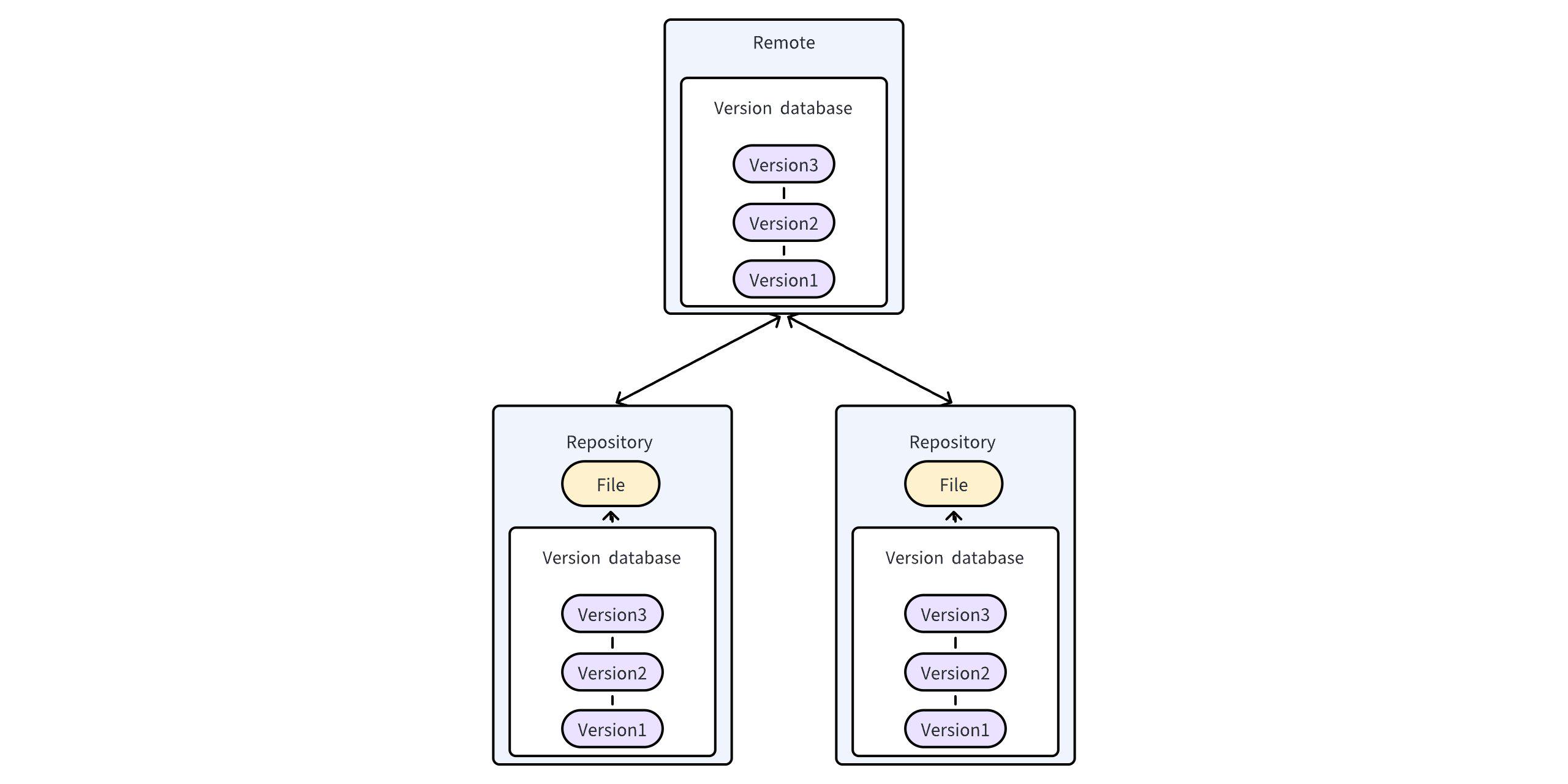
**git常用命令**

**1.是什么**

**一个分布式版本控制系统**

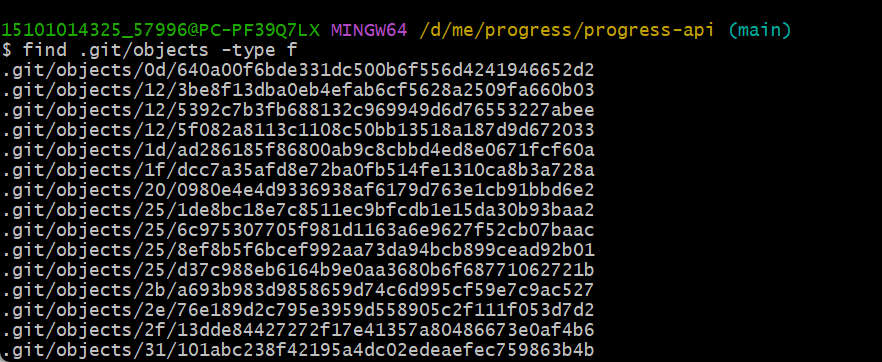
客户端不只提取最新版本的文件快照， 会把代码仓库完整地镜像下来，包括完整的历史记录。每一次的克隆操作，都是一次对代码仓库的完整备份。



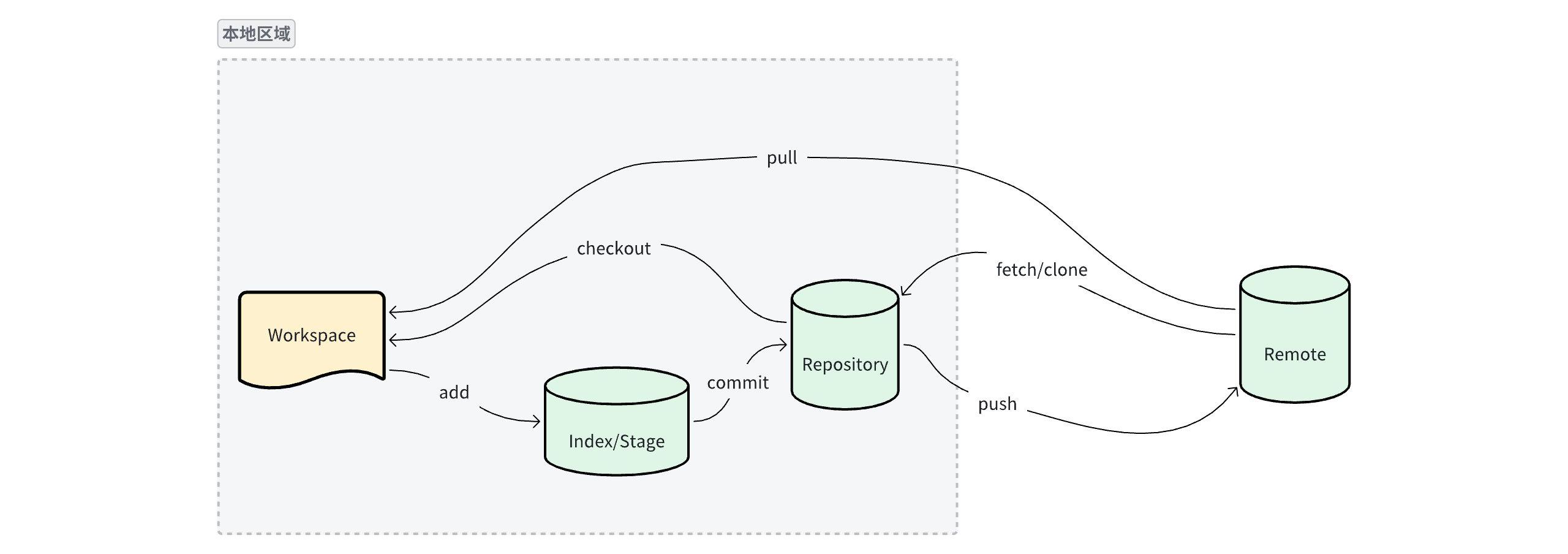
**2.基本原理**

Git是一个内容寻址文件系统，核心部分是一个简单的键值对数据库（key-value data store）。可以向Git仓库中插入任意类型的内容，会返回一个唯一的键，通过该键可以在任意时刻再次取回该内容。

对象数据库，保存着git的对象，其中最重要的是blob、tree和commit对象。blob数据对象记录内容，tree对象记录文件名和文件目录结构，commit对象记录版本的信息。



**3.工作流程**



* Workspace：工作空间

本地项目存放文件的位置，代码增删修改即使更改的区域。

* Index/Stage：暂存区

暂时存放文件的地方，通过add命令将工作区的文件添加到暂存区区。

* Repository：本地仓库

通过commit命令可以将暂存区的文件添加到本地仓库，HEAD指针指向最新一次commit的分支，本地仓库中存在多个历史记录的版本。

* Remote：远程仓库

远端托管代码的服务器。通过clone命令将远程仓库代码拷贝下来，本地代码更新后，通过push推送给远程仓库。

**4.常用命令**

**1.git add**

将工作区修改的内容添加到暂存区

|  |
| --- |
| Bash git add [文件名] #单个文件 **git add -A** #工作空间增删改所有修改 git add . #当前文件和子文件，git2.x版本等同于git add -A，1.x 不会处理删除的文件 git add -u #修改和删除的文件，不包括新增的文件 |

**2.git commit**

将暂存区的内容添加到本地仓库

|  |
| --- |
| Bash **git commit -m "feat:m-xxx message"** #将暂存区内容提交到本地仓库 git commit -am **"feat:m-xxx message"** # 跳过git add, 将所有已被跟踪的文件更改提交到版本库 **git commit --amend -m "feat:m-xxx message"**  #用新的commit, 替代上一次提交  #如果代码没有任何新变化, 则用来改写上一次commit的提交信息  #多用于修改上一次的提交信息或者合并两次功能相似的提交  #最好期间没有别人提交过，否则替代的是别的的提交信息 |

**3.git status**

查看文件状态

|  |
| --- |
| Bash Untracked files #新增还没有add的file，只存在于工作区 Changes to be committed #add但是没有commit的file，存在于工作区和暂存区 Changes not staged for commit #表示工作区，暂时区都存在的file，工作区修改后没有add  Your branch is ahead of 'origin/main' by 2 commits.  (use "git push" to publish your local commits) #本地仓库存在没有push到远程仓库的commit |

**4.git diff**

查看文件差异

|  |
| --- |
| Bash git diff #查看工作区和暂存区的文件差异 git diff -- [文件名] #查看指定文件工作区和暂存区的文件差异 |

**5.git pull**

拉取远程仓库完整代码更新本地代码 git pull = git fetch + git merge

|  |
| --- |
| Bash **git pull** #拉去远程仓库完整代码到本地 git pull origin <远程分支名> #将远程指定分支 拉取到 本地当前分支上 git pull origin <远程分支名>:<本地分支名> #将远程指定分支 拉取到 本地指定分支上 |

**6.git push**

本地仓库的commit推送到远程仓库

|  |
| --- |
| Bash **git push** #当前分支的commit推送到远程绑定分支上 git push origin <本地分支名> #未建立绑定关系使用 git push -f #强制推送 慎用！会导致远程主机上更新的版本被覆盖   #如果没有指定强制push的分支，可能会把所有的本地分支强制覆盖远程分支  #git push origin <本地分支名> -f 指定分支名  #idea的terminal中 git push 是把当前所在的分支，推送到远程分支，可不用指定分支名  #git commit --amend -m [message] 必须使用强推   删除远程分支： git push origin :分支名 (origin 后面有空格) |

**7.git log**

查看提交历史，可以查看要回退到那个版本

|  |
| --- |
| Bash git log #产看提交历史   commit b133ffddcfb3784b8703c551fa317ce00ea0cc9a  Author: wangxiaoning   Date: Thu Aug 31 16:53:50 2023 +0800    测试   git log --graph --pretty=oneline --abbrev-commit #查看分支合并情况   \* | 702e6d5 (origin/master, origin/HEAD) Merge branch 'release-2.24.0' into 'master'  |\|   | \* bf39c2b feat:m-15646166 交付物修改名称后相关流程版本加1  | \* f475dea Merge remote-tracking branch 'origin/release-2.24.0' into release-2.24.0  | |\  | | \* b603968 feat:m-13773807 1、获取用户中心权限信息日志级别修改  | \* | 6d4e509 feat: m-15646165 调整UC返回值 |

**8.git merge**

合并操作

|  |
| --- |
| Bash git merge 分支名 #指定分支合并到当前分支 |

**9.git branch**

分支管理

|  |
| --- |
| Bash git branch #查看所有本地分支 git branch -r #查看所有远程分支 git branch -a #查看所有本地和远程分支 git branch -d 分支名 #删除分支，如果该分支有提交未进行合并，则会删除失败。 git branch -D 分支名 #强制删除分支，如果该分支有提交未进行合并，也会删除成功。 git branch -m 分支名 新分支名 #重命名 git branch 分支名 #创建分支 |

**10.git checkout**

分支管理

|  |
| --- |
| Bash git checkout 分支名 #切换到指定分支 git checkout -b 分支名 #在本地当前分支的基础上，新建一个新分支 git checkout -b 本地分支名 origin/远程分支名 #拉取远程分支去创建本地分支 git checkout -b 分支名 某次commit\_id #用某个commit新建分支 |

**11.git fetch**

分支管理

|  |
| --- |
| Bash git fetch origin 远程分支名:本地分支名 #用远程分支新建分支 |

**12.git reset**

回退

|  |
| --- |
| Bash git reset --hard HEAD #当前版本 git reset --hard HEAD^ #回退到上个版本 git reset --hard HEAD^^ #回退到上上个版本 git reset --hardHEAD~n #回退n个版本 git reset --hard commit-id #回退到指定commit git reset --soft HEAD^ #撤销上次commit保留修改内容 |

**13.git stash**

缓存操作

|  |
| --- |
| Bash git stash list #查询缓存池记录 git stash save 注释 #缓存当前的工作区与暂存区的状态并添加注释 git stash pop #应用最近的一次缓存并删除 git stash pop num #应用指定缓存并删除 git stash apply #应用指定缓存但不删除 git stash drop #删除最近一次缓存 git stash drop num #删除指定缓存 git stash clear ##清楚所有缓存 |

**14. git clone**

克隆仓库

|  |
| --- |
| Bash git clone URL #克隆远程仓库到本地 |

**5.常见问题**

**1.上次commit漏提文件或者提交信息不准确需要修改**

|  |
| --- |
| Bash 第一步：git add -A 第二步：**git commit --amend "message"** 第三步：git push / git push -f 如果上次commit已经push需要强推 git push -f |

**2.工作区修改内容和远程仓库冲突，git pull报错**

|  |
| --- |
| Bash 第一步：**git stash save** 第二步：git pull 第三步：**git stash pop** 第四步：merge冲突 第五步：git add -A 第六步：git commit **-m "message"** 第七步：git push |

**3.代码回退**

|  |
| --- |
| Bash 1、撤销工作区修改(尚未add，尚未commit)  git checkout -- 文件名（带路径） 2、撤销暂存区修改(已经add，尚未commit)  git reset HEAD 文件名（带路径）, 再git checkout -- 文件名（带路径） 3、撤销版本库修改(已经commit，尚未推送到远程)  git reset --hard HEAD^(版本回退) |

**3.代码回退--已push**

|  |
| --- |
| Bash 第一步：git add -A 第二步：git commit --amend **"message"** 第三步：git push 如果上次commit已经push需要强推 git push -f |