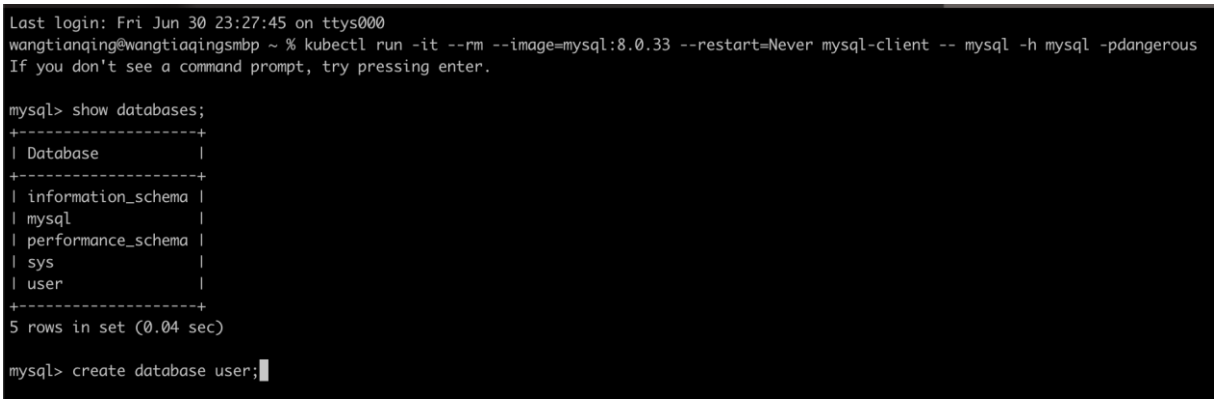
基于云原生技术的软件开发 - 第二次作业

作业说明

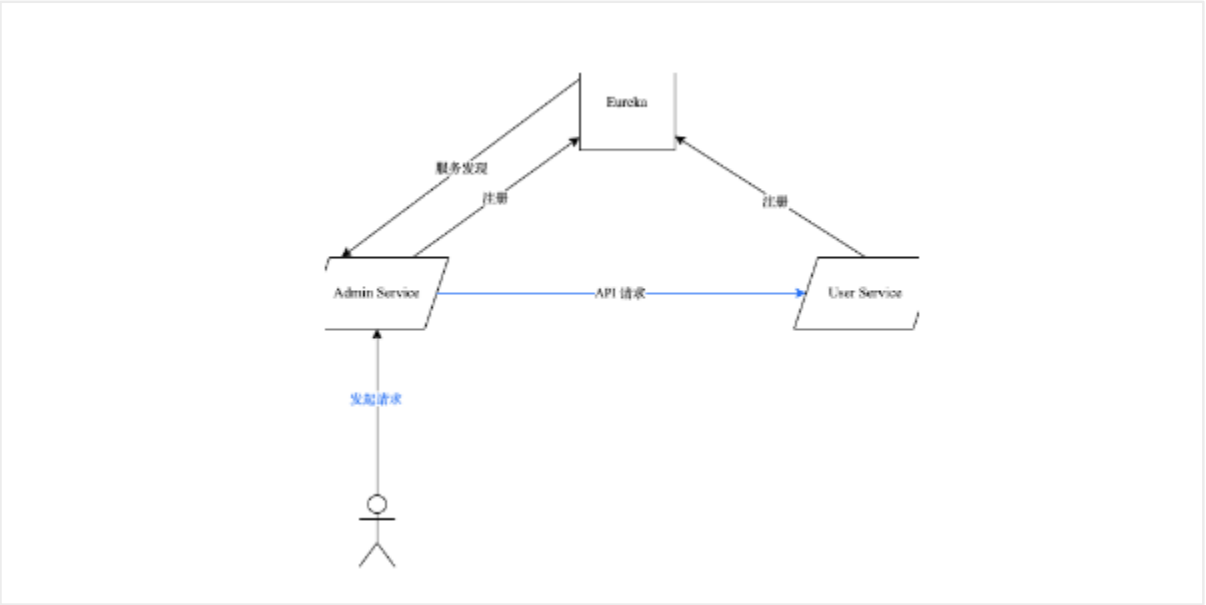
使用 Spring Boot 开发两个微服务：(Admin Service 和 User Service)：

1. 两个微服务启动时都注册到 Eureka 服务；
2. Admin Service 提供一个「**增加用户**」的 API：对 API 中的用户名和密码要做不为空校验，然后通过服务发现调用 User Service 完成增加用户的操



作（接口设计遵循 RESTful 风格）；

3. User Service 也提供增加用户的 API：完成对数据库中「**用户表**」的操作；
4. 为每个项目编写 Dockerfile，并通过 docker build 构建 Admin 服务，User 服务以及 Eurkea 服务镜像；
5. 编写 yaml，使用 Kubernetes 运行 Admin，User 和 Eureka 服务；



项目要求

项目基本要求是覆盖以上说明的需求，功能需求完成「**增加用户**」的需求以及在 Kubernetes 上运行

分数说明

本次作业占总评 35 分，分数分配如下

1. 基本功能点实现以及加入 Eureka 组件（10 分）
2. 编写 Dockerfile 并成功构建成镜像（5 分）- 如果使用多阶段构建，可以加分
3. 编写 YAML 文件，在 Kubernetes 上运行三个服务，且【增加用户功能】可以正常工作（15 分）
    a. Admin Service：一个 Deployment（replica 为 1）+ 一个 Service（nodeport）
    b. Eureka Service：一个 Deployment（replica 为 1）+ 一个 Service（nodeport）
    c. User Server：一个 Deploymnent + 一个 Service （nodeport）
4. **加入负载均衡功能（使用 Spring Cloud 的方案）** - 上述负载均衡指根据 Spring

Cloud 的组件或自己代码实现，可以让 Admin Service 的流量打到多个 User Service 上，负载均衡策略自定 （5 分）

提交要求

提交一份项目说明文档，必须包含：

1. 项目模块说明
2. 执行结果截图（比如这次作业，必须的就是正常创建成功的截图以及用户名或密码为空的创建失败的截图，如果有其它功能，需添加相应截图）
3. 关键代码说明（包括 Dockerfile，YAML 文件，Docker 以及 Kubectl 命令）

文档内容不限于以上所述。

注意：因为只会根据文档评分，文档一定要完整准确清晰地体现所做的工作，请大家对自己负责！

提交方式

发邮件到：191250192@smail.nju.edu.cn，邮件标题为：学号 + 姓名 + 第二次作业

代码参考

- https://start.spring.io/
- Getting Started | Building an Application with Spring Boot
- https://spring.io/guides
- Introduction to Spring Cloud Netflix - Eureka | Baeldung
- tutorials/spring-cloud-modules/spring-cloud-eureka at master · eugenp/tutorials · GitHub

Java 代码构建参考

- Maven settings.xml 设置：将 maven 源改为国内阿里云镜像 - 知乎

```
mv ~/.m2/settings.xml ~/.m2/settings.xml.bak
vim ~/.m2/settings.xml
```

settings.xml

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
                      http://maven.apache.org/xsd/settings-
1.0.0.xsd">
  <localRepository/>
  <interactiveMode/>
  <usePluginRegistry/>
  <offline/>
  <pluginGroups/>
  <servers/>
  <mirrors>
    <mirror>
     <id>aliyunmaven</id>
     <mirrorOf>central</mirrorOf>
     <name>阿里云公共仓库</name>
     <url>https://maven.aliyun.com/repository/central</url>
    </mirror>
    <mirror>
      <id>repo1</id>
      <mirrorOf>central</mirrorOf>
      <name>central repo</name>
      <url>http://repo1.maven.org/maven2/</url>
    </mirror>
    <mirror>
     <id>aliyunmaven</id>
     <mirrorOf>apache snapshots</mirrorOf>
     <name>阿里云阿帕奇仓库</name>
     <url>https://maven.aliyun.com/repository/apache-
snapshots</url>
    </mirror>
  </mirrors>
  <proxies/>
  <activeProfiles/>
  <profiles>
    <profile>
        <repositories>
           <repository>
                <id>aliyunmaven</id>
                <name>aliyunmaven</name>

<url>https://maven.aliyun.com/repository/public</url>
                <layout>default</layout>
```

- 本地构建：mvn -B -Dmaven.test.skip clean package
- 容器化构建

```
docker run -it --rm -v "$PWD":/usr/src/mymaven -v
"$HOME/.m2":/root/.m2 -w /usr/src/mymaven maven:3.8.7-eclipse-
temurin-8 mvn -B -Dmaven.test.skip clean package
```

- 多阶段构建：在 Dockerfile 中使用多阶段构建打包 Java 应用

MySQL 运行参考

- 运行一个单实例有状态应用 | Kubernetes
- 运行服务端
  - mysql-deploymennt.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  ports:
  - port: 3306
  selector:
    app: mysql
  clusterIP: None
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - image: mysql:8.0.33
        name: mysql
        env:
          # 在实际中使用 secret
        - name: MYSQL_ROOT_PASSWORD
          value: dangerous
        ports:
        - containerPort: 3306
          name: mysql
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pv-claim
```

- mysql-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-pv-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 100Mi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/Users/wangtianqing/data"
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
```

- 运行客户端：

```
kubectl run -it --rm --image=mysql:8.0.33 --restart=Never mysql-client -- mysql -h mysql -pdangerous
```

- 创建数据库

必须要创建数据库，否则程序运行会报错

```
create database user;
```

功能验证

- Post 请求