git笔记

1. 安装git之后，做的基本配置

git config --global user.name ‘liugang’

git config --global user.email [‘860721890@qq.com’](mailto:‘860721890@qq.com’)

设置默认主分支的名称：

设置为main

$ git config --global init.defaultBranch main

查看刚才设置的user.name

$ git config user.name

git config --local 本地仓库设置（当前仓库）

git config --global 全局仓库设置（当前用户）

git config --system 所有用户仓库统一设置（当前主机）

1. 初始化git仓库

两种场景：

1. 把已有的项目代码纳入Git管理

$ cd 项目代码所在的文件夹

$ git init

1. 新建的项目直接用Git管理

$ cd 某个文件夹

$ git init your\_project #会在当前路径下创建和项目名称同名的文件夹

$ cd your\_project

然后，在这个文件夹里面新建文件，然后跟踪文件，再提交，提交之后，还可以查看提交历史啊。

跟踪文件：

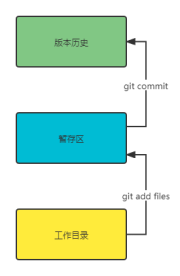
$ git add 文件名

$ git commit -m “新增文件”

查看提交历史：

$ git log

1. Git文件生命周期



查看当前git项目的状态:

$ git status

把被git管理的被改动的文件都加到暂存区：

$ git add -u

1. 给文件重命名最简便的方法
2. 传统方法

$ mv readme readme.txt

$ git add readme.txt

$ git rm readme

$ git status

1. 便捷方法

$ git mv readme readme.md

$ git status

暂存区工作目录所有的操作都会被清楚掉：

$ git reset --hard

1. 查看Git版本历史
2. 简洁的看下Git历史

$ git log --oneline

1. 指定最近的commit

$ git log -n4 --oneline

1. 所有分支演进的历史都要看下

$ git log --all

1. 如3操作之后，历史看不明白，可以使用图形化来看

$ git log --all --graph

1. 如4操作，还可以简洁的图形化来看(所有分支，最近的四个来看)

$ git log --oneline --all -n4 --graph

查看本地有多少分支

$ git branch -v

$ git branch -av

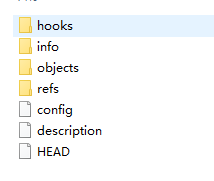
1. 通过图形界面工具查看Git版本历史

$ gitk

变更既有作者（author），还有提交人（commiter）

比如说，master分支有个commit，我很想把它放到temp分支里面去，这个叫（cherry-pick），这个时候，git设计的时候，尊重版权，提交人是你，author是从别的分支拿出来的，当时作者可能是张三。

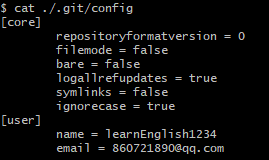
1. 探秘.git文件夹



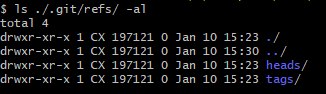
HEAD文件：存储的是，当前指向的分支



config文件：存储的是，配置信息



refs文件夹：存储的是，HEADS文件夹，tags文件夹



查看文件类型(commit,tag,tree,blob)：

$ git cat-file -t aceled245d

查看文件的内容：

$ git cat-file -p aceled245d

一个commit，就是那个文件在那个时刻的一个快照

1. 分离头指针
2. 进一步理解HEAD和branch

基于某个分支去创建新分支：

$ git checkout -b fix\_read fix\_css(某个分支的名称)

基于远程分支新建本地分支

$ git checkout-bprod(本地分支名称) origin/prod(远程仓库别名/远程仓库分支名)

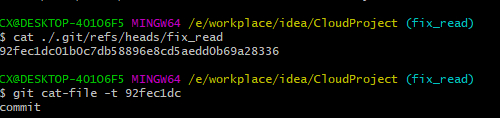
基于当前分支的某个commit点去新建分支

$ git checkout -b fixed-bug-wukuang-branch(分支名称) bc5c4b0a(commit的sha值)

修改本地分支名称

$ git branch -m oldBranchName(旧分支名称) newBranchName(新分支名称)

HEAD落脚于某个分支，分支也是要落脚到某一个commit



HEAD可以表示当前最新的一个commit

使用方法对比如下：

1. 使用传统方法对比：

$ git diff 435ddf1 xfd12

1. 使用HEAD指针替代

$ git diff HEAD HEAD~1

HEAD~1表示parent，那parent的parent如何表示呢

$ git diff HEAD HEAD~2

1. 怎么删除不需要的分支

普通删除：

$ git branch -d xxxx(分支名称)

强制删除：

$ git branch -D xxxxx(分支名称)

删除远程分支：

$ git push origin -d xxxx(分支名称)

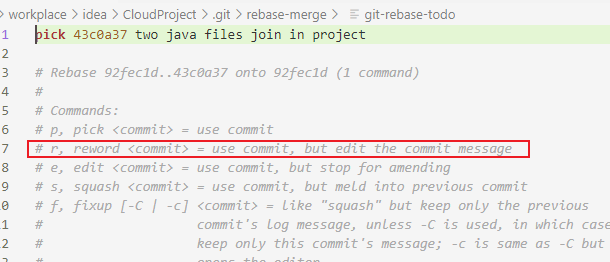
1. 怎么修改最新的commit的message

$ git commit --amend

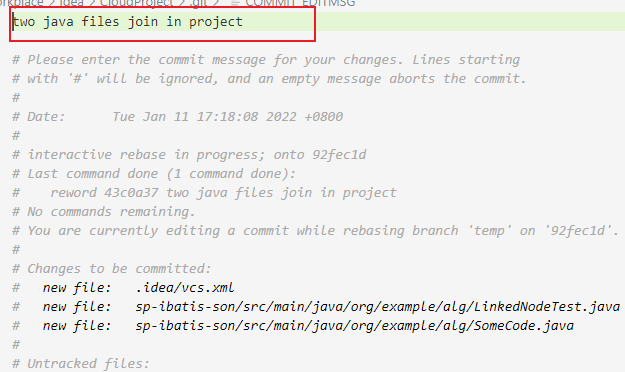
1. 怎么修改老旧的commit的message

$ git rebase -i 92fec1dc01b0c7db58896e8cd5aedd0b69a28336

上面那串乱码是修改目标commit的parent，-i就是交互式的



修改pick为r，保存文件，再退出



再去修改红框里的message内容，保存退出就好了。

1. 怎么把连续的多个commit整理成1个？

$ git rebase -i 92fec1dc01b0c7db58896e8cd5aedd0b69a28336

上面那串乱码是修改目标commit的parent，-i就是交互式的

必须要有一个pick，保留哪个commit

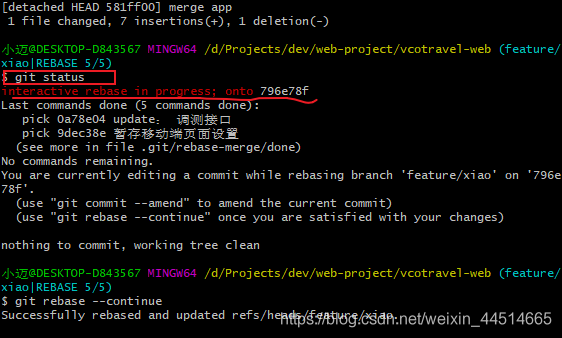


修改pick为s，保存退出



这个pick改成s，应该是在下面把pick改成s，而不是上面的

出现如图这种错误，有两种解决办法。



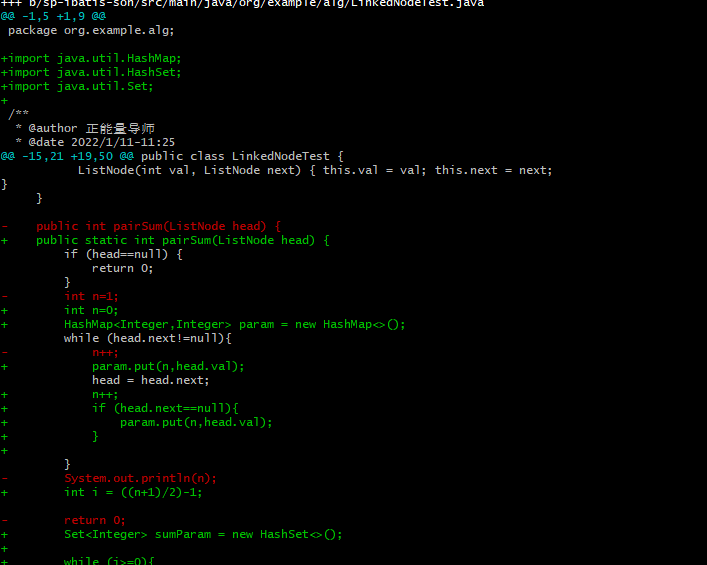
1. git commit --amend修改当前提交
2. git rebase --continue 继续当前提交
3. 暂存区和HEAD如何比较?

把你修改的内容先加入暂存区

$ git add sp-ibatis-son/src/main/java/org/example/alg/LinkedNodeTest.java

再使用对比命令：

$ git diff --cached

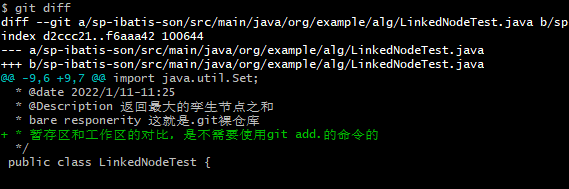


如图，这个+号就是新添加的

1. 工作区和暂存区如何比较?

修改了文件之后啊，直接使用git diff

$ git diff



1. 如何让暂存区恢复成和HEAD一样?
2. 首先可以使用如下命令来比较暂存区和HEAD：

$ git diff --cached

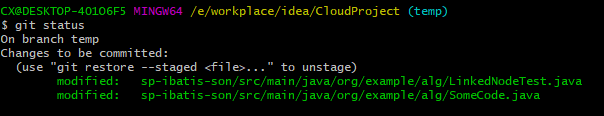
1. 使用如下命令来恢复暂存区和HEAD一致，暂存区的内容就回退到工作区了

$ git reset HEAD

1. 如何让工作区的文件和暂存区一样?

$ git checkout file\_name

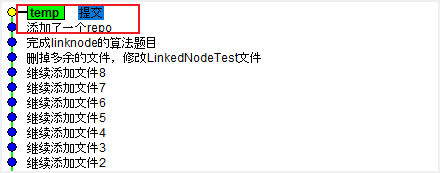
1. 如何取消暂存区部分文件的更改?



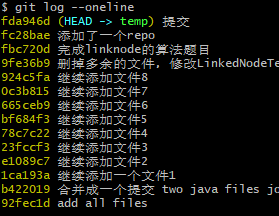
$ git reset HEAD -- file\_name

如果想要取消多个file\_name用空格隔开就好了。

1. 消除最近几次的提交

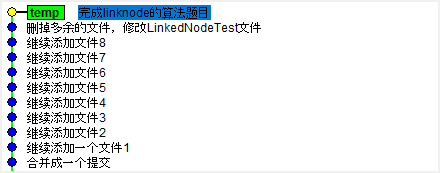


使用gitk查看，前面两个不想要了。



$ git reset --hard fbc720d

再使用gitk看看，上面两个已经被干掉了



1. 看看不同提交的指定文件的差异

对某一些文件看看它们的差别，

$ git diff temp(分支) master(分支)

$ git diff temp(分支) master(分支) --file(具体文件)

还可以diff不同分支的commit带上具体文件



1. 正确删除文件的方法

$ git rm file\_name(文件名)

git就会把直接删除的文件呢，放到暂存区

1. 开发中临时加塞了紧急任务怎么处理？

一部分文件在暂存区，一部分文件还在修改中，突然之间，对应的分支又出问题了，要临时修复这个分支，把手头工作放到临时区域里，修复完成之后，再恢复到以前的状态。

$ git stash

把工作区的代码，还有暂存区的代码，都弄到的堆栈

$ git stash list

$ git stash apply

$ git stash pop

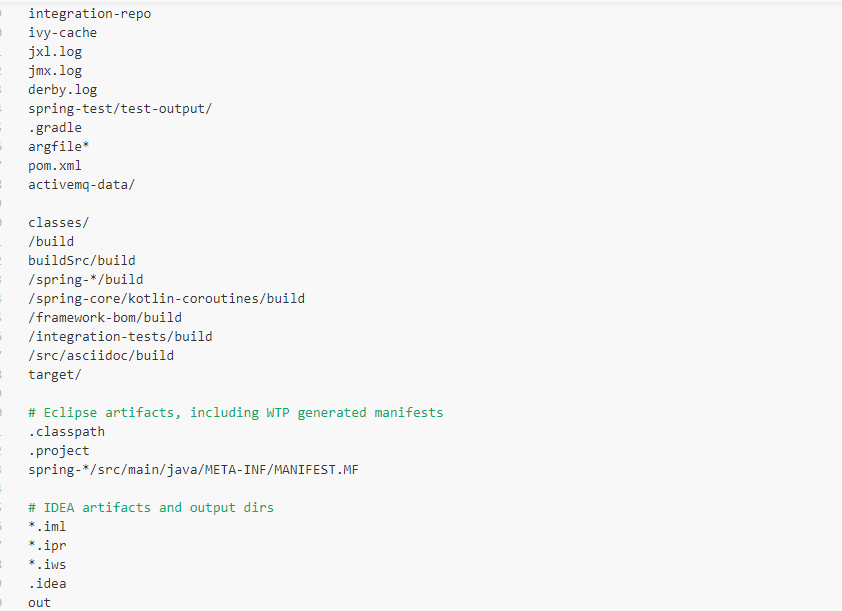
apply呢，那stash还保留着，pop直接删除了

------------------------P28

1. 如何指定不需要Git管理的文件？

在项目文件夹下，新增文件，在这里面设置不被管控的类型，如图所示

.gitignore



简单的说下规则：

1. 开头打#号，就是注释
2. Git如何备份？

常用的协议，哑协议，智能协议，http、https，ssh协议

哑协议，传输进度不可见，智能协议船速课件

传输速度智能协议比哑协议快。

哑协议：

$ git clone --bare 地址 ya.git

智能协议：

$ git clone --bare <file://地址> zhineng.git

1. Git如何跟远端交互？

$ git remote add zhineng <file://地址>

$ git branch -av

$ git push zhineng

$ git push --set-upstream zhineng suling

git pull = git fetch+远端master与本地master合并

rebase是线性的只有一个，merge是有两个parent

报此种错误 refusing to merge unrelated histories

1. Git假设在Github仓库上初始化了一个仓库，里面有一个LECENSE文件，本地也有一个spring-learn的仓库，怎么把local仓库跟Github远端仓库建立连接呢？

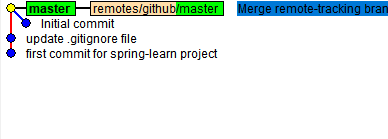
$ git remote add github(远端别名) xxxxxxxxxxxxxx.git(远端地址)

$ git fetch github master

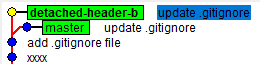
$ git checkout master

$ git merge --allow-unrelated-histories github/master

$ git push github master



1. Git分离头指针是什么回事?



类似这种，当前最新的节点是master，我基于add.gitignore file 节点，checkout了一下

$ git checkout des7d14(add .gitignore file的commitId节点)

这时候master的最新指针和HEAD指针的，不一致了，就是分离头指针(detached HEAD)

我就是要基于add .gitignore file节点去开发新任务对吧，然后修改了文件，然后commit，但是当我们切回master，然后用gitk是看不到这条记录的，因为它没有跟branch绑定起来

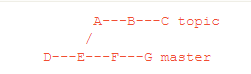
$ git checkout master

$ gitk --all

可以新建分支，把你这个分支和commitId绑定起来，这样就可以了。

$ git branch detached-header-b(分支名称) eedcaad(提交的commitId的值)

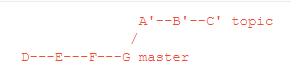
1. Git的rebase变基操作是怎么回事?



当前使用的分支是topic，执行如下命令

$ git rebase master

$ git rebase master topic



这个版本节点变成线性显示了，

把远程6.0.0版本的tag，拉到本地新建分支

Git取消与远程仓库的连接

$ git remote remove origin

新建分支

$ git checkout -b xxxx(分支名称)

如果用gitbash拉远程分支失败，可以使用git gui拉可以成功，或者强制拉取

--.gitignore 文件使用

Git删除.idea文件夹

$ git rm --cached -r .idea

Git删除target文件夹

$ git rm --cached -r target

删除之后commit，再push

github中插入图片

将图片保存在本地文件夹中，再push到github的远程仓库

然后再md中

![image](https://github.com/xx/xxx/blobl/master/xx.png)

git把其他分支的commit点，弄到另外一个分支

git checkout master

git chery-pick xxxxxxxx(commit点)

当出现推送远程仓库失败的情况，fatal: unable to access 'https://github.com/cocodx/spring-security-learn.git/': Failed to connect to github.com port 443: Timed out

或者SSL certificate problem: self signed certificate

可以尝试取消https代理

git config --global --unset https.proxy

**export GIT\_SSL\_NO\_VERIFY=true**

**git config --global http.sslVerify "false"**

打tag

git tag -a v1.0 -m 'release 1.0'

把tag推送到远程仓库

git push github --tags

<https://www.cnblogs.com/senlinyang/p/8527764.html>

远程仓库地址变更了，怎么办？

git remote set-url origin

https://192.168.8.7/enterprise-be/RichMail\_STD\_Server.git