BitterDonut网站项目

- 1. 安装Linux虚拟机

  1. 下载&安装 VirtualBox（Linux虚拟机），要开启 CPU 虚拟化（默认即开启）
  2. 下载&安装 Vagrant（Linux虚拟机配置工具），打开 window cmd 窗口，**运行 Vagrant init centos/7（官方提供的镜像），即可初始化一个 centos7 系统**
  3. 日常使用linux虚拟机，**只需cmd中输入：`vagrant up` 以及 `vagrant ssh` 即可连接上虚拟机**
  4. 在 VirtualBox配置文件中修改虚拟机ip地址

- 2. 在虚拟机上安装Docker

  1. 根据doker官网中的安装文档安装docker
  2. cmd启动docker：`sudo systemctl start docker`；配置docker自启动：`sudo systemctl enable docker`；
  重启docker服务：`sudo systemctl restart docker`
  3. 在阿里云中启动容器镜像加速
  （1）登录阿里云 -> 控制台 -> 产品与服务 -> 容器镜像服务 -> 镜像加速器 -> centOS -> 在vagrant中执行阿里云提供的镜像加速命令

- 在Docker中部署mysql服务器

  1. 命令行安装mysql：`docker pull mysql:5.7`

  2. 创建mysql实例并启用：
  （1）由于CentOS7中的安全模块selinux把权限禁掉了，无法使用-v命令进行挂载，所以需要我们事先将需要挂载的目录放到白名单中：

```
chcon -Rt svirt_sandbox_file_t /mydata/mysql/log
chcon -Rt svirt_sandbox_file_t /mydata/mysql/data
chcon -Rt svirt_sandbox_file_t /mydata/mysql/conf
```

  （2）然后执行挂载命令将docker中的mysql配置文件挂载到Linux中：

```
docker run -p 3306:3306 --name mysql \
-v /mydata/mysql/log:/var/log/mysql \
-v /mydata/mysql/data:/var/lib/mysql \
-v /mydata/mysql/conf:/etc/mysql \
-e MYSQL_ROOT_PASSWORD=root \
-d mysql:5.7
```

  该初始化语句将docker容器的**接口与linux虚拟机的接口映射**，并且**将容器内的mysql配置文件与虚拟机中的配置文件相映射。**这样修改mysql配置只需要在虚拟机中修改就好了。
  挂载失败的解决方法：https://blog.csdn.net/qq_41999034/article/details/106162366

  3. 在linux虚拟机中修改mysql容器的配置，将其编码方式改为utf-8（退出编辑 Fn键+ESC+:+wq）

  4. mysql日常启动方式：`docker start mysql`

- 在Docker中部署redis服务器

  1. 命令行安装：`docker pull redis`
  思维导图

2. 创建实例并启动

4. 在虚拟机中修改Redis配置文件，开启Redis的持久化功能
（1）将需要挂载的目录添加到白名单

```
chcon -Rt svirt_sandbox_file_t /mydata/redis/data
chcon -Rt svirt_sandbox_file_t /mydata/redis/conf
```

```
mkdir -p /mydata/redis/conf
touch /mydata/redis/conf/redis.conf
docker run -p 6379:6379 --name redis -v /mydata/redis/data:/data \
-v /mydata/redis/conf/redis.conf:/etc/redis/redis.conf \
-d redis redis-server /etc/redis/redis.conf
```

3. 使用 redis 镜像执行 redis-cli 命令连接

```
docker exec -it redis redis-cli
```

解释：docker exex -it命令可以在镜像中执行指令。指定redis镜像，在其中执行redis -cli命令（redis -cli是一个命令行客户端程序，可以将命令直接发送到**Redis**）

4. redis日常启动方式：`docker start redis`

5. 安装redis可视化软件redis desktop manager

- 构建项目框架

1. 构建五个微服务：common、chatRoom、user、resource、gateway（在初始化过程中使用依赖包含springweb与openfegin）
2. common：公共微服务，提供公共的依赖
3. chatRoom：聊天微服务
4. user：用户微服务
5. resource：资源微服务
6. gateway：网关微服务

在每个微服务的application.properties配置文件中配置微服务运行的端口号：`server.port=8000`

7. 将common微服务设置为公共微服务，将其他微服务添加到模组中：

```
<modelVersion>4.0.0</modelVersion>
<groupId>com.bitter</groupId>
<artifactId>common</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>common</name>
<description>聚合服务</description>
<packaging>pom</packaging>
<modules>
    <module>../user</module>
    <module>../chatRoom</module>
    <module>../resource</module>
    <module>../gateway</module>
</modules>
```

8. 在每个微服务中都对公共服务进行依赖：

```
  <dependency>
    <groupId>com.bitter</groupId>
```

```
    <artifactId>common</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</dependency>
```

- 将项目与github关联上

1. 在github上构建仓库
2. 下载git客户端
3. 在项目文件文件夹内上打开git bash
4. 执行git初始化命令

`git init`

5. 输入github注册的账号名称和邮箱

`git config --global user.name "ymyyh"` （GitHub相对应的帐号名称）

`git config --global user.email "865895777@qq.com"` （GitHbu相对应的邮箱帐号）

6. 在git命令行内生成ssh密钥

`cd ~/.ssh`

7. 自动复制ssh密钥

`clip < ~/.ssh/id_rsa.pub`

8. 连接github

打开GitHub 进入setting找到新建ssh key并输入之前复制的密钥

9. 建立github与git的连接

依次执行以下命令

`git init` 初始化git

`git add .` 将当前文件夹下所有文件添加到本地仓库

`git commit -m "提交描述"` 输入版本描述信息

`git remote add origin git@github.com:ymyyh/BitterDonut.git` 输入仓库地址建立连接，并将这个连接称为orgin

10. 将当前文件传输至github中的main分支

解释：由于github默认将所有上传的文件更新到master分支，如果先直接更新到main分支的话需要进行分支合并

`git checkout -b main` 切换分支到main

`git fetch origin main` 将当前连接orgin的分支main下载到本地（主要作用是下载github自动生成的readme文件，确保远程和本地仓库文件的一致性）

`git push origin -u main` 将本地仓库的orgin中的所有文件全部上传到github仓库的main分支

11. git从仓库将文件拉取到本地的命令：

（1） `git pull origin main` ：将仓库中origin连接的main分支文件拉取到本地

（2）如果发现拉取无效的情况，原因是本地的 `git add.` 缓存未清空，应使用以下方式执行：

`git reset HEAD` ：清空本地缓存

`git pull origin main` ：从仓库拉取文件

12. git将本地文件提交到github的命令：

`git init` 初始化git

`git add .` 将当前文件夹下所有文件添加到本地仓库

`git commit -m "提交描述"` 输入版本描述信息

`git push origin -u main` 将本地仓库的orgin中的所有文件全部上传到github仓库的main分支

- SpringBoot版本号问题说明

1. 由于SpringBoot的项目默认继承自父项目，父项目中已经实现了部分依赖的管理，这部分依赖引入的时候就不需要写版本号
2. 未被父项目管理的项目就需要写版本号
3. 通过 `<dependencyManagement></dependencyManagement>` 可以手动进行版本管理，如：

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-alibaba-dependencies</artifactId>
            <version>2.1.0.RELEASE</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

通过版本控制可以指定 `spring-cloud-alibaba-dependencies` 系列依赖的版本，在具体引用每个依赖时就不用一一指定版本了

- 解决依赖下载缓慢的问题

1. 右键点击pom文件，选择maven -> 创建setting.xml
2. 右键点击pom文件，选择maven -> 打开setting.xml
3. 在设置文件中配置阿里巴巴下载镜像：

```
<mirrors>、
    <mirror>
        <id>alimaven</id>
        <name>aliyun maven</name>
        <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
        <mirrorOf>central</mirrorOf>
    </mirror>
</mirrors>
```

- 设置配置中心/注册中心

1. 进入github，spring cloud alibaba中文文档，查看nacos中间件的使用说明

2. 引入Nacos服务注册发现依赖

```
<dependencies>
    <dependency>
        <groupId>com.alibaba.cloud</groupId>
        <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
    </dependency>
</dependencies>
```

3. 下载并运行Nacos服务器
（1）在github上下载nacos服务器的压缩包，解压之后运行nacos/bin/startup文件（在此电脑上在gulimallproject中）
（2）nacos服务器运行之后在控制台可见一个ip地址和端口号，即为服务的入口（一般为本机的8848端口）

4. 给application.properties文件中配置微服务注册发现的地址，并且为每个微服务设置在注册中心中的名字

```
spring.cloud.nacos.discovery.server-addr=127.0.0.1:8848
spring.application.name=chatRoom
```

5. 在各微服务的主类的 `@SpringBootApplication` 注解的上方添加 `@EnableDiscoveryClient` 注解，开启注册发现功能

- SpringBoot使用中报错解决

（1）java: 错误: 不支持发行版本 5

1）问题成因：SpringBoot项目中，总项目的jdk版本，各模块的jdk版本，与电脑安装jdk版本匹配不上

2）解决方法1：在pom中手动指定jdk版本：

```
<properties>
    <maven.compiler.source>13</maven.compiler.source>
    <maven.compiler.target>13</maven.compiler.target>
</properties>
```

3）解决方法2：

a. 文件 --> 项目结构 --> 项目/模块

指定jdk版本，使得所有模块与项目的jdk版本保持一致

b. 在文件 --> 设置 -->构建、执行、部署 -->编译器 -->java编译器中将编译器版本调整为与项目jdk版本一致

（2）端口号已占用的解决方式：

控制台输入命令：

`netstat -ano | findstr 8848`：查询占用8848端口号的进程id

`taskkill -f -t /pid 2668`：根据查询到的pid杀死占用端口号的进程

以上内容整理于 幕布文档