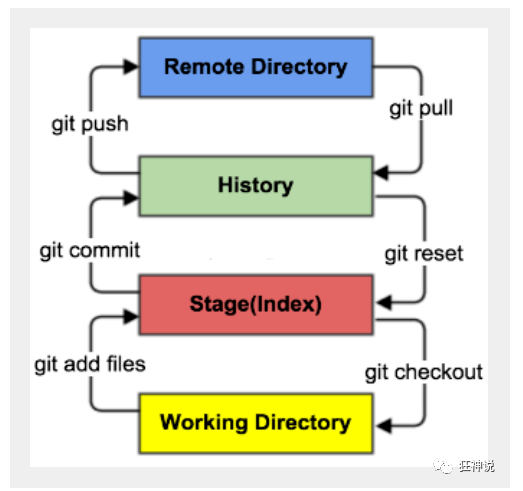
# Git相关理论

链接：https://mp.weixin.qq.com/s/Bf7uVhGiu47uOELjmC5uXQ

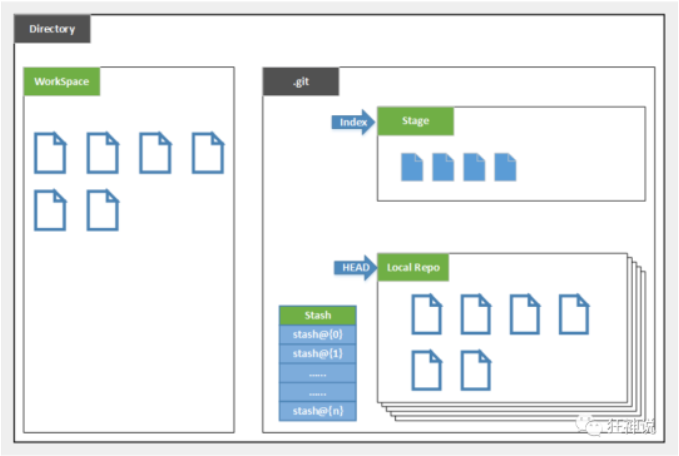
## 工作区域

Git本地有三个工作区域：工作目录（Working Directory）、暂存区(Stage/Index)、资源库(Repository或Git Directory)。如果在加上远程的git仓库(Remote Directory)就可以分为四个工作区域。文件在这四个区域之间的转换关系如下：



* Workspace：工作区，就是你平时存放项目代码的地方
* Index / Stage：暂存区，用于临时存放你的改动，事实上它只是一个文件，保存即将提交到文件列表信息
* Repository：仓库区（或本地仓库），就是安全存放数据的位置，这里面有你提交到所有版本的数据。其中HEAD指向最新放入仓库的版本
* Remote：远程仓库，托管代码的服务器，可以简单的认为是你项目组中的一台电脑用于远程数据交换

本地的三个区域确切的说应该是git仓库中HEAD指向的版本：



Directory：使用Git管理的一个目录，也就是一个仓库，包含我们的工作空间和Git的管理空间。

WorkSpace：需要通过Git进行版本控制的目录和文件，这些目录和文件组成了工作空间。

.git：存放Git管理信息的目录，初始化仓库的时候自动创建。

Index/Stage：暂存区，或者叫待提交更新区，在提交进入repo之前，我们可以把所有的更新放在暂存区。

Local Repo：本地仓库，一个存放在本地的版本库；HEAD会只是当前的开发分支（branch）。

Stash：隐藏，是一个工作状态保存栈，用于保存/恢复WorkSpace中的临时状态。

## 工作流程

git的工作流程一般是这样的：

１、在工作目录中添加、修改文件；(UserMapper.xml)

２、将需要进行版本管理的文件放入暂存区域；(git add .)

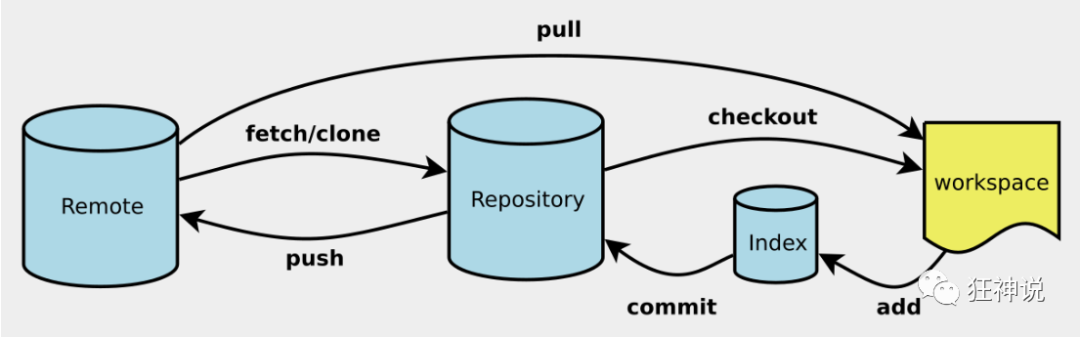
３、将暂存区域的文件提交到git仓库。(git commit)

因此，git管理的文件有三种状态：已修改（modified）,已暂存（staged）,已提交(committed)

## 创建工作目录与常用指令

工作目录（WorkSpace)一般就是你希望Git帮助你管理的文件夹，可以是你项目的目录，也可以是一个空目录，建议不要有中文。

日常使用只要记住下图6个命令：



主要使用：add、commit、push、clone

**本地仓库搭建**

创建本地仓库的方法有两种：一种是创建全新的仓库，另一种是克隆远程仓库。

1、创建全新的仓库，需要用GIT管理的项目的根目录执行：

# 在当前目录新建一个Git代码库

$ git init

2、执行后可以看到，仅仅在项目目录多出了一个.git目录，关于版本等的所有信息都在这个目录里面。

**克隆远程仓库**

1、另一种方式是克隆远程目录，由于是将远程服务器上的仓库完全镜像一份至本地！

# 克隆一个项目和它的整个代码历史(版本信息)

$ git clone [url] # https://gitee.com/kuangstudy/openclass.git

2、去 gitee 或者 github 上克隆一个测试！

## Git文件操作

**文件的四种状态**

版本控制就是对文件的版本控制，要对文件进行修改、提交等操作，首先要知道文件当前在什么状态，不然可能会提交了现在还不想提交的文件，或者要提交的文件没提交上。

Untracked: 未跟踪, 此文件在文件夹中, 但并没有加入到git库, 不参与版本控制. 通过git add 状态变为Staged.

Unmodify: 文件已经入库, 未修改, 即版本库中的文件快照内容与文件夹中完全一致. 这种类型的文件有两种去处, 如果它被修改, 而变为Modified. 如果使用git rm移出版本库, 则成为Untracked文件

Modified: 文件已修改, 仅仅是修改, 并没有进行其他的操作. 这个文件也有两个去处, 通过git add可进入暂存staged状态, 使用git checkout 则丢弃修改过, 返回到unmodify状态, 这个git checkout即从库中取出文件, 覆盖当前修改 !

Staged: 暂存状态. 执行git commit则将修改同步到库中, 这时库中的文件和本地文件又变为一致, 文件为Unmodify状态. 执行git reset HEAD filename取消暂存, 文件状态为Modified

**查看文件状态**

上面说文件有4种状态，通过如下命令可以查看到文件的状态：

#查看指定文件状态

git status [filename]

#查看所有文件状态

git status

# git add . 添加所有文件到暂存区

# git commit -m "消息内容" 提交暂存区中的内容到本地仓库 -m 提交信息

**忽略文件**

有些时候我们不想把某些文件纳入版本控制中，比如数据库文件，临时文件，设计文件等

在主目录下建立".gitignore"文件，此文件有如下规则：

1. 忽略文件中的空行或以井号（#）开始的行将会被忽略。
2. 可以使用Linux通配符。例如：星号（\*）代表任意多个字符，问号（？）代表一个字符，方括号（[abc]）代表可选字符范围，大括号（{string1,string2,...}）代表可选的字符串等。
3. 如果名称的最前面有一个感叹号（!），表示例外规则，将不被忽略。
4. 如果名称的最前面是一个路径分隔符（/），表示要忽略的文件在此目录下，而子目录中的文件不忽略。
5. 如果名称的最后面是一个路径分隔符（/），表示要忽略的是此目录下该名称的子目录，而非文件（默认文件或目录都忽略）。

#为注释

\*.txt #忽略所有 .txt结尾的文件,这样的话上传就不会被选中！

!lib.txt #但lib.txt除外

/temp #仅忽略项目根目录下的TODO文件,不包括其它目录temp

build/ #忽略build/目录下的所有文件

doc/\*.txt #会忽略 doc/notes.txt 但不包括 doc/server/arch.txt

## 在vs code 中的操作

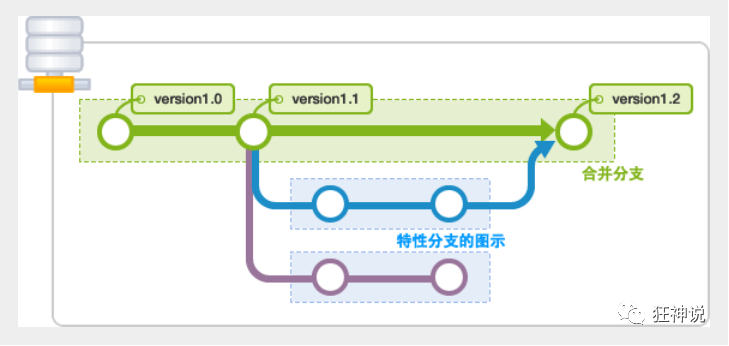
修改完代码在终端运行

1. git add .
2. git commit -m “message”
3. git push

## GIT分支

分支在GIT中相对较难，分支就是科幻电影里面的平行宇宙，如果两个平行宇宙互不干扰，那对现在的你也没啥影响。不过，在某个时间点，两个平行宇宙合并了，我们就需要处理一些问题了！





git分支中常用指令：

# 列出所有本地分支

git branch

# 列出所有远程分支

git branch -r

# 新建一个分支，但依然停留在当前分支

git branch [branch-name]

# 新建一个分支，并切换到该分支

git checkout -b [branch]

# 合并指定分支到当前分支

$ git merge [branch]

# 删除分支

$ git branch -d [branch-name]

# 删除远程分支

$ git push origin --delete [branch-name]

$ git branch -dr [remote/branch]

如果同一个文件在合并分支时都被修改了则会引起冲突：解决的办法是我们可以修改冲突文件后重新提交！选择要保留他的代码还是你的代码！

master主分支应该非常稳定，用来发布新版本，一般情况下不允许在上面工作，工作一般情况下在新建的dev分支上工作，工作完后，比如上要发布，或者说dev分支代码稳定后可以合并到主分支master上来。