in:name JavaScript 搜索名字带JavaScript的项目

in:readme JavaScript 所搜下面的标题名字

in:description JavaScript 在描述中搜索带JavaScript的

language:java 限制语言为Java

pushed:>2019-09-03 更新时间2019.9.3号后的

stars:>3000 搜索stars数大于3000的（一般上千都是优质的）

forks:>3000 搜索forks数大于3000的

可以同行写多个例如： in:name JavaScript stars:>3000 forks:>3000

全局用户名 username,root\_text

全局邮箱 [aaa@cc.com](mailto:aaa@cc.com)

查看全局的签名 git config -l

1.设置签名

-项目级别/仓库级别：仅在当前本地仓库范围有效

--$ git config user.name XXX

--$ git config user.email xxx@xxx

-系统用户级别：登录当前操作系统的用户范围

-- $ git config --global user.name XXX

-- $ git config --global user.email xxx@xxx

2.本地库的初始化

--- $ git init

3.查看工作区和暂存区的文件状态

--- $ git status

4.上传暂存区, 删除暂存区

--- $ git add XXX 上传

--- $ git rm --cached xxx 删除

5.提交到库

--- $ git commit xxx 如果这样提交需要进入VIM进行解释后才行 用i插入，然后ESC 退出 :wq出去即可

--- $ git commit -m “xxxx” xxx 上面的简单方式

6.查看历史版本

--- $ git log 以最完整的方式打印出来

--- $ git log --pretty=oneline 一行的方式打印出来

--- $ git log –oneline --pretty=oneline的简写方式

--- $ git reflog 比oneline多显示了一个版本，还显示前进和后退的记录

-/\*空格可以翻页 b向上翻页 q退出\*/

-/\*HEAD@{移动到当前版本需要多少步} 🡺 \*/

7.版本的前进和后退

--- 基于索引[推荐]

--$ git reset --hard hash 前进and后退

--- 基于^符号

--$ git reset --hard HEAD^ 有几个^就退几步，这个方法只能后退，并且不显示当 前版本前面的

--- 基于~符号

--$ git reset --hard HEAD~n n是几就退几步，^的简写形式，只能后退

-/\* reset命令的三个参数的对比 \*/

--soft：仅仅在本地库移动指针

-$ git reset –soft hash

--mixed：移动指针 重置暂存区

-

--hard ：移动指针 重置暂存区 重置工作区

8.删除文件

--- rm xxx 删除什么文件

9.找回文件

--- $ git reset –hard hash 回到哪个版本就是恢复

---$ git reset --hard HEAD 添加暂存区的删除文件恢复

10.比较文件差异

---$ git diff xxx 将工作区和暂存区文件比较

---$ git diff HEAD[^|~] 文件名 将工作区和本地库历史版本比较

---$ git diff HEAD 不加文件名会和工作区所有文件进行比较

11.分支操作

---$ git branch [分支名] 创建分支

---$ git branch -v 查看所有分支

---$ git checkout [分支名] 切换分支

---$ git merge [分支名] 合并分支。 用这个前要先切换到要合并到的哪一条分支上， 然后用merge命令把想合并的分支合进来

---当遇到分支冲突的时候，它会把现在的和要合并的不同全给写进去并标记出来，这 时要编辑文件，删除特殊符号，把文件修改到满意的程度保存退出，然后用git add xx 上传在commit -m “”（这时的状态不能带文件名）提交即可解决冲突

1.为地址起别名（作用域只在本初始化的仓库）

---$ git remote -v 查看所有别名

---$ git remote add [想要起的别名] [远程仓库的地址] 为远程仓库起一个别名

-- pull=fatch(拉取) + merge

-$ git fatch [远程仓库别名] [远程分支名] 白话:拉取等于下载

-$ git merge [远程仓库别名/远程分支名] 合并下载分支的代码

-$ git pull [远程仓库别名] [远程分支名] 下载和合并同时做

--$ git push [起的别名 | 远仓地址] [要加入到的分支(master)] 本地推送到远程仓 库

--$ git clone [远程仓库的克隆地址] 从任意远程仓库克隆到本地，克隆完成后会自 动为克隆地址增加别名（origin），还会自动初始化本地库

2.第三方团队的代码怎么添加到自己的远仓库

---先fork克隆别人仓库到自己团队的远程仓库，然后克隆到本地，修改正常的上 传到自己的远程仓库，再然后pull request(一路找绿的点下去)。

---接收对方pull request的人要先点pull request🡺commits(这里查看有哪些提交)/ files shanged（这里可以看到具体有哪些代码被修改了）🡺如果没问题的话回到conversation里，点击merge pull request进行合并，在然后填写日志（相当于commit -m “这里的信息”）