7.23

配置jdk、tomcat、maven

1. 前期准备
2. 访问云平台开发环境安装文件：具体操作为：

* 找到hosts文件，地址为C:\windows\system32\drivers\etc
* 以记事本方式打开hosts文件，添加192.168.10.29 fileserv.localhost.com
* 浏览器访问fileserv.localhost.com

遭遇问题：hosts文件无法修改

问题原因：权限不够无法写入

解决方法：

* 修改hosts文档，另存为.txt格式保存到桌面
* 删除原路径下的hosts文件，将修改后的hosts.txt复制到该路径
* 删除.txt后缀

1. 安装notepad++
2. 安装jdk环境
3. 下载jdk1.8，安装到D:\Program Files\JAVA\JDK
4. 配置环境变量

* 添加变量 JAVA\_HOME，设置为D:\Program Files\JAVA\JDK
* 打开Path变量，增加记录 %JAVA\_HOME%\bin
* 添加变量 CLASSPATH， 设置为 %JAVA\_HOME%\lib

1. 检查环境是否安装成功

* 打开dos环境，输入java –version验证java版本
* 输入javac检验编译环境

1. 安装tomcat环境
2. 下载并解压安装包至D:\Program Files\JAVA\Tomcat
3. 配置环境变量

* 添加变量CATALINA\_HOME，设置为D:\Program Files\JAVA\Tomcat
* 打开Path变量，添加两条记录

%CATALINA\_HOME%\bin; %CATALINA\_HOME%\lib;

1. 检验tomcat环境

* 打开dos环境，输入service.bat install
* 输入startup.bat，启用tomcat服务
* 浏览器访问 localhost:8080

1. 修改管理员密码

* 访问D:\Program Files\JAVA\Tomcat\apache-tomcat-8.5.16\conf，以记事本格式打开 tomcat-users.xml
* 增加代码：

<role rolename="manager-gui"/>

<user username="admin" password="123456" roles="manager-gui"/>

* 浏览器访问localhost:8080，输入账号密码访问ManagerApp

1. 安装maven
2. 下载并解压安装包至D:\Program Files\JAVA\Maven
3. 配置Maven环境变量

* 添加环境变量MAVEN\_HOME，设置为D:\Program Files\JAVA\Maven
* 打开Path变量，添加记录 %MAVEN\_HOME%\bin

1. 检验Maven环境

* 打开dos环境，输入mvn -v

1. 配置本地仓库

* 访问D:\Program Files\JAVA\Maven，新建本地仓库文件夹maven-repository
* 访问D:\Program Files\JAVA\Maven\apache-maven-3.3.9\conf，以记事本格式打开settings.xml
* 增加一行代码 <localRepository>D:\Program Files\JAVA\Maven\maven-repository</localRepository>

1. 测试

* 打开dos，输入mvn help:system，可以看到本地仓库中下载了一些文件

7.24

安装idea，部署maven、tomcat

1. 安装并激活idea
2. 下载并安装idea至D:\Program Files\IntelliJ IDEA 2018.1.5
3. 访问<http://idea.lanyus.com/>，找到激活码
4. 配置JDK环境

* 在启动页面中打开Configure>Project Defaults>Project Structure
* 在Project SDK中配置JDK版本和路径

1. 部署maven
2. 打开 Intellij IDEA>File>Setting
3. 找到Build, Execution, Deployment > Build Tools >Maven
4. 配置maven home和local repository
5. 创建maven项目
   * File>New>Project,找到Maven项目
   * 勾选Create from archetype，找到maven-archetype-webapp
   * 设置GroupId和ArtifactId
   * 设置项目名和路径，完成
6. 部署tomcat
7. Run>EditConfigurations,点击左上角+号
8. 找到Tomcat Server，选择Local
9. 配置Application server，JRE，端口号
10. 在Deployment中添加ROOT文件
11. 部署完成启动Tomcat

遇到问题：启动Tomcat后，访问localhost:8080报错

问题原因：需要把Tomcat首页的工程部署到服务器

解决方法：

* Run>EditConfigurations>Deployment
* 点击右上角+号，选择External Source
* 将tomcat根目录下的webapps目录下的ROOT文件选中
* 完成部署

1. Mysql安装
2. 下载安装包并解压缩至D:\Program Files\Mysql\mysql-8.0.17-winx64
3. 创建并配置my.ini
4. 已管理员身份打开cmd，执行初始化流程

* 进入mysql的bin目录下
* 初始化数据库：mysqld –initialize –console,记录数据库初始密码
* 安装数据库：mysqld install
* 初始化data目录：mysqld –initialize –insecure
* 启动数据库：net start mysql

1. 登录Mysql，修改密码

* 输入登录指令：mysql –u root –p
* 输入初始密码，登录成功
* 修改初始密码：alter user user() identified by "123456";

1. Navicat12安装
2. 下载并解压安装包，运行安装文件安装至D:\Program Files\Navicat Premium 12
3. 解压破解补丁” version.dll”至安装目录下
4. 运行Navicat12，连接Mysql root

遭遇问题：

1. 安装包无法解压，解压文件一直损坏

问题原因：压缩包格式为.7z，winrar版本过低无法解压

解决方案：使用7-Zip解压工具解压成功

1. 连接Mysql失败，报错2059

问题原因：mysql8之前的版本加密规则为”mysql\_native\_password”，mysql8以后的加密规则为”cashing\_sha2\_password”，Navicat以”mysql\_native\_password”的方式无法访问，将Mysql的加密规则改为”cashing\_sha2\_password”即可解决。

解决方法：

* 打开cmd，登录Mysql
* 输入命令：ALTER USER 'root'@'localhost' IDENTIFIED BY 'password' PASSWORD EXPIRE NEVER;
* 输入命令：ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'password';
* 输入命令： ALTER USER 'root'@'localhost' IDENTIFIED BY '123456' PASSWORD EXPIRE NEVER;
* 输入命令：ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY '123456';

1. Git安装
2. 解压安装包并安装至D:\Program Files\Git
3. 初始化Git并配置仓库

7.25

1.安装Visio，Xmind，Typora，Adobe Reader

2.安装postman，wireshark，BeyondCompare,MQTTBox,Xshell

3.没有找到Snipaste安装包，使用FSCapture截图工具代替

4.MongoDB和RoboMongo安装loacalhost:27017

1. 下载MongoDB并安装至D:\Program Files\MongoDB\Server\4.0
2. 配置环境变量
3. 配置mongd.cfg文件
4. 启动MongoDB，以管理员身份打开cmd，输入net start mongodb
5. 浏览器输入<http://localhost:27017>，验证安装成功
6. 下载并安装Robomongo至D:\Program Files\Robomongo 1.0.0
7. 创建连接，连接地址localhost:27017
8. 连接成功

遭遇问题：cmd输入net start mongodb，提示发生系统错误5，拒绝访问

问题原因：权限不够指令无法执行

解决方案：以管理员身份启动cmd

5.Redis和RedisDesktop安装loacalhost:6379

1. [https://github.com/MSOpenTech/redis/releases](https://github.com/MSOpenTech/redis/releases" \t "_blank)下载windows版redis
2. 解压安装包到D:\Program Files\Redis
3. 添加环境变量 Path：D:\Program Files\Redis
4. 打开cmd，输入redis-server.exe redis.windows.conf启动redis
5. 新加一个cmd，输入redis-cli.exe，进入redis数据库
6. 将redis加入windows服务，打开cmd，输入redis-server --service-install redis.windows.conf
7. 在服务中，设置redis启动参数，-a 123456
8. 安装RedisDesktop到D:\Program Files\RedisDesktopManager
9. 启动并创建链接到localhost:6379
10. 连接成功

7.26

Git相关学习

**GitHub相关操作**

1. GitHub注册
2. 创建开源仓库
3. 点击右上角+号，选择new repository
4. 填写repository name 和description，选择初始化README文件
5. 点击Create Repository完成创建
6. 新建分支并提交修改
7. 点击branch标签，在Switch branches/tag中输入readme-edit
8. 点击Create branches:readne-edit选项
9. 点击分支下的README.md文件，点击右侧的编辑按钮
10. 增加一行记录This is my first repository on github
11. 填写Commit changes信息，点击Commit changes按钮提交修改
12. 推送合并请求
13. 点击Pull requests标签，点击new pull request按钮
14. base选择master分支，compare选择readme-edit分支
15. 点击create pull request按钮
16. 合并分支
17. 点击Merge pull request按钮
18. 点击Confirm merge按钮

配置Git

1. 获取SSH key
   1. 右键选择 git bash,
   2. 输入 ssh–keygen –t rsa –C [Eacol-Xie@yiqian.xie@neoway.com](mailto:Eacol-Xie@yiqian.xie@neoway.com)
   3. 在该目录下生成 .ssh 文件夹，在改文件夹中找到id\_rsa.pub文件
   4. 以txt格式打开该文件，复制其中的ssh key

注意：ssh-keygen之间没有空格

1. 设置远程仓库 SSH key
2. 登录github，在个人账户中找到setting
3. 点击SSH Keys选项卡，在右上角点击new SSH key选项
4. 输入 Title，将复制的ssh key粘贴到Key
5. 点击Add SSH key，完成
6. 验证配置是否成功
7. 在git bash环境下输入 ssh –T [git@github.com](mailto:git@github.com)
8. 如果显示continue信息，输入yes
9. 显示：You've successfully authenticated, but GitHub does not provide shell access
10. 连接成功

**相关名词**

Repository：远程仓库，部署在github上的个人仓库，仓库中存放项目文件

Branch：项目分支，不同分支上的操作互不影响，可以避免对msater主分支的误操作，通过merge指令可以将其他分支的合并到master分支

SSH：SSH协议，通过配置SSH密钥，从本地访问github时不再需要账号密码

**Git基本概念**

1. 在完成仓库初始化后，本地仓库可以分为工作区和版本库两大部分，其中，工作区是该仓库下可操作的文件目录，也就是可编写的项目文件，版本库是初始化后生成的隐藏的.git文件夹
2. 工作区，版本库，远程仓库三者之间的关系：工作区是本地主机上可修改的项目文件，在该区域可以对项目代码进行操作修改。远程仓库时部署在github上的项目文件。版本库是连接工作区和远程仓库的缓冲区。在没有git指令操作下三者之间互不影响。
3. 版本库下的几个重要文件：
   1. index：暂存区，记录当前分支的文件目录
   2. HEAD：分支游标，指向当前所处的分支
   3. objects：对象库，包含分支下的所有对象及内容
4. Git工作流程：
   1. 在工作区完成操作后，使用git add指令将工作区内容添加到index下的暂存区，使用git commit指令将暂存区的目录树按照HEAD指向的分支提交到objects对象库，使用git push指令将版本库中指定分支的项目文件推送到远程仓库

**Git操作**

1. git init:初始化仓库
   * git init:使用当前目录创建仓库
2. git clone:从Git仓库中拷贝项目，git仓库可以是本地仓库，也可以是远程仓库
   * git clone <repo>:复制本地仓库，或者将远程仓库复制的本地
   * git clone <repo> <directory>:复制本地仓库或远程仓库到指定目录

git clone 时，可以所用不同的协议，包括 ssh, git, https 等，其中最常用的是 ssh，因为速度较快，还可以配置公钥免输入密码。各种写法如下：

git clone git@github.com:Eacol-Xie /hello-world.git --SSH协议

git clone git://github.com/ Eacol-Xie /hello-world.git --GIT协议

git clone https://github.com/ Eacol-Xie /hello-world.git --HTTPS协议

1. git add:将工作区文件目录添加到暂存区

* git add <file>:向暂存区添加指定文件，可以有躲过<flie>参数
* git add .:向暂存区添加所有文件

1. git status:查看当前项目状态
   * git status:查看上次提交以后，当前项目文件的修改状态
   * git status –s:以简短格式输出结果。
2. git diff:对比文件的修改
   * git diff:显现尚未添加到暂存区的修改（工作区与暂存区对比）
   * git diff --cached:显示已添加到暂存区的修改（工作区与暂存区对比）
   * git diff HEAD:显示未提交的所有修改（与object分支文件对比）
   * git diff --stat:已概要模式显示修改信息
3. git commit:将缓存文件提交的对应的分支
   * git commit:将暂存区文件写入仓库中对应的分支，Git会打开一个vim以填写提交注释
   * git commit –m ‘message’:Git不会打开vim，而是将message作为提交注释
   * git commit –a:自动执行add流程，并执行commit流程
4. git reset HEAD:取消已提交的指定修改
   * git reser HEAD <file>:取消指定文件已提交的上一次指定修改
5. git rm:从以跟踪文件清单中删除文件
   * git rm <file>:从工作区和暂存区删除指定文件（未修改）
   * git rm –f <file>:强制删除文件，如果已修改并且已经add过，需要用该指令强制删除
   * git rm --cached <file>:只从暂存区删除文件，工作区文件保留
   * git rm –r \*:递归删除才做，会依次删除该目录下所有文件

rm指令只针对已跟踪的文件目录，即已经add到缓存区中的文件，对于index目录下没有记录的文件，会提示fatal: pathspec '\*\*\*' did not match any files

1. git mv:移动或重命名文件
   * git mv <file1> <file2>:将file1重命名为file2
2. git branch:分支管理指令

* git branch:列出仓库中的所有分支
* git branch <branch\_name>:创建branch\_name分支
* git branch –b <branch\_name>:删除branch\_name分支

1. git checkout:分支切换指令

* git checkout <branch\_name>:切换到branch\_name定分支
* git checkout –b <branch\_name>:创建branch\_name分支，并切换到该分支

1. git merge:分支合并指令

* git merge <branch\_name>:将branch\_name分支合并到当前分支

1. git log:查看提交日志

* git log:查看提交操作的日志（只能看到commit的操作记录）
* git log --oneline:简洁模式输出日志
* git log --graph:图示模式输出日志
* git log --decorate:显示提交关联的分支、标签等
* git log --reverse:反序输出日志
* git log --author=name:查看名为name的用户提交的日志
* git log --since={date1} --until={date2}:查看date1-date2之间的日志
* git log --before={date1} --after{date2}:查看date1之前，date2之后的日志

1. git tag:标签指令，版本提交快照

* git tag <tagname>:给当前最近依次提交添加一个标签
* git tag -a <tagname>:创建一个带注解的标签
* git tag -a <tagname> -m “message”:指定标签注解信息
* git tag -a <tagname> <commit>:给一个指定的commit添加标签
* git tag:显示所有标签

1. git remote:远程操作

* git remote:显示远程仓库
* git remote -v:实现short\_name对应得链接url
* git remote add <short\_name> <url>:添加url远程仓库并将其命名为short\_name
* git remote rm <short\_name>:删除指定的远程仓库

1. git fetch:从远程仓库下载项目文件

* git remote <short\_name>:从远程仓库下载文件

1. git merge:合并分支

* git merge <short\_name>/<branch>:将下载下来的文件合并到本地仓库的分支上

1. git push:推送指令

* git push <short\_name> <branch>:将本地的分支推送到远程仓库的分支

1. git pull:拉取指令

* git pull:拉取远程文件到本地，并自动合并分支

**几个指令的区别：**

* git commit：是将本地修改过的文件提交到本地库中；
* git push：是将本地库中的最新信息发送给远程库；
* git pull：是从远程获取最新版本到本地，并自动merge相当于fetch+merge；
* git fetch：是从远程获取最新版本到本地，不会自动merge；
* git merge：是用于从指定的commit(s)合并到当前分支，用来合并两个分支

7.27

7.29