**Git**

* 常用Git命令：

|  |  |
| --- | --- |
| **初级常用命令** | |
| git config - -list | 查看配置文件 |
| git config --global user.name ‘zhupeng911’  git config --global user.email ‘377964646@qq.com’ | 设置用户名  设置用户邮箱 |
| git status | 掌握工作区的状态 |
| git help config/commit/add  git config/commit/add -h | 获取命令的帮助  获取命令的帮助 |
| **git文件操作** | |
| git init | 初始化为git仓库 |
| git add a.txt | 将文件修改添加到暂存区（stage） |
| git commit -m “first commit” | 提交更改，将暂存区的所有内容提交到当前分支 |
| git push | 本地仓库文件推送到github master分支 |
| git log  **git log --graph**  git log --pretty=oneline | 查看从最近到最远的提交日志，以便确定退回到哪个版本 |
| **git 版本对比** |  |
| git diff | 对比工作区、暂存区内容 |
| git diff head | 对比工作区、最近的一次提交commit |
| git diff 5da78a4 c7c0e3b | 对比两个历史快照 |
| git diff --cached 32e233 | 对比仓库、暂存区 |
| **git版本回退：**reset 和 checkout | |
| **回退快照**  **git reset --hard head~**  **git reset --mixed head~**  **git reset --soft head~**  git reset --hard head~ ~  git reset --hard head~ 5 | 回退到head的上一个快照，--hard回退版本库、暂存区、工作区  回退到head的上一个快照，-- mixed回退版本库、暂存区(**默认**)  回退到head的上一个快照，-- soft回退版本库  回退到head的上上一个快照  回退到head的上5个快照 |
| git reflog | 查看所有回退快照ID |
| **git reset --hard 6c8sdc** | 回退到指定的快照，不仅可以往回滚，也可以往前滚，**必须知道快照ID** |
| **回退文件**  git checkout -- a.txt | 将工作区/暂存区删除的文件恢复到工作目录 |
| **分支管理** | |
| **git branch** feature01 | 创建分支feature01（通过git log查看是否创建成功） |
| git branch | 没有参数时，git branch 会列出你在本地的分支 |
| **git checkout** feature01 | 切换到分支feature01进行操作（git status查看目前处在的分支） |
| git checkout -b (branchname) | 创建一个叫做branchname的临时分支，并切换过去 |
| **git merge** feature01 | 将分支feature01合并到master |
| **git branch -d** feature01 | 删除分支feature01 |
| **git remote add origin** <https://github.com/zhupeng911/git_study.git> | 添加一个远程仓库 |
| **git push -u origin master**  **git push --all origin** | 将代码推送到该远程仓库  将本地的所有分支都推送到远程主机 |
| git remote -v  git remote show [remote rep1] | 列出所有远程仓库  列出某个远程仓库信息 |
| **变基rebase：**将本地多次的commit合并成一个commit， 还可以修改commit的描述等 | |
| git rebase -i head ~ ~  git rebase -i hash值 | 合并前两次的commit  合并前i次commit在最新commit的提交 |

* Maven常用命令

|  |  |
| --- | --- |
| mvn clean | 清理项目生产的临时文件，一般是target目录 |
| mvn compile | 编译源代码，在target目录下生成 项目名.class文件 |
| mvn test-compile | 编译测试代码，一般是src/test目录 |
| mvn package | 将项目打包jar，在target目录下生成jar文件（项目名-版本号） |
| mvn test | 运行测试，或执行src/test/java下junit的测试用例 |
| mvn -v | 显示maven版本信息 |
| mvn install | 将打包的jar文件复制到本地仓库 |
| mvn deploy | 将打包的jar文件发布到远程 |
| mvn site | 生成项目相关信息的网站 |
| mvn idea:idea | 生成IDEA项目 |
| mvn dependency:tree | 打印项目的依赖树 |
| mvn archrtype:create | 创建maven的普通java项目 |

**1、常见git流程**

* Git创建文件

1、cd 当前目录

2、**touch** a.txt 🡪 工作区

3、**git add** a.txt 🡪 工作区—>暂存区

4、**git commit** -m '提交描述' 🡪 暂存区—>Git仓库

5、**git push** 🡪 Git仓库—>Github

* Git修改文件

1、cd 当前目录

2、**vi** a.txt 修改文件

3、**git add** a.txt 🡪 工作区—>暂存区

4、**git commit** -m '提交描述' 🡪 暂存区—>Git仓库

5、**git push** 🡪 Git仓库—>Github

* Git删除文件

1、cd 当前目录

2、**rm** a.txt 删除本地文件

3、**git rm** a.txt 删除Git仓库文件

4、**git commit** -m ‘提交描述’🡪 暂存区—>Git仓库

5、**git push** 🡪 Git仓库—>Github

**2、本地Git仓库更新 -> Github代码托管更新**

0、git init

1、cd git-study git-study是一个本地git仓库

2、echo readme >> README.md 在本地进行操作

git add README.md

git commit -m ‘commit README.md’

3、git remote add origin <https://github.com/zhupeng911/git_study.git> 添加一个远程仓库

4、**git push -u origin master** 将本地最新master分支代码推送到该github远程仓库

5、 git remote -v 列出所有远程仓库

git remote show [remote] 列出某个远程仓库信息

6、**新建分支dev**

git branch dev

git checkout dev

echo this is dev >> dev.txt

git add .

git commit -m 'commit dev.txt'

**git push --set-upstream origin dev** 将本地最新dev分支代码推送到该github远程仓库

7、将dev分支合并到master分支

git checkout master

git merge dev

git push (如果没用，则使用 git push --set-upstream origin master)，更新github

**3、Github团队开发流程**

工作流程：

1、git clone <https://github.com/zhupeng911/git_study.git> 从github上clone开发库dev

2、git checkout dev 转到dev分支

3、git branch -b zhupeng 新建并转到zhupeng分支进行开发

4、 echo do it on branch-zhupeng >> add1.txt

git add .

git commit -m 'branch-zhupeng:commit add1'

5、将分支zhupeng合并到分支dev

git checkout dev

git merge zhupeng

**git push --set-upstream origin zhupeng**

**4、仓库恢复**

图示

描述已自动生成

git reset --hard HEAD^ 退回到上一个版本

git reset --hard HEAD^^ 退回到上上一个版本（HEAD指向当前的版本，git允许在版本的历史之间穿梭）

git reset --hard 版本id 返回到最新版本

git reset HEAD "要撤销的文件名" 可以撤销添加到暂存区的内容

git checkout -- "要撤销的文件名" 撤销工作区修改的内容（修改了工作区内容还没有添加到暂存区 + 添加到暂存区后，修改了工作区的内容）

**5、分支管理**

master分支是一条线，git用master指向最新的提交，再用HEAD指向master，就能确定当前分支，以及当前分支的提交点（HEAD指向当前的分支）

图示

描述已自动生成

**工作流程：**

1、git branch feature01 创建分支feature01（通过git log查看是否创建成功）

2、git checkout feature01 切换到分支feature01进行操作（git status查看目前处在的分支）

3、echo dddd on feature01 >> d.txt

git add .

git commit -m ‘commit d.txt on feature01’

此时分支feature01有d.txt，但是master分支还没有

4、git checkout master 切换到分支master

5、git merge feature01 将分支feature01合并到master

图示

描述已自动生成

6、git branch -d feature01 删除分支feature01

**Maven**

|  |  |
| --- | --- |
| mvn clean | 清理项目生产的临时文件，一般是target目录 |
| mvn compile | 编译源代码，在target目录下生成 项目名.class文件 |
| mvn test-compile | 编译测试代码，一般是src/test目录 |
| mvn package | 将项目打包jar，在target目录下生成jar文件（项目名-版本号） |
| mvn test | 运行测试，或执行src/test/java下junit的测试用例 |
| mvn -v | 显示maven版本信息 |
| mvn install | 将打包的jar文件复制到本地仓库 |
| mvn deploy | 将打包的jar文件发布到远程 |
| mvn site | 生成项目相关信息的网站 |
| mvn idea:idea | 生成IDEA项目 |
| mvn dependency:tree | 打印项目的依赖树 |
| mvn archrtype:create | 创建maven的普通java项目 |

注意：运行maven命令必须先定位到maven项目的目录，也就是pom.xml的目录。否则必须通过参数来指定项目的目录。

命令参数

1、-D传入属性参数

mvn package -Dmaven.test.skip=true maven打包的时候跳过单元测试

mvn deploy -Dmaven.test.skip=true maven部署的时候跳过单元测试

2、-P使用指定的Profile环境配置

一般项目开发有开发、测试、预发、正式四个环境，在pom.xml的配置如下:

文本

描述已自动生成

mvn package -Pdev -Dmaven.test.skip=true 将dev环境的项目打包部署