**华南农业大学信息学院**

设计性、综合性实验

|  |  |  |  |
| --- | --- | --- | --- |
| **实验题目** | **决战Linux** | **学期** | **2022-2023-2** |
| **课程名称** | Linux系统及程序设计 | | |
| **学生信息** | 学号 学生姓名  院系 专业班级 | | |
| **实验内容及要求** | **总体要求：**  利用docker/docker-comose工具完成至少3种不同动态语言（PHP/Java/Python/Go/Perl/Nodejs...）运行环境的搭建，并实现从mysql/mongodb等数据库中读取数据到Web页面中显示。  **具体要求：**  1. 实验必须在云服务器上完成（阿里云、腾讯云、华为云等），不能在本机或虚拟机上完成。可以远程访问部署好的服务，并保持至少2周(截止日期后算起2周)，否则按抄袭零分处理；[可以现在本地弄好，再传上去云服务器运行]  2. 文档一开始，必须有服务器的外网IP截图，例如：执行curl ip.sb 。如果没有按抄袭零分处理；  3. 所有浏览器页面的运行效果截图必须带上地址栏信息，并且和curl ip.sb截图中的IP信息一致，否则按抄袭零分处理；  4. 从数据库中读取的内容必须包含自己的姓名和学号信息，否则按抄袭零分处理；  5. 每一个环境镜像都只能通过dockerfile的方式来构建，基础镜像只能使用centos或者ubuntu镜像，不能使用其他现成的环境。可以把所有服务部署在同一个镜像中。  6. 各种语言的运行代码没作要求，可以从网上找可用的代码，但最后的实现效果需要符合实验的要求(第4点)。  **提交要求：**  提交综合实验报告 + 源代码  实验报告文档命名为: 学号.docx  源代码/dockerfile/docker-compose.yml等统一打包命名为：学号.rar  源代码按分类设置子目录，如：php、java、python  文档和压缩包分开上传 | | |
| **-评语**  **成绩** |  | | |

1.1实验环境描述

系统发行版信息：

指令：lsb\_release -a

LSB Version: :core-4.1-amd64:core-4.1-noarch

Distributor ID: CentOS

Description: CentOS Linux release 7.9.2009 (Core)

Release: 7.9.2009

Codename: Core

系统IP信息(附上截图)：

IP：8.139.5.39



Mysql:Ver 15.1 Distrib 5.5.68-MariaDB

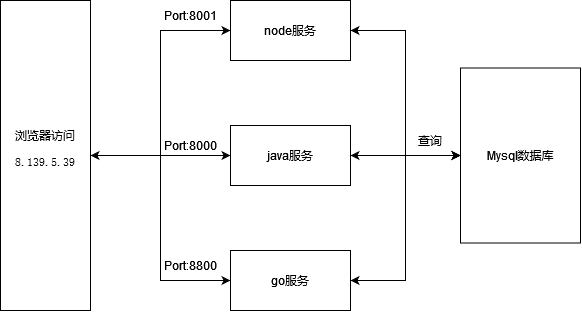
Docker:version 1.13.1

golang:1.20.3

node:14

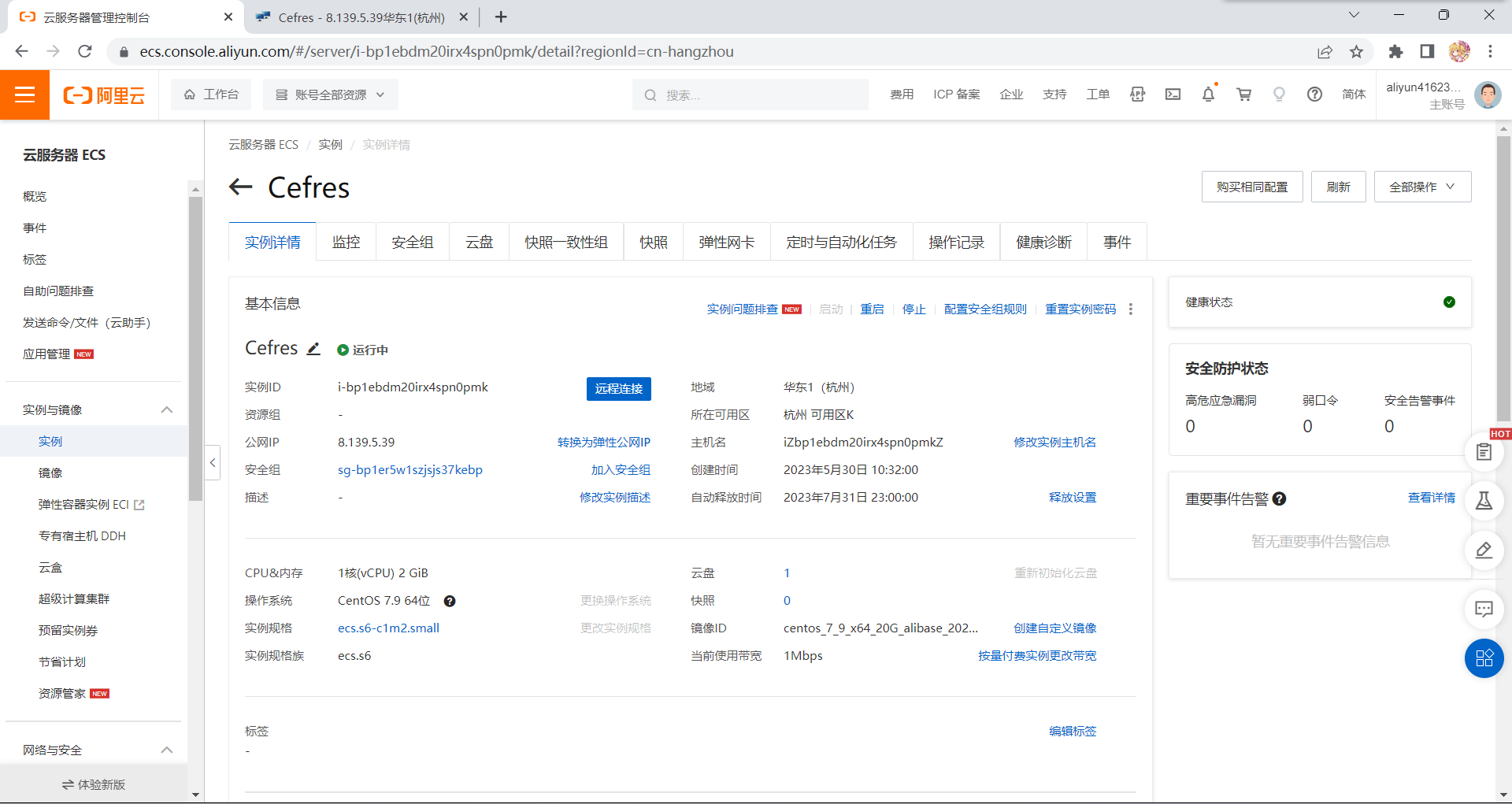
jdk:17.0

1.2系统架构图，系统交互图

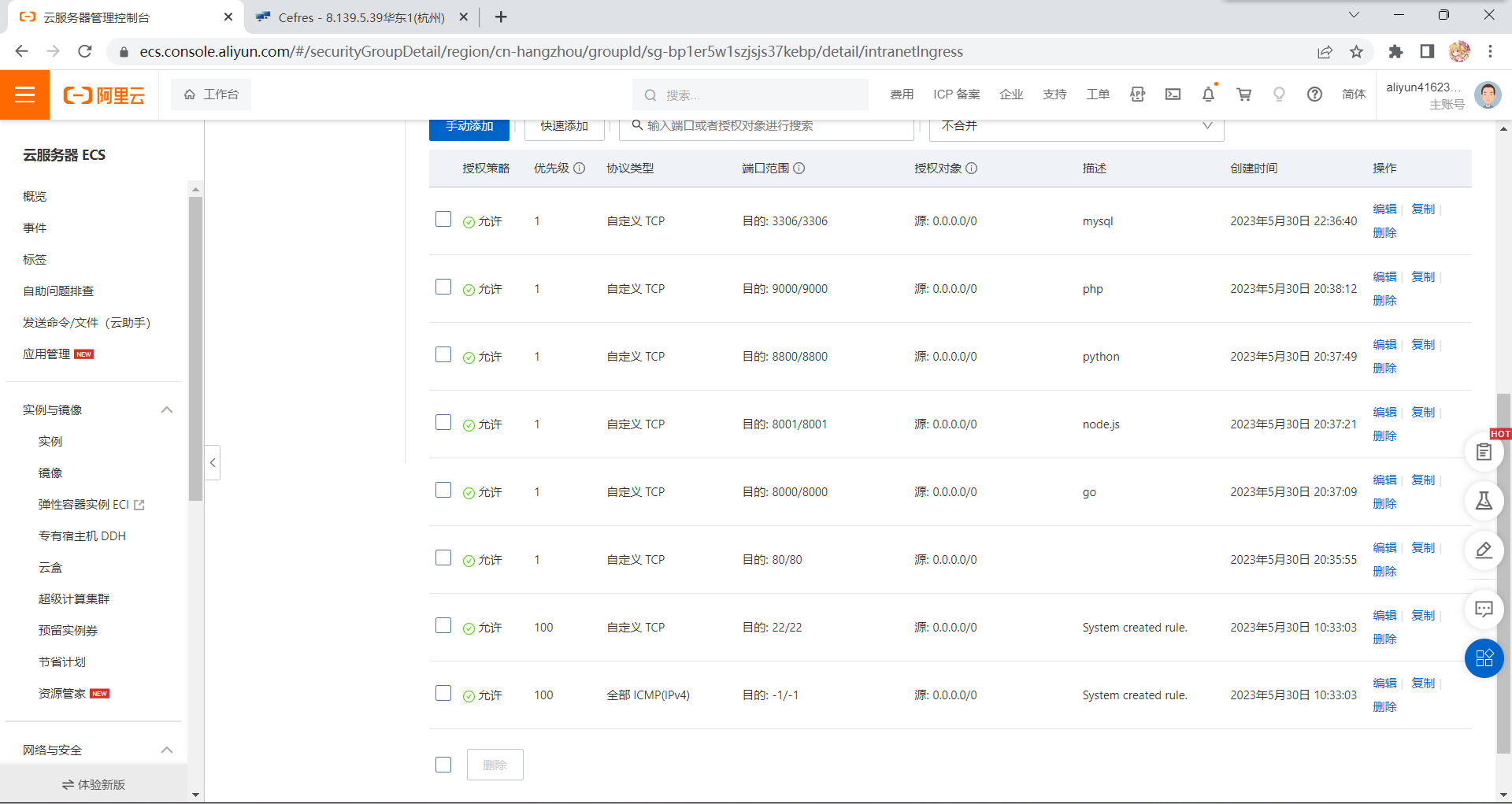


1.3环境搭建过程

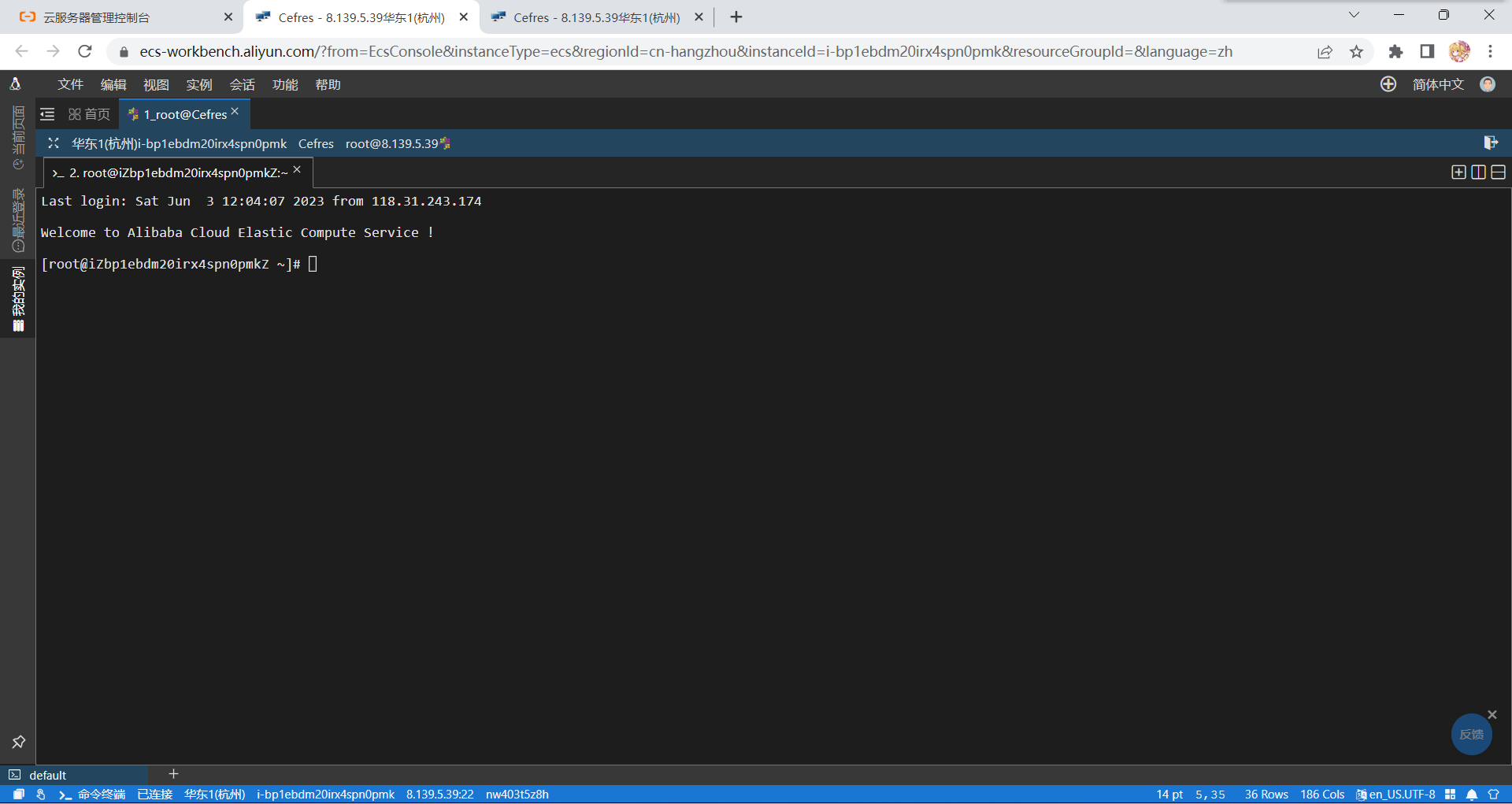
1.试用服务器，进入管理控制台



2.点击安全组，为当前实例添加开放端口



3.使用管理平台自带的ssh连接实例



4.安装必要的软件

yum install docker

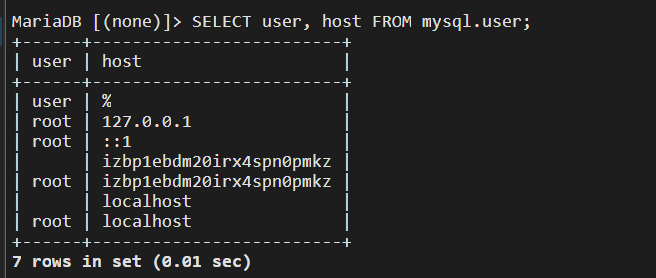
yum install npm

yum install nodejs

yum install mysql

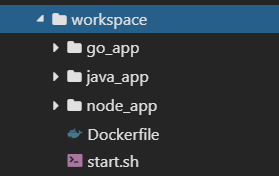
yum install go

5.设置mysql，建表，插数据，开放访问

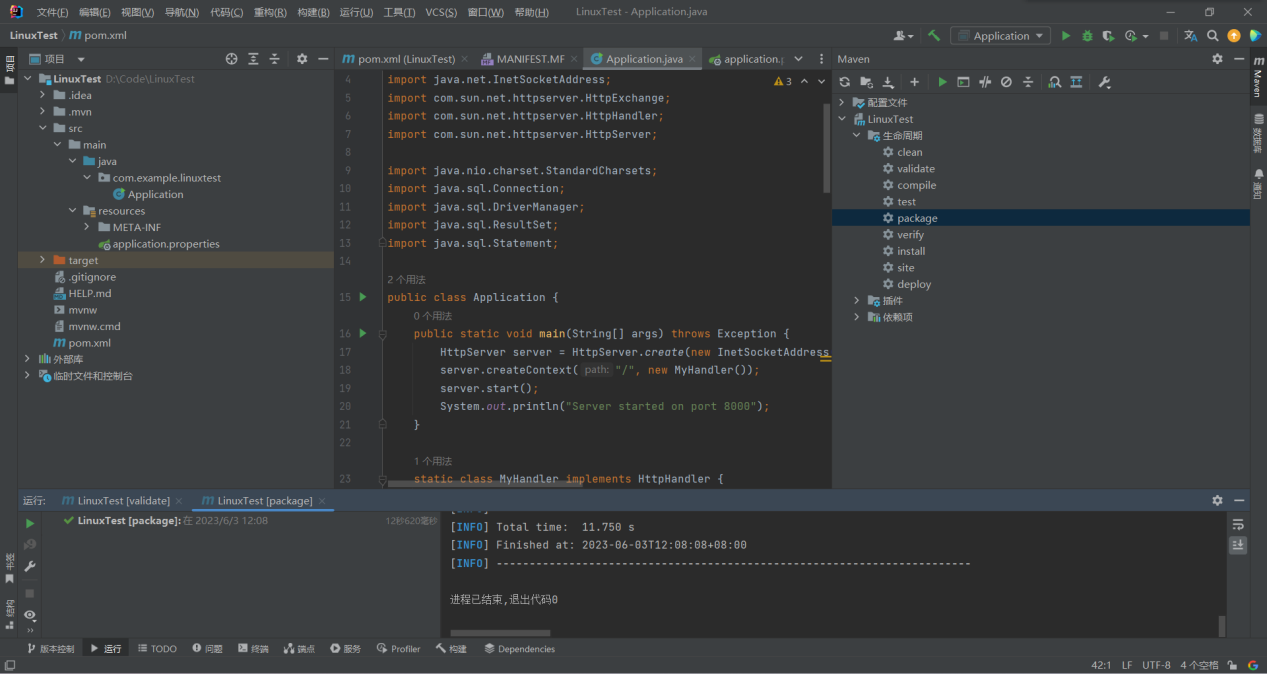


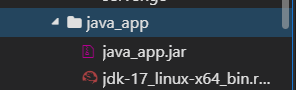
指令见关键代码

6.创建工作目录



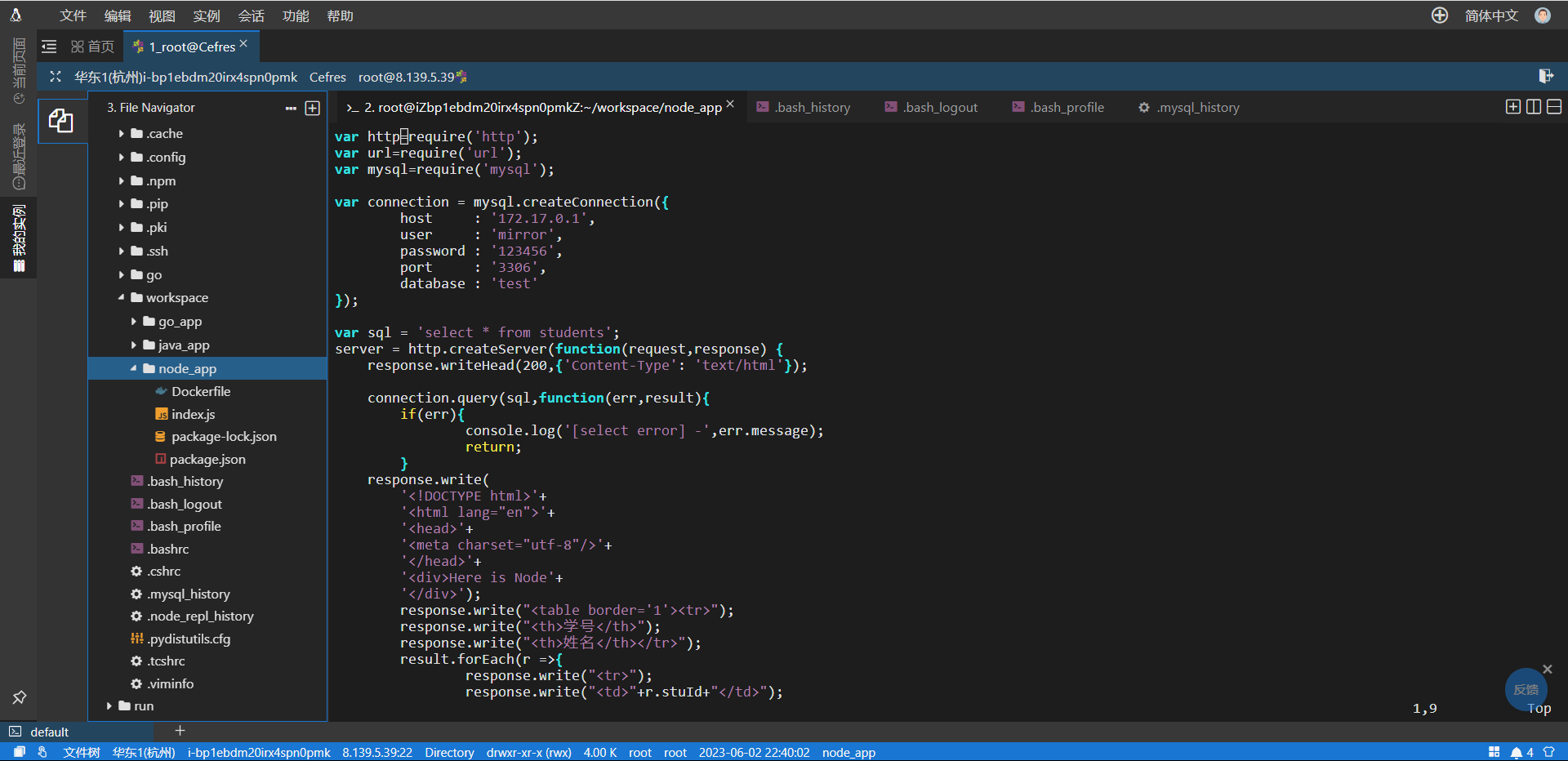
7.写好java服务端代码，打包上传到实例

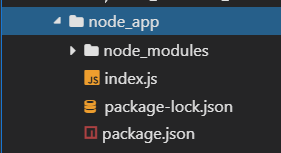




注：由于本人下载jdk17遇到困难，所有这里的jdk-17\_linux-x64\_bin.rpm为oracle官网下载的安装包手动安装jdk17.

8.在node工作文件夹编辑js文件



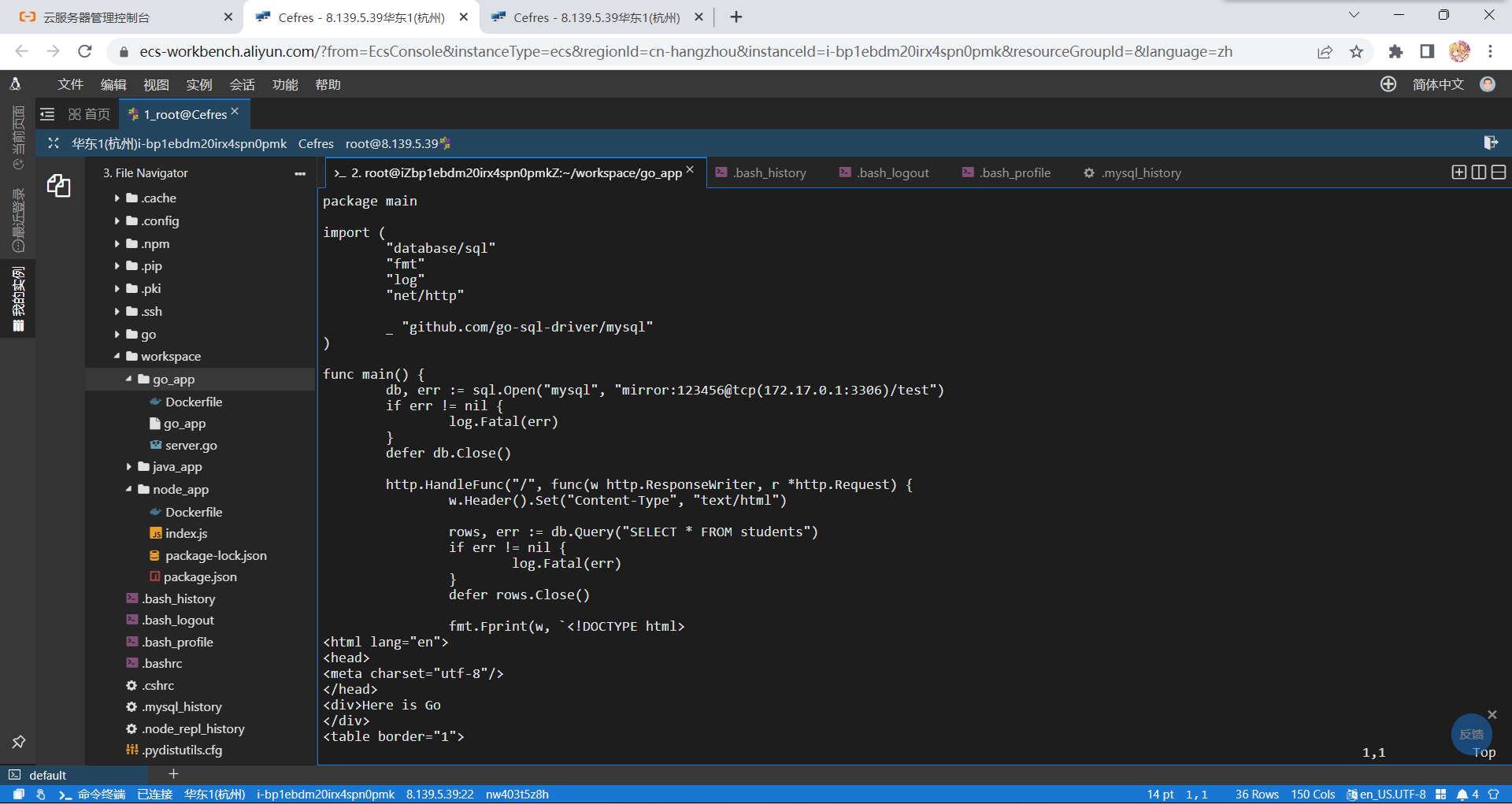


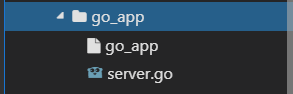
需要依赖，在工作文件夹下：

npm init

npm install mysql

9.在go工作文件夹中编辑server.go文件，修改下载镜像安装依赖后编译，容器将直接运行编译后的文件。





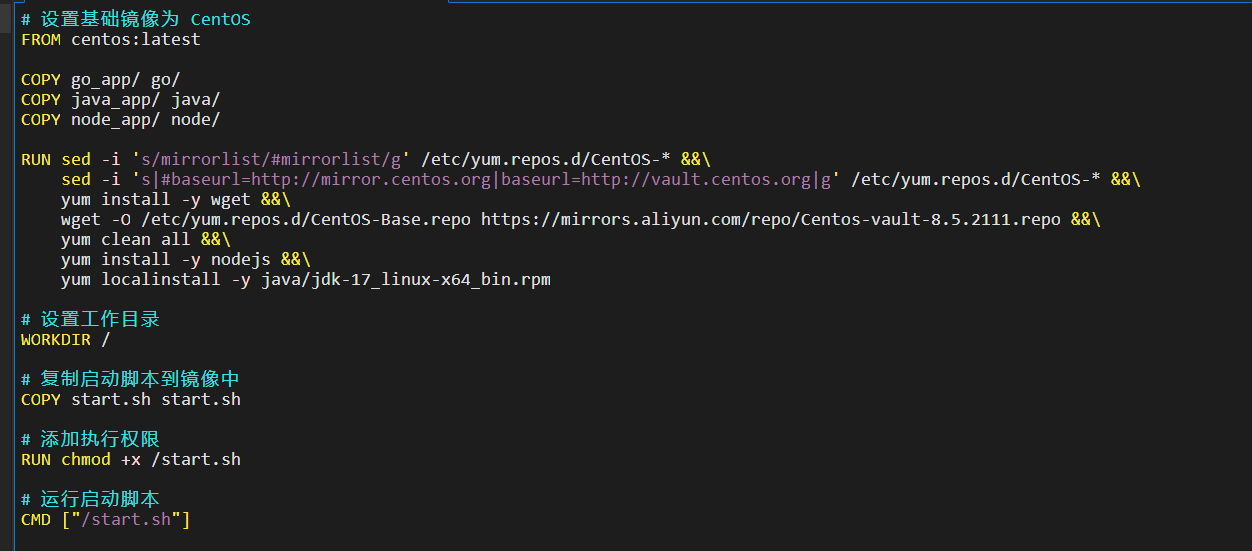
在工作文件夹下：

go env -w GOPROXY=https://mirrors.aliyun.com/goproxy/

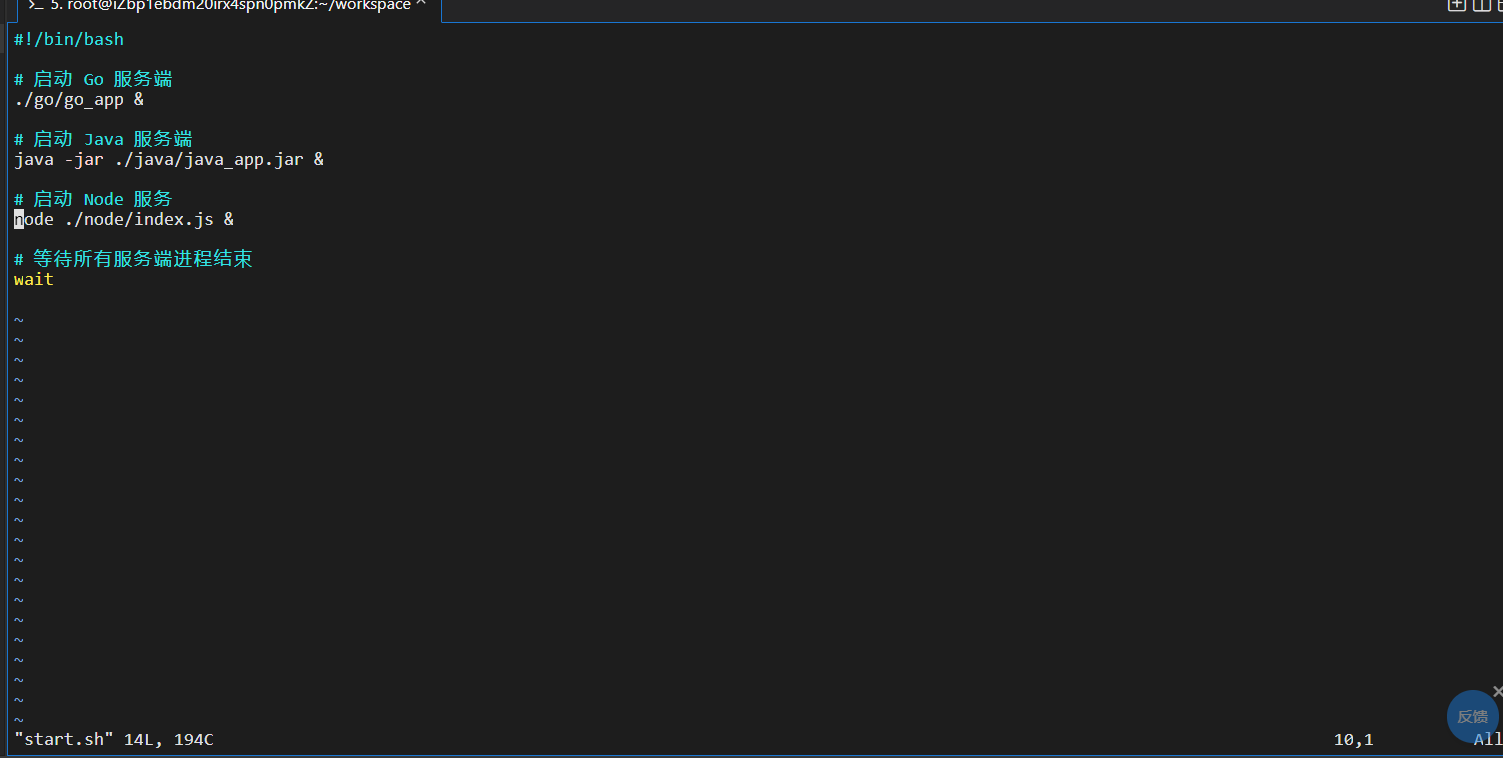
go get github.com/go-sql-driver/mysql

go build

10.编写Dockerfile文件



11.编写.sh文件



12.镜像搭建运行

//在工作目录下

docker build -t server .

//运行服务

docker run -p 8000:8000 -p 8001:8001 -p 8800:8800 -d server

//部分有帮助的指令

docker rm -f $(docker ps -aq) //删除所有container

docker rmi -f $(docker images -aq) //删除所有image

docker container prune //删除所有已停止container

docker image prune //删除所有已停止image

1.4核心代码展示

**Mysql:**

CREATE TABLE students(

stuId VARCHAR(12) NOT NULL,

stuName VARCHAR(50) NOT NULL,

PRIMARY KEY (stuId)

);

GRANT ALL PRIVILEGES ON \*.\* TO 'user'@'%' IDENTIFIED BY '123456';

//test

show tables;

SELECT \* FROM students;

SELECT user, host FROM mysql.user;

**Java：**

public static void main(String[] args) throws Exception {

HttpServer server = HttpServer.create(new InetSocketAddress(8000), 0);

server.createContext("/", new MyHandler());

server.start();

System.out.println("Server started on port 8000");

// 创建数据库连接

connection = DriverManager.getConnection("jdbc:mysql://8.139.5.39:3306/test", "user", "123456");

if (connection != null) {

System.out.println("Database connection successful!");

} else {

System.out.println("Failed to connect to the database!");

}

// 定时发送心跳查询

Timer timer = new Timer();

timer.schedule(new TimerTask() {

public void run() {

try {

Statement stmt = connection.createStatement();

stmt.execute("SELECT 1");

System.out.println("Heartbeat query executed.");

stmt.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}, 0, 3600000); // 每一小时执行一次查询

}

**Node.js:**

server = http.createServer(function(request,response) {

response.writeHead(200,{'Content-Type': 'text/html'});

connection.query(sql,function(err,result){

if(err){

console.log('[select error] -',err.message);

return;

}

response.write(

'<!DOCTYPE html>'+

'<html lang="en">'+

'<head>'+

'<meta charset="utf-8"/>'+

'</head>'+

'<div>Here is Node'+

'</div>');

response.write("<table border='1'><tr>");

response.write("<th>学号</th>");

response.write("<th>姓名</th></tr>");

result.forEach(r =>{

response.write("<tr>");

response.write("<td>"+r.stuId+"</td>");

response.write("<td>"+r.stuName+"</td>");

response.write("</tr>");

})

response.end();

});

});

server.listen(8001);

**Go:**

db, err := sql.Open("mysql", "mirror:123456@tcp(172.17.0.1:3306)/test")

if err != nil {

log.Fatal(err)

}

defer db.Close()

http.HandleFunc("/", func(w http.ResponseWriter, r \*http.Request) {

w.Header().Set("Content-Type", "text/html")

rows, err := db.Query("SELECT \* FROM students")

if err != nil {

log.Fatal(err)

}

defer rows.Close()

fmt.Fprint(w, `<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8"/>

</head>

<div>Here is Go

</div>

<table border="1">

<tr>

<th>学号</th>

<th>姓名</th>

</tr>`)

for rows.Next() {

var stuId, stuName string

err := rows.Scan(&stuId, &stuName)

if err != nil {log.Fatal(err)}

fmt.Fprintf(w, "<tr><td>%s</td><td>%s</td></tr>", stuId, stuName)}

fmt.Fprint(w, "</table></html>")})

log.Fatal(http.ListenAndServe(":8800", nil))}

**Dockerfile:**

# 设置基础镜像为 CentOS

FROM centos:latest

COPY go\_app/ go/

COPY java\_app/ java/

COPY node\_app/ node/

RUN sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-\* &&\

sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g' /etc/yum.repos.d/CentOS-\* &&\

yum install -y wget &&\

wget -O /etc/yum.repos.d/CentOS-Base.repo https://mirrors.aliyun.com/repo/Centos-vault-8.5.2111.repo &&\

yum clean all &&\

yum install -y nodejs &&\

yum localinstall -y java/jdk-17\_linux-x64\_bin.rpm

# 设置工作目录

WORKDIR /

# 复制启动脚本到镜像中

COPY start.sh start.sh

# 添加执行权限

RUN chmod +x /start.sh

# 运行启动脚本

CMD ["/start.sh"]

**start.sh:**

#!/bin/bash

# 启动 Go 服务端

./go/go\_app &

# 启动 Java 服务端

java -jar ./java/java\_app.jar &

# 启动 Node 服务

node ./node/index.js &

# 等待所有服务端进程结束

wait

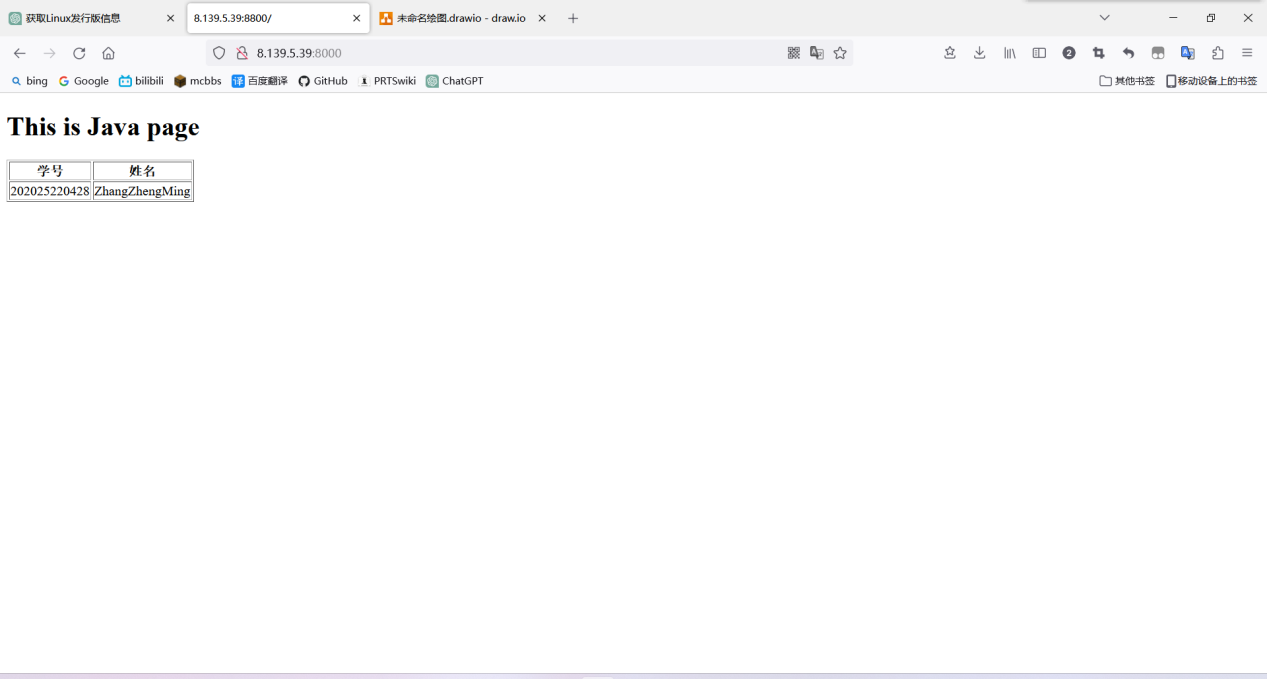
1.5测试效果

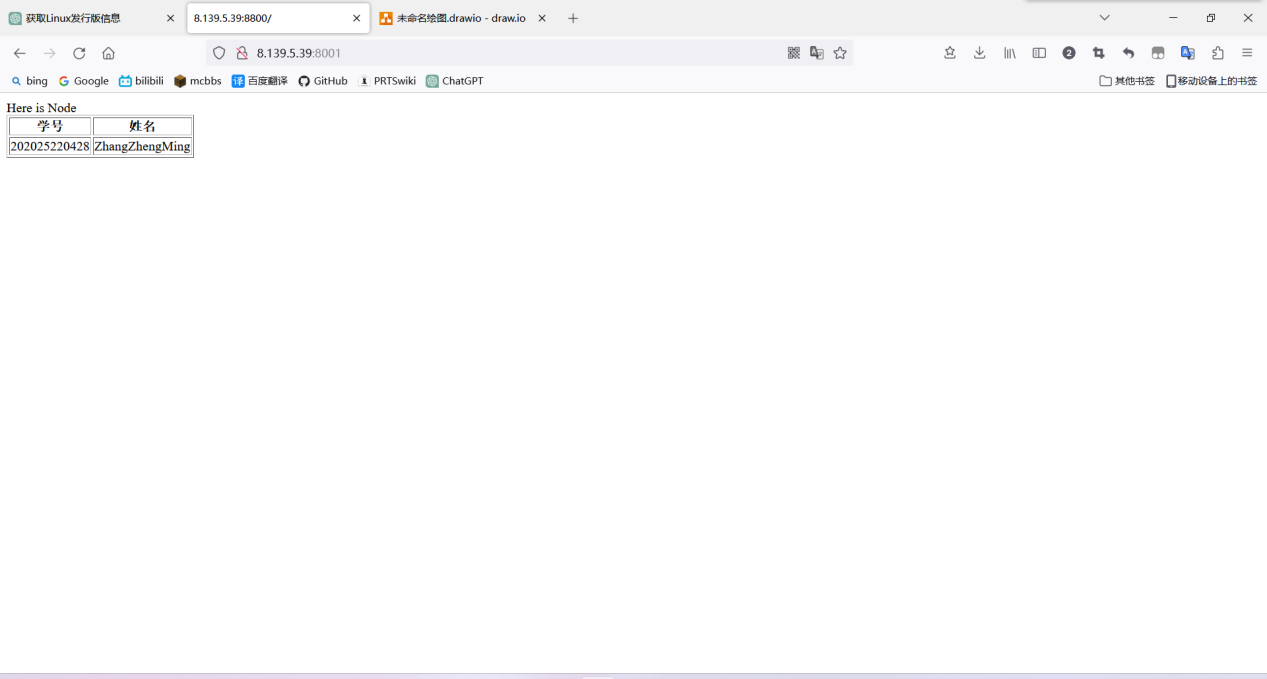
ip:

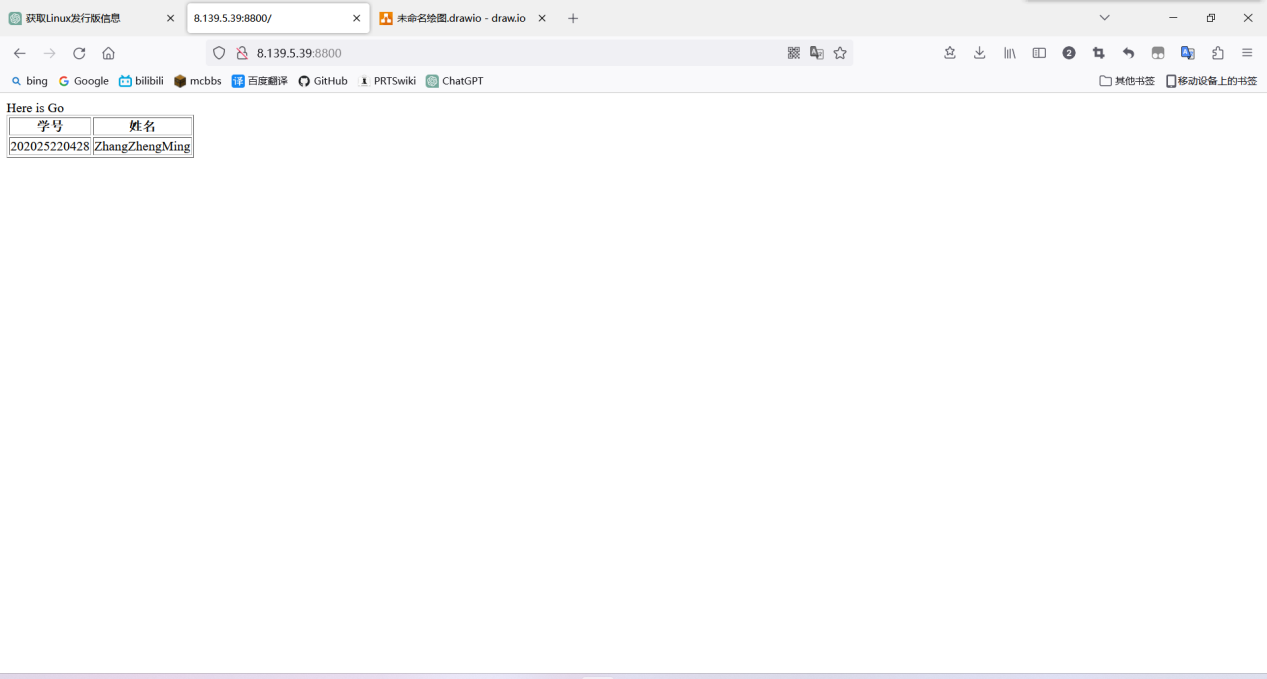
java:http://8.139.5.39:8000/

node.js:http://8.139.5.39:8001/

Go:http://8.139.5.39:8800/







1.6.实验总结

在完成本次实验中，我利用Docker工具搭建了Java、Go和Node.js的运行环境，并实现了从MySQL数据库读取数据并在Web页面中显示的功能。

在整个实验过程中，我对Docker工具的使用有了更深入的了解。通过使用Docker，我能够轻松地创建和管理不同语言的运行环境，并且能够快速部署应用程序。使用Docker镜像的方式，我可以确保应用程序在不同环境中的一致性，并且可以方便地进行扩展和升级。

在编写Dockerfile时，我发现了一些问题。首先，我需要确保在Dockerfile中指定了正确的基础镜像。不同语言的应用程序可能需要不同的基础镜像，所以要仔细选择适合自己应用程序的镜像。其次，我需要确保在Dockerfile中正确地设置工作目录、复制应用程序文件以及安装依赖项等步骤，以确保应用程序能够正确运行。

在连接MySQL数据库并从中读取数据的过程中，我也遇到了一些问题。首先，我需要确保在应用程序中正确配置数据库连接信息，包括主机地址、端口、用户名和密码等，需要使用Grant指令来授权数据库访问。与此同时，要让镜像内的程序访问宿主机的mysql还需要指定特定的ip而不是localhost。

总的来说，这个实验对我来说是一次有意义的学习和实践机会。我通过使用Docker搭建了多个语言的运行环境，并成功实现了从MySQL数据库读取数据并在Web页面中展示的功能。这个实验不仅加深了我对Docker的理解，还提升了我在多语言开发和数据库连接方面的技能。我相信这些经验将对我的未来学习和工作有很大的帮助。