1. git 分布式版本控制软件

2. 工作区：自己写代码的地方

暂存区：临时存储的地方

本地库：存放历史版本的地方；

3.git 和代码托管中心

代码托管中心：帮我们维护远程库；

本地库和远程库的交互方式

团队内：

图示

描述已自动生成

团队外：

图示

描述已自动生成

**4.git 命令操作**

1.git init 初始化git respository

2.设置签名：

项目级别/仓库级别：仅在当前本地库范围有效；

git config user.name menglei

git config user.email [3082298291@qq.com](mailto:3082298291@qq.com)

系统用户级别：登录当前的操作系统的用户范围

项目级别优先于系统用户级别

user.name & user.email 存放在.git/config文件中

git config --global user.name menglei

git config --global user.email [3082298291@qq.com](mailto:3082298291@qq.com)

文本

描述已自动生成

**常见命令：**

git commit -m "提交的信息" 【修改的文件】 提交代码

git log --pretty=oneline 每一次提交显示一行

git log --oneline 提交的哈希值（索引值）只显示一部分

git reflog 显示移动的参考指针

**前进后退**

> 基于索引值后退：git reset --hard 【索引值】 可以前进后退,配合git reflog使用

>使用^符号 ：git reset --hard HEAD^

git reset --hard HEAD~3 后退三步

只能后退，配合git reflog使用

**reset命令的三个参数对比**

> --soft

1.仅仅在本地库移动HEAD指针

图片包含 图形用户界面

描述已自动生成

>--mixed

1.在本地库移动HEAD指针

2.重置暂存区

图形用户界面, 应用程序, Word

描述已自动生成

>--hard

1.在本地库移动HEAD指针

2.重置暂存区

3.重置工作区

图片包含 图形用户界面

描述已自动生成

**本地删除的文件如何找回**

1.git add .

2. git commit

3. git reset --hard HEAD^

前提：文件删除前，文件存在的状态已经提交到本地库；

操作：git reset --hard 【指针所在的位置】

> 删除操作已经提交到本地库，指针位置指向历史记录

> 删除操作尚未提交到本地库，指针位置使用HEAD;

**比较文件差异**

>git diff [文件名] 将工作区的文件和暂存区的文件进行比较

>git diff [本地库中的历史版本] [文件名] 将工作区的文件和本地库中的历史版本进行比较

>git diff 比较多个文件

**分支管理**

图示

中度可信度描述已自动生成

**常用命令**

git branch -v 查看所有分支

git checkout [分支名] 切换分支

git branch [分支名] 创建分支

**合并分支**

1.切换到接受修改的分支，进行修改

git checkout [接受修改的分支]

edit

git add

git commit

2.切换到master分支，执行merge

git checkout master

git merge [第一步修改的分支]

**解决冲突**

文本

描述已自动生成

**push 远程库**

复制github上面的仓库的链接

1.git remote -v 查看别名

2.git remote add origin [<https://github.com/menglei999/testgit.git>] 给仓库创建别名

3.git push master origin 将本地库推送到远程库

4.git clone [仓库链接] 注意：要在一个空的文件夹下面执行该操作

>克隆一份代码

>初始化本地库

>创建origin 远程地址别名：