# Stage0学习报告

## Git学习

Git代码托管工具之前也接触过，但是没有花集中的时间来学习，现在集中时间来学习Git的相关知识，更清楚得认识到Git这个工具的特点。与之前熟悉的SVN相比，Git的分布式以及本地提交的特点更加方便了多人协作开发项目，每个人clone原工程，多分支开发，最后一起merge代码。

### 1.1基本操作

#### 1.1.1版本库的创建

Git是以仓库为单位来对代码进行托管的，首先是要在自己的工作区创建版本库，git init创建版本库，如果是使用远程仓库，则git clone远程仓库到自己的工作区。可使用git config对自己的Git进行相关的配置。

#### 1.1.2代码的提交

在版本库中将修改的代码写好后需要进行提交。Git add将需要提交的文件加入到提交状态，加入到暂存区，git commit完成代码的本地提交，这样完成了代码状态从工作区到暂存区的转换，但是代码还是在本地，这就是与SVN不同的地方，最后还需要git push将某个分支的代码提交到远程仓库，完成代码的提交。

#### 1.1.3代码回退

在项目中发现提交的代码不是最终所需的情况会经常出现，这是就需要对版本进行回退。这里就需要用到git reset –hard <commit id>来将版本回退到指定版本，commit id可以用git log来查看提交记录。如果是只add了代码，也能使用git reset将代码回退到工作区。

Git status来查看当前仓库的文件情况信息。

### 1.2远程仓库

这也是Git相对于SVN的一个不同点，每个人都可以从最初仓库fork到自己的仓库，然后git clone到自己的工作区进行开发，还可以将自己的开发过的版本修改提交到原始版本来对一个工程进行多人合作开发，这也方便了开源项目能够得到很多人的贡献。

### 1.3分支管理

分支管理能很方便的对一个项目进行开发，不同的分支处理不同的情况，另外不同的分支可以多人开发，最后合并分支。对分支的管理，git内部是通过指针实现的，每一分支都有对应的指针在跟随，这样分支的创建、合并和切换就很快。

使用git branch来查看当前分支以及仓库拥有的分支，git checkout来切换分支以及创建分支。

当然若在主分支上面的改动和子分支上面改动不一样，这时就会产生冲突，这时就需要明确各个分支的情况来手动解决冲突，使用git diff来查看冲突情况。

### 1.4标签管理

当一个版本完成后，需要使用标签来标识这个版本，就可以使用git tag来对这个版本打标签，使用git push同样是可以将标签提交到远程仓库。

## Docker学习

任务中虽然是使用Docker云服务，但为了更好的了解Docker，便去看了些Docker的知识，以及自己进行了相关的操作。

Docker是一种容器技术，使用容器对项目进行封装，一方面是方便部署管理，另一方面也能够快速根据当前的访问流量大小进行横向扩容。

Docker的组成分为三部分，包括Docker client、Docker Daemon、Docker Hub，在使用时，通过Docker Client来接收用户的命令，再由Daemon程序响应命令，从Docker Hub拉取镜像，在本地进行运行。

在自定义Docker镜像时，需要自己编写Dockerfile文件，Docker是分层进行封装的，这也体现在Dockerfile文件的编写中，在Dockerfile中应该尽可能的将同效果的命令放在一条执行语句中完成，这样Docker能根据Dockerfile中的命令更好的进行分层build Docker镜像，这样能使性能更好。DockerFile有一定的规则以及语法，根据不同的服务安装对应的软件以及做相关的配置，在容器中启动服务。

Docker运行在服务器中，相当于在一个操作系统上运行着一个虚拟机，但又不是全部的虚拟化，系统内核还是用得宿主机器的操作系统内核，Docker只是将上层应用进行了一个封装，独立出来了一个容器环境。这样我们就需要对容器中的数据对应到宿主机器上，以及将容器的开放端口对应到宿主端口上，等等这些操作是为了外界对Docker封装的服务的访问。

## MongoDB学习

在任务中关于数据库建议使用MongoDB，于是提前学习了关于MongoDB的知识。MongoDB是一个Nosql数据库，nosql数据库最大的特点是数据不需要结构化，这样更适合多样性数据的业务需求，并且做扩展也更方便。

### 3.1特点介绍

MongoDB的数据模型是基于文档的数据模型，也就是一个键值对的组合，相当于是一个JSON数据，Mongodb中的文档相当于是MySQL中表结构中的一行，多个文档就组成了一个集合，而这个集合就相当于是MySQL中的表。MongoDB中同一集合中的文档不需要结构相同，因此能很方便的插入不同的数据。

### 3.2数据类型

MongoDB对数据类型的支持还是比较全面的，除了在MySQL中常见的数据类型外，还支持正则表达式、数组、二进制数据和Javascript代码，这就是Mongodb在数据类型上面支持的优势。当然，在文档内部也能内嵌文档，整个文档看起来就是一个JSON数据。每一个文档都有一个唯一的ObjectId，这就是这个集合的唯一索引。

### 3.3数据操作

MongoDb的数据操作和Mysql还是很相似的，只是在语句上面换了一种形式，不再是SQL语句，而是JavaScript语句。

包括插入文档、删除文档、更新文档、查询文档等，db.col.insert()进行对数据的插入，在删除、更新、查询操作上，都需要进行匹配操作，这就相当于是Mysql中的where子句的作用，在Mongodb中使用了键值对的方式来构造这些语句，传入相关的函数中。

Mongodb中的聚合操作，包括$group,$project,$limit,$sort,$skip等等和MySQL中也是很相似的。

### 3.4索引

Mongodb中的索引有唯一索引、符合索引等，在索引的设计和使用上和Mysql类似，可以使用explain()函数来查看一个语句背后的执行计划，以及执行这条语句内部做了哪些操作等等，根据这些信息来查看一个语句的执行性能。

### 3.5 Security

在使用Mongodb的过程中，发现启动mongodb并没有对用户名密码进行验证便对数据库进行了操作，在Mongodb的官方文档中发现，mongodb默认是不需要用户名、密码，但是默认只绑定了本地IP，因此需要让外部访问Mongodb，需要对配置文件进行修改。

Mongodb的配置文件是YAML文件格式，在/etc/mongod.conf文件中进行了配置，注释掉bindIp，在security下配置authorization: enabled使MongoDB开通安全验证，重新启动MongoDB便能使配置生效。

然后需要在集合admin中对用户名和密码以及用户的权限进行配置。在进行配置时可知MongoDB对权限的分配做得很详细，用户拥有的数据库的权限、以及读写权限等等做得很详细，这些权限在添加用户时需要进行划分。后面也可以针对不同的情况对用户的权限进行修改。

## Shell学习

因为之前也对shell学习过，平时也是在用shell，对某些shell工具也有些熟悉，但是由于没有写过shell脚本文件，对shell脚本的一些语法还不够熟悉，便主要将学习重点放在了这个上面以及安排任务中强调的重点。

在看完知识重点后，便使用shell简单完成了后面任务重采集数据的agent.sh，提交到github上的repo了。

获取load average是通过读取/proc/loadavg文件进行的，此文件中包含了前一分钟，前五分钟，前15分钟内系统的平均负载情况。获取CPU信息是读取的/proc/stat文件，读取文件的第一行内容得到。关于内存信息是读取/proc/meminfo文件可以得到。

## Python学习

由于平时也是在使用python，对python的基础知识也有一定的了解，这次再次学习了《Python核心编程》这本书，也着重关注了任务书中强调的章节。再次看这本书也是学到了之前遗忘的内容，做了一次巩固学习。对列表、字典的一些内置方法加深了印象，更加熟悉了集合这个自己用得不多的数据结构，以及关于集合的一些数学操作。还有就是熟悉了一些功能函数，例如数值处理的abs(),pow(),round()等函数，还有oct(),hex()等进制处理函数，还有编码方面的函数ord(),chr()等函数。另外熟悉了pickle,datetime,subprocess等模块。

另外练习脚本提交到GitHub中的repo了，也是简单实现了后面要求的采集数据的agent。使用了getopt库对参数进行解析，然后使用了psutil获取的CPU和内存信息，因为psutil没有提供获取平均负载的接口，我便直接读取了/proc/loadavg文件。

## 开发规范学习

代码规范在实际开发中还是有很重要的地位的，好的规范不仅能使代码更高效，还能更安全，更方便维护。

在看Python开发规范中，印象比较深也是自己平时没有注意到的地方有如下几点：异常处理、默认迭代器和操作符、条件表达式、True/False求值、TODO注释。在异常处理中，要注意捕捉细小异常，而不是全部异常；尽量使用finally来处理一定会执行的语句，这样一个好处就是方便系统对资源的回收处理；捕捉异常时使用as，而不要使用逗号。在迭代器中，之前是没注意最好使用for来迭代可迭代对象，这样效率更高，而不是使用index来进行迭代；最好使用in 和not in来进行条件判断，这样也能更快，而不是使用”==”来对值进行判断。关于True/False求值，最好是直接if对值进行判断，而不是使用”==”来对值是否相等作为判断的条件。TODO注释是没有用过，这次了解到了对短期的解决方法，后面还会对此方案进行优化的地方使用TODO注释。

## 七、项目管理规范

Smart原则主要是将一个项目更加具体化了，量化了。Smart原则要求的是明确性、可衡量性、可达成性、相关性、时限性。这样对一个项目进行具体的设计规划，并根据各方面做出阶段性的完成任务，这样能对一个无形的项目在时间上以及人员分配上做出一个规划，能够将任务细化在时间上，能够使项目可控、可管理。这样不仅使人员分配更合理，还能是项目完成得更有效率。项目管理最终达到的目标就是项目的高质量和高效率。

## 八、总结

这次基础知识的学习学到了很多的知识，Docker和MongoDB是我之前一直想学但还没来得及学习的知识，这次去好好地了解了一些这方面的知识，终于感受到了为什么这两个技术现在能那么受欢迎，得到了那么多人的认可，因为这两者可以极大方便现在的运维自动化和数据多样化的开发需求，并且在性能上也不错，还能够在不影响服务性能的基础上降低成本。在这一阶段对这两者的学习是一个入门学习，后面对这的深入学习需要在项目中去学习，由项目为引导，去带动学习。

关于git的使用，现在也只是熟悉了git的基础知识以及常用场景下的命令操作，但是这涵盖不了所有的问题，需要在使用git的过程中针对不同的问题去解决，在使用过程中去不断熟悉git的使用。

关于python和shell语言，再次学习也收获了不少的知识，有些知识点一直没有用到就容易遗忘，这次是一次巩固基础知识的学习，站在不是一个初学者的角度去学习，能够得到不一样的收获，能够掌握更多的细节，使在写代码方面注重规范，写出充分利用语言特点的代码，便可以优化代码，增加性能。

这一阶段基础知识的学习基本上完成了，但是对这些知识的进一步掌握还要在后面的阶段进行巩固，只有深入到项目之中才能让基础知识得到发挥之地，才能对基础知识有进一步的认识。