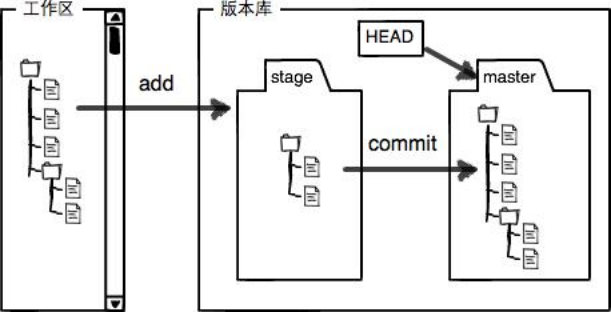
# Git实战

## **一 集中式vs分布式**

集中式版本控制系统：版本库是存放在中央服务器的，每个操作都需要联网与中央服务器进行交互。

分布式版本控制系统：版本库是存放在每个人自己的电脑上，即使没有网络，你也可以想玩就玩。分布式版本控制没有中央服务器概念，其远程服务器主要是用来多人协作时交换数据而已

## **二 工作区&暂存区**



## 三 Git常用命令

假如张三和李四协作开发一个任务，任务jira id为：jira-1。

### 导入远程版本库到本地

命令：git clone remote\_url

示例：git clone <http://gitlab.zhonganonline.com/vehicle_core/za-castle-claim-ms>

### **创建分支**

命令：git branch <分支名> [基准分支名]

示例：

a）git branch test origin/master，表示以origin/master为基准新建test分支，

b）git branch CXJGXM-1 master，表示以本地master分支为基准新建test分支

### **查看分支**

查看本地分支（只显示分支名）git branch

查看本地分支详情（带有最后一次提交信息）：git branch -v

查看所有分支（只显示分支名）：git branch -a

查看所有分支详情（带有最后一次提交信息）：git branch -av

查看本地分支详情及关联分支（带有最后一次提交信息，并显示远程关联分支）：git branch -avv

### **切换分支**

命令：git checkout <本地分支名>

示例：git checkout test，表示切换到test分支

**备注：创建并切换分支命令：git checkout -b <分支名> <基准分支名>**

### **查看工作区状态**

命令：git status

### **建立与远程分支的追踪关系**

命令：git branch --set-upstream-to=<远程主机名>/<远程分支名>

示例：git branch --set-upstream-to=origin/test，表示将当前分支与远程test分支关联

### **添加文件到暂存区**

命令：git add <file>，添加指定文件。git add .添加所有文件

### **将暂存区文件提交**

命令：git commit -m "提交注释"

备注：git commit -am "提交注释",这个是git add .与git commit -m的联合使用。但是对于未跟踪文件，将不能添加进来

### 查看提交历史

命令：git log

最佳实践：设置成别名，详见配置文件

### 代码推送

命令：git push [参数] <远程主机名> <本地分支名>:<远程分支名>

详解：

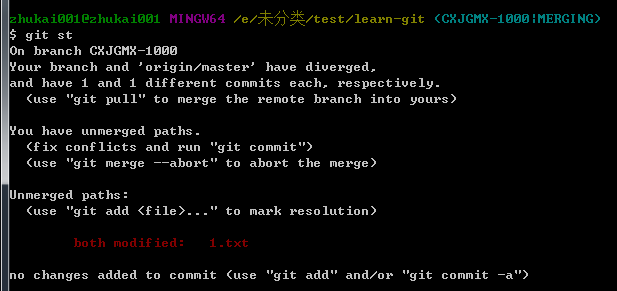
1. 注意<本地分支名>:<远程分支名>，“:”左右不要有空格
2. 参数解释
   1. -u：表示将<本地分支>与<远程分支>建立追踪关系
   2. -f：用<本地分支>【强制覆盖】<远程分支>代码，不用考虑冲突问题，慎用，有血案！
3. 省略写法
   1. 若要推送的<远程分支>与<本地分支>同名，则可以简写为：git push origin <分支名>
   2. 若要将“当前分支”推送到同名的远程分支，且当前分支与该同名的远程分支有追踪关联，则可以简化为：git push origin，若远程主机只有一个，则可以进一步简化为：git push

### 更新代码

1. 执行git fetch，将远程版本库信息更新到本地版本库，工作区代码不变动
2. 执行git rebase，关联的远程分支（存储在本地版本库中）信息将更新工作区文件
3. 执行git merge <分支名>，将指定分支合并到当前分支
4. 执行git pull，效果相当于执行git fetch + git merge
5. 执行git pull -r，效果相当于执行git fetch + git rebase

### 解决冲突

在执行某些命令之后，如git pll, git merge, git rebase，可能会出现文件冲突，执行git status也可以查看到提示“git add <file>... to mark resolution”等，如下图



冲突解决方式如下：

1. 方式一：找到冲突文件，手动更改代码，代码调整好之后，执行【git add <该文件>】标记该文件已解决。依次对所有文件进行相同操作，此时表示所有冲突已解决。然后执行git status，根据提示，后续需要执行的命令，如git commit，或者git rebase --continue等等
2. 推荐方式：使用工作进行代码冲突的解决，如idea

### 查看远程仓库

只显示远程主机名：git remote

显示远程主机名+远程仓库地址：git remote -v

### 添加远程仓库

命令：git remote add <远程主机名> <远程仓库地址>

注意：<远程主机名>可自己随意命名，一般使用origin

备注：如果使用git init方式创建的本地版本库，是布包好远程分支信息的，如果此时想要与某个远程仓库关联，可以使用该命令

### 删除远程仓库

命令：git remote rm <远程主机名>

### 删除分支

删除本地分支：git branch -d <分支名>

强制删除本地分支：git branch -D <分支名>

删除远程分支：git push <远程主机名> :<远程分支名>

### 标签-tag

1. 查看标签：git tag
2. 创建标签：给指定的commit打tag
3. 示例：  
    git tag release\_1.0 29138e0b23013bb12d60cfc6fde80278660ee1da
4. 推送标签到远程版本库

示例：git push origin release\_1.0

1. 删除本地标签：本地删除名为release\_1.0的tag

示例：git tag -d release\_1.0

1. 删除远程标签

示例：git push origin :refs/tags/release\_1.0

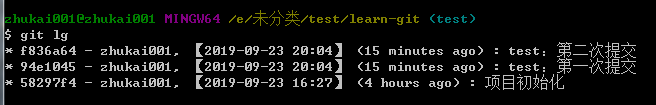
### 删除未跟踪文件

删除当前目录下的未跟踪文件：git clean -f

删除当前目录下的未跟踪文件及其子文件夹：git clean -df

删除当前目录下的未跟踪文件及其子文件夹，连在.gitignore中忽略的未跟踪文件都会被删除，慎用：git clean -xdf

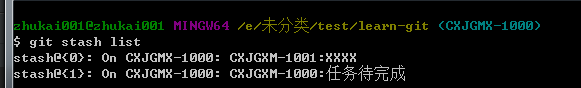
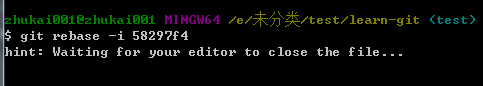
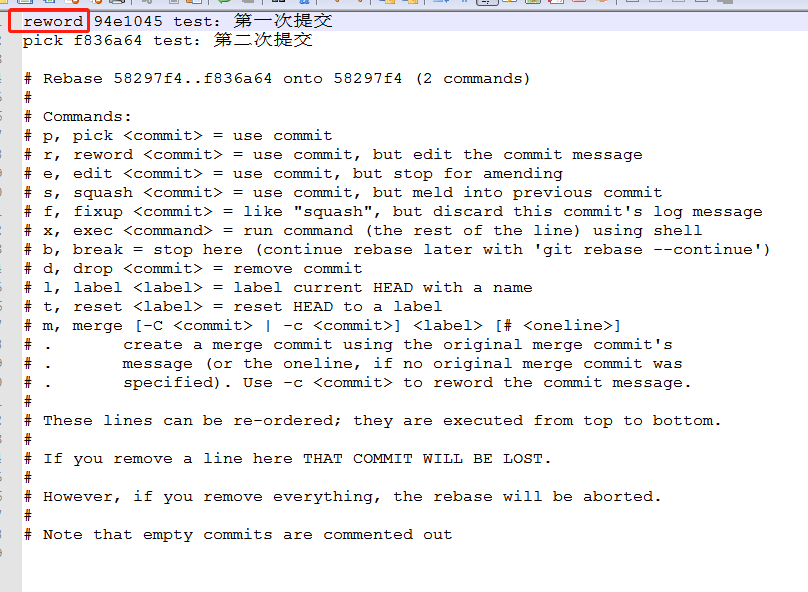
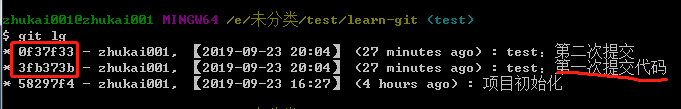
## 四 疑难杂症

1. 如果你开发进行了一小部分，发现逻辑不太对，想要撤销代码。假如此时的代码都只在工作区，且未add到暂存区，该怎么办？
2. 如果你的代码add到了暂存区，但尚未commit，想要撤销，该怎么办？
3. 如果你的代码已经commit了，想要撤销，该怎么办？
4. 撤销已提交的代码，又想要恢复，该怎么办？
5. 如果你的代码已经push了，想要撤销，该怎么办？
6. 如果代码开发进行到了一半，突然有个紧急任务需要处理，又或者需要去其他分支查看代码，排查线上问题，而这部分代码你又不想commit？
7. 代码提交之后，发现之前有的commit备注不太准确，想改改，怎么办？  
   具体示例：如下图，将【commit=94e1045】这个提交的备注改为“第一次提交代码”  
   
8. 代码提交之后，发现之前有的commit，代码存在明显的bug，想在那个提交中将其修改，怎么办？
9. 代码提交之后，发现有些提交显得多余，想要合并到一起，怎么办？

### 解答

1. 答：可以使用git checkout -- <file>命令来回滚工作区指定文件的修改，如果执行git checkout .表示撤销工作区所有文件的修改

**注意**：上面的命令针对git版本管理的文件，而“未跟踪文件”以及“被git忽略的文件”是不会受到影响的

1. 答：可以通过执行命令：git reset HEAD <file>将暂存区的内容放回到工作区，然后再执行git checkout <file>命令来撤销修改
2. 答：可以通过执行命令：git reset --hard commit\_id命令，将工作区及版本库代码回滚到指定commit\_id的版本  
    **扩展如下：**
   1. git reset --hard commit\_id  
      工作区文件**发生改变**，与commit\_id版本代码完全一致，
   2. git reset --soft commit\_id  
      工作区文件**不发生改变**，在commit\_id之后提交的文件全部在暂存区中。
   3. g it reset --mixed commit\_id  
      工作区文件**不发生改变**，在commit\_id之后提交的文件全部在工作区中。
3. 执行git reflog，查找之前的commit\_id，然后执行git reset --hard commit\_id
4. 答：使用git reset --hard commit\_id先将本地版本库回滚到指定commit\_id，然后使用git push -f origin HEAD:<远程分支名>来强制覆盖。注意：这样的话commit\_id之后的提交都会丢失；再者，即使你这么操作了，在你这个操作之前如果有人已经pull过代码，你之前的代码还是存在的。
5. 答：使用git stash save "描述"命令来保存当前工作区的变动。然后去做自己想做的事情，做完之后，在回到该分支，执行git stash pop，将之前的保存提取出来  
    **扩展如下：**
   1. git stash save "描述"：保存工作区的变动
   2. git stash list：查看stash save保存的列表，如下图  
      
   3. git stash pop：提取最新的stash，即stash@{0}，并将其删除
   4. git stash apply [stash@{对应编号}]：提取指定编号的stash，不删除该项stash，若执行git stash
   5. git stash clear：清空所有的stash
6. 答：使用git rebase -i commit\_id解决，commit\_id如何取呢？假如要修改的commit范围是commitA~commitB，那么commit\_id就去commitA的前一个commit。如果要修改第一次的提交，使用git rebase -i --root，具体步骤如下：
   1. 执行命令git rebase -i 58297f4，如下图：  
      
   2. 之后弹出一个文本编辑器，（注意：windows系统一般是弹出一个文本编辑器，mac系统一般默认使用vim编辑），我们将第一个**pick改为reword**，保存并退出！如下图：  
      
   3. 之后弹出新的文本编辑器，将备注修改成你想要的，然后保存并退出，如下图：  
      
   4. 此时执行查看提交历史信息，发下备注已修改，不过需要注意，commit\_id及其之后的SHA值都发生了变化，如下图：  
      
7. 答：使用git rebase -i commit\_id解决，流程如下：
   1. 执行git rebase -i commit\_id
   2. 在编辑器中，对想要修改的commit\_id，将pick改为edit
   3. 此时工作区代码退回到commit\_id这个版本，修改你要改动的地方之后，先执行`git add .`然后执行git commit --amend，在弹出的编辑器中，修改提交信息（当然也可以不修改），然后保存退出。
   4. 以上步骤完成，表示你已经对该提交已作出了修改，此时执行命令：git rebase --continue
   5. 此时一般会出现文件冲突，执行`git status`查看冲突文件，手机将所有冲突文件调整之后，执行“git add .”来标记解决所有冲突文件
   6. 执行git rebase --continue，然后弹出文本编辑器，直接保存退出，此步骤会反复执行，直到结束
8. 答：使用git rebase -i commit\_id解决，对应commit的pick改为squash，详细过程略

## 五 温馨提示

1. 在代码提交之前，将工作区的代码与本地版本库的代码对比，保证提交的修改没有误操作，推荐使用工具比较，如idea
2. 在代码提交push之前，一般工作区是干净的，对比工作区与远程分支代码，保证push的修改没有误操作，推荐使用工具比较，如idea
3. 在文件发生冲突时，推荐使用工具进行解决（如idea），这样讲更加清晰，且避免遗漏。

## 六 配置文件样例

在用户目录下添加配置文件.gitconfig，功能自行百度吧，比较简单

[user]

name = zhukai001

email = zhukai001@zhongan.com

[credential]

helper = store

[alias]

st = status

br = branch

bv = branch -vv

co = checkout

ci = commit

lg = log --graph --pretty=format:\"%h - %an, 【%ad】 (%ar) : %s \" --date=format:\"%Y-%m-%d %H:%M\"

mlg = log --graph --pretty=format:\"%h - %an, 【%ad】 (%ar) : %s \" --date=format:\"%Y-%m-%d %H:%M\" --author=zhukai001