mvn clean source:jar package

mvn deploy

maven私服的账号密码 dev 0788t996j67n62lp

权限的项目 https://git.coding.net/ruicheng/dls-cloud.git

常用的git命令

把这个目录变成Git可以管理的仓库

git init

文件修改添加到暂存区

git add [文件名]

暂存区的所有内容提交到当前分支,我们创建Git版本库时，Git自动为我们创建了唯一一个master分支，所以，现在，git commit就是往master分支上提交更改

git commit -m "worte a file"

一次往暂存区添加多个文件

git add file2.txt file3.txt

看当前仓库的状态

git status

可以查看修改内容

git diff [文件名]

历史记录

git log

git log --pretty=oneline

版本回退

git reset --hard HEAD^

上一个版本就是HEAD^，上上一个版本就是HEAD^^，当然往上100个版本写100个^比较容易数不过来，所以写成HEAD~100

git reset --hard [commit id]

git reset命令既可以回退版本，也可以把暂存区的修改回退到工作区。当我们用HEAD时，表示最新的版本

记录你的每一次命令

git reflog

如果不用git add到暂存区，那就不会加入到commit中

查看工作区和版本库里面最新版本的区别：

git diff HEAD -- readme.txt

撤销修改

git checkout -- readme.txt

命令git checkout -- readme.txt意思就是，把readme.txt文件在工作区的修改全部撤销，这里有两种情况：一种是readme.txt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readme.txt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、

git checkout -- file命令中的--很重要，没有--，就变成了“切换到另一个分支”的命令

git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”



删除版本库文件

用命令git rm删掉，并且git commit：

创建SSH Key

ssh-keygen -t rsa -C "youremail@example.com"

GitHub允许你添加多个Key。假定你有若干电脑，你一会儿在公司提交，一会儿在家里提交，只要把每台电脑的Key都添加到GitHub，就可以在每台电脑上往GitHub推送了，

友情提示，在GitHub上免费托管的Git仓库，任何人都可以看到喔（但只有你自己才能改）。所以，不要把敏感信息放进去。

一个已有的本地仓库与远程仓库关联

git remote add origin https://github.com/wuxiaofangcoder/learngit.git

把本地内容推送到远程

第一次

git push -u origin master

由于远程库是空的，我们第一次推送master分支时，加上了-u参数，

Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令。

以后

git push

克隆远程库到本地

git clone https://git.coding.net/ruicheng/dls-cloud.git

创建dev分支，然后切换到dev分支：

$ git checkout -b dev

git checkout命令加上-b参数表示创建并切换，相当于以下两条命令

git branch dev //创建分支

git checkout dev //切换分支

查看当前分支：

git branch

git branch命令会列出所有分支，当前分支前面会标一个\*号

切换分支

git checkout master

合并分支（当前在master分支上）

git merge dev

删除分支

git branch -d dev

查看分支：git branch

创建分支：git branch <name>

切换分支：git checkout <name>

创建+切换分支：git checkout -b <name>

合并某分支到当前分支：git merge <name>

删除分支：git branch -d <name>

查看分支合并图：

git log --graph --pretty=oneline --abbrev-commit

正常合并分支

git merge dev



如果要强制禁用Fast forward模式，Git就会在merge时生成一个新的commit，这样，从分支历史上就可以看出分支信息

git merge --no-ff -m "merge with no-ff" dev



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。

**分支策略**

在实际开发中，我们应该按照几个基本原则进行分支管理：

首先，master分支应该是非常稳定的，也就是仅用来发布新版本，平时不能在上面干活；

那在哪干活呢？干活都在dev分支上，也就是说，dev分支是不稳定的，到某个时候，比如1.0版本发布时，再把dev分支合并到master上，在master分支发布1.0版本；

你和你的小伙伴们每个人都在dev分支上干活，每个人都有自己的分支，时不时地往dev分支上合并就可以了。

保存工作现场、

git stash

查看工作现场存到哪里？

git stash list

恢复工作现场、

1. A、git stash apply//恢复

B、git stash drop//来删除stash内容；

2. git stash pop stash //恢复的同时把stash内容也删了

注：你可以多次stash，恢复的时候，先用git stash list查看，然后恢复指定的stash，用命令git stash apply stash@{0}

未合并分支强制删除

git branch -D feature-vulcan

查看远程库的信息

git remote

git remote -v

推送分支

git push origin master

如果要推送其他分支，比如dev，就改成git push origin dev

### 多人协作：

因此，多人协作的工作模式通常是这样：

1. 首先，可以试图用git push origin <branch-name>推送自己的修改；
2. 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；
3. 如果合并有冲突，则解决冲突，并在本地提交；
4. 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。

### 小结

* 查看远程库信息，使用git remote -v；
* 本地新建的分支如果不推送到远程，对其他人就是不可见的；
* 从本地推送分支，使用git push origin branch-name，如果推送失败，先用git pull抓取远程的新提交；
* 在本地创建和远程分支对应的分支，使用git checkout -b branch-name origin/branch-name，本地和远程分支的名称最好一致；
* 建立本地分支和远程分支的关联，使用git branch --set-upstream branch-name origin/branch-name；
* 从远程抓取分支，使用git pull，如果有冲突，要先处理冲突。