有点像git cherry-pick一样，执行rebase后依次将当前的提交cherry-pick到目标分支上，然后将在原始分支上的已提取的commit删除。

1. **简述fetch 分别与 pull和 pull --rebase的区别。**

git fetch只是拉取到本地，git pull不仅拉取到本地还merge到本地分支中。所以git pull是git fetch与git merge的集合体。Pull --rebase 是 git fetch + git rebase的集合体

1. **简述 git 中工作区、暂存区和版本库的区别。**

工作区：就是你在电脑里能看到的目录。

暂存区：英文叫stage, 或index。一般存放在 ".git目录下" 下的index文件（.git/index）中，所以我们把暂存区有时也叫作索引（index）。

版本库：工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。

当对工作区修改（或新增）的文件执行 "git add" 命令时，暂存区的目录树被更新，同时工作区修改（或新增）的文件内容被写入到对象库中的一个新的对象中，而该对象的ID被记录在暂存区的文件索引中。

当执行提交操作（git commit）时，暂存区的目录树写到版本库（对象库）中，master 分支会做相应的更新。即 master 指向的目录树就是提交时暂存区的目录树。

当执行 "git reset HEAD" 命令时，暂存区的目录树会被重写，被 master 分支指向的目录树所替换，但是工作区不受影响。

当执行 "git rm --cached <file>" 命令时，会直接从暂存区删除文件，工作区则不做出改变。

当执行 "git checkout ." 或者 "git checkout -- <file>" 命令时，会用暂存区全部或指定的文件替换工作区的文件。这个操作很危险，会清除工作区中未添加到暂存区的改动。

当执行 "git checkout HEAD ." 或者 "git checkout HEAD <file>" 命令时，会用 HEAD 指向的 master 分支中的全部或者部分文件替换暂存区和以及工作区中的文件。这个命令也是极具危险性的，因为不但会清除工作区中未提交的改动，也会清除暂存区中未提交的改动。

1. **简述使用 feature 分支模式进行开发的流程中（拉取代码、创建分支、开发完成后合并分支解决冲突）如何使用一些列的 git 命令。（如 pull, commit, merge, add, fetch,push等）**

拉取代码： git clone/pull

创建分支：git checkout -b

合并冲突：git commit ---->git merge--->git push