sudo docker run -d -p 80:80 -p 443:443 --name nginx --restart=always -e TZ="Asia/Shanghai" -v /home/opc/nginx/html:/usr/share/nginx/html:ro -v /home/opc/nginx/conf:/etc/nginx/conf.d:ro -v /home/opc/nginx/log:/var/log/nginx nginx

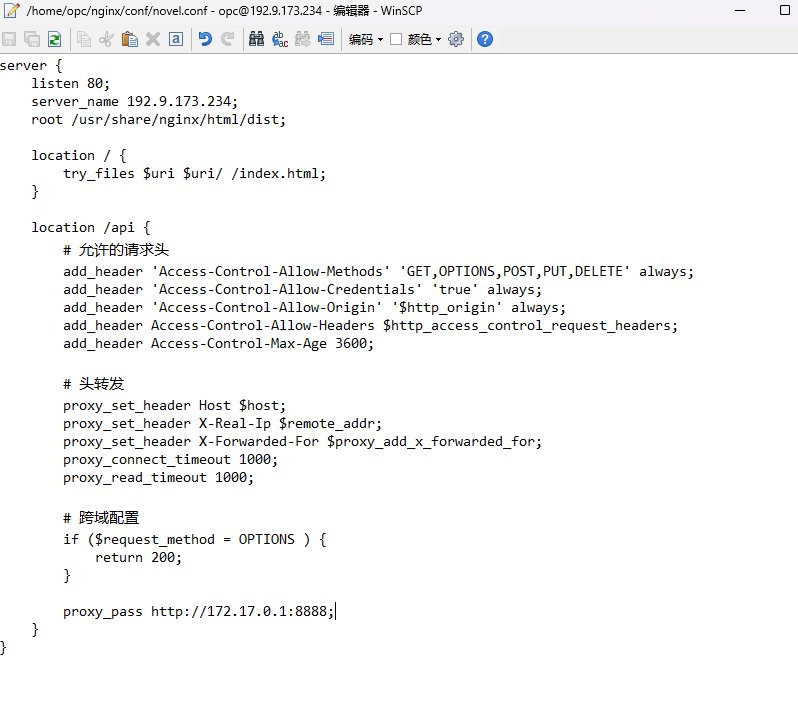
服务器需要打开端口，记得要配置安全设置。

部署环境，需要先在服务器新建nginx文件夹，然后在nginx里面继续新建log, html, conf三个文件夹，其中html放dist文件夹。

dist文件夹根据yarn build来生成，将生成的dist文件夹放入html内

conf里面是nginx的配置

novel.conf的配置如下



server {

listen 80;

server\_name 192.9.173.234;

root /usr/share/nginx/html/dist;

location / {

try\_files $uri $uri/ /index.html;

}

location /api {

# 允许的请求头

add\_header 'Access-Control-Allow-Methods' 'GET,OPTIONS,POST,PUT,DELETE' always;

add\_header 'Access-Control-Allow-Credentials' 'true' always;

add\_header 'Access-Control-Allow-Origin' '$http\_origin' always;

add\_header Access-Control-Allow-Headers $http\_access\_control\_request\_headers;

add\_header Access-Control-Max-Age 3600;

# 头转发

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-Ip $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_connect\_timeout 1000;

proxy\_read\_timeout 1000;

# 跨域配置

if ($request\_method = OPTIONS ) {

return 200;

}

proxy\_pass http://172.17.0.1:8888;

}

}

其中proxy\_pass <http://172.17.0.1:8888;的ip是docker的ip。>将所有后端请求转发到docker的ip来访问。

如果前后端分在不同的服务器的话，ip需要修改

其中要注意的是，所有请求都是/api开头的而不是localhost:8888/api开头的

然后后端是



Development environment:

MySQL 8.0

Redis 6.0

JDK 17

Maven 3.8

IntelliJ IDEA (optional)

Node 16.14

Installation environment under lubuntu

First install Docker

1. Update the package list

First, make sure your system's package list is up to date:

sudo apt-get update

2. Install dependencies

Docker requires some additional packages to use the virtualization features of the operating system. Install them:

sudo apt-get install \

apt-transport-https \\

ca-certificates \

curl \

gnupg \

lsb-release

3. Add the official GPG key for Docker

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

4. Set up the Docker repository

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ ubuntu \

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

5. Install the Docker Engine

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io

6. Start Docker and configure it to boot up

sudo systemctl enable docker

sudo systemctl start docker

Install MySQL 8.0

1. Pull the MySQL 8.0 image:

2. Run the MySQL container:

docker run --name mysql-container -e MYSQL\_ROOT\_PASSWORD=123456 -p 3306:3306 -d mysql:8.0

This will start a MySQL 8.0 container named mysql-container with the root password set to 123456. you can change this to a password of your choice.

Installing Redis 6.0

Pull the Redis image:

docker pull redis:6.0

Run the Redis container:

docker run --name redis-container -p 6379:6379 -d redis:6.0 redis-server --requirepass 123456

At this point, Redis 6.0 is installed and the password 123456 is set.

Adding Database Files

1. Creating the database

First, you can connect to the MySQL container and create the required database. Let's assume you set the root password to 123456.

docker exec -it mysql-container mysql -uroot -p123456 -e "CREATE DATABASE novel\_dev CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;"

This command uses the utf8mb4 character set and the associated checksum rules.

2. Importing the SQL file

Next, let's assume that you have a SQL file named novel\_dev.sql and that it is located in a path on your host (e.g. /path/to/novel\_dev.sql). You can import this file into the database you just created using the following command:

docker exec -i mysql-container mysql -uroot -pmy-secret-pw novel\_dev < /path/to/novel\_dev.sql

Install JDK17, Node 16.14, yarn and Maven.

# Use the official JDK 17 image as a base

sudo apt install oracle-java17-installer

# Install Node.js 16.x

sudo apt-get install -y nodejs

# Install Yarn

sudo apt-get install yarn

# Install Maven

# Download Maven

wget https://dlcdn.apache.org/maven/maven-3/3.9.4/binaries/apache-maven-3.9.4-bin.tar.gz

Unzip Maven: Use the tar command to unzip the file.

tar -xzvf apache-maven-3.9.4-bin.tar.gz

Move the Maven directory: Move the extracted Maven folder to /usr/local or another suitable location.

sudo mv apache-maven-3.9.4 /usr/local/maven

Update the PATH variable: Edit the ~/.bashrc or ~/.profile file and add Maven's bin directory to the PATH.

echo "export PATH=\$PATH:/usr/local/maven/bin" >> ~/.bashrc

source ~/.bashrc

At this point we have installed all the required environments

1. Download the backend source code

novel Backend service installation

2. Modify the data source configuration in the src/resources/application.yml configuration file.

3. Modify the redis connection configuration in the src/resources/application.yml configuration file.

4Modify the cross-domain configuration in application.yml according to the actual deployment of the front and back ends (this step can be ignored by default)

5 Run the following command in the root directory of the project to start the back-end service (if you have an IDE installed, you can import the source code and run it in the IDE)

Then run

mvn spring-boot:run

1 Download the front-end source code

2.

According to the actual deployment of the front-end and back-end, modify the VUE\_APP\_BASE\_API\_URL attribute in .env.development (by default, this step can be ignored).

3. Run the following command in the project root directory to install project dependencies

yarn install

4. In the project root directory, run the following command to start the project

yarn serve

5. Browser through http://localhost:8080