一、Docker部署

1.1.Docker环境准备

1.1.1.Ubuntu下安装

```
## Ubuntu
如果你过去安装过 docker，先删掉：
for pkg in docker.io docker-doc docker-compose podman-docker containerd runc; do apt-get remove $pkg; done

首先安装依赖：
apt update
apt install -y apt-transport-https ca-certificates curl software-properties-common  gnupg

信任 Docker 的 GPG 公钥并添加仓库：
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  tee /etc/apt/sources.list.d/docker.list > /dev/null

# 安装
apt update
apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

1.1.2. Docker的使用配置

1.1.2.1. 配置docker加速器

镜像加速代理（镜像下载慢使用）：https://dockerproxy.com/docs

仅使用dockerhub时可配置 daemon.json 文件： { "registry-mirrors": ["https://dockerproxy.com"] }

1. Docker Hub 官方镜像代理：

- 常规镜像代理
  官方命令：docker pull stilleshan/frpc:latest
  代理命令：docker pull dockerproxy.com/stilleshan/frpc:latest

- 根镜像代理
  官方命令：docker pull nginx:latest
  代理命令：docker pull dockerproxy.com/library/nginx:latest

2. GitHub Container Registry：

- 常规镜像代理
  官方命令：docker pull ghcr.io/username/image:tag
  代理命令：docker pull ghcr.dockerproxy.com/username/image:tag

3. Google Container Registry ：

- 常规镜像代理
  官方命令：docker pull gcr.io/username/image:tag
  代理命令：docker pull gcr.dockerproxy.com/username/image:tag

4. Google Kubernetes ：

- 常规镜像代理
  官方命令：docker pull k8s.gcr.io/username/image:tag
  代理命令：docker pull k8s.dockerproxy.com/username/image:tag

- 根镜像代理
  官方命令：docker pull k8s.gcr.io/coredns:1.6.5
  代理命令：docker pull k8s.dockerproxy.com/coredns:1.6.5

5. Quay.io ：

- 常规镜像代理
  官方命令：docker pull quay.io/username/image:tag
  代理命令：docker pull quay.dockerproxy.com/username/image:tag

1.1.2.2.配置方法

```
## 1.临时配置
格式如下：（不要加上前缀https://）
```

```
docker pull+镜像源地址+/+要拉取的镜像名

例如：镜像源为“docker.m.daocloud.io”，要拉取的镜像名为“hello-world”
docker pull docker.m.daocloud.io/hello-world

## 2.永久配置
格式如下：（需要加上前缀https://）
{
    "registry-mirrors": [
        "镜像源1",
        "镜像源2"
    ]
}

# 方法一：直接命令行输入直接将下列文本粘贴到终端中，然后回车运行即可。
tee /etc/docker/daemon.json <<-'EOF'
{
    "registry-mirrors": [
        "https://docker.m.daocloud.io",
        "https://docker.imgdb.de",
        "https://docker-0.unsee.tech",
        "https://docker.1ms.run",
        "https://func.ink",
        "https://lispy.org",
        "https://docker.xiaogenban1993.com"
    ]
}
EOF

方法二：通过文本编辑器打开daemon.json，然后粘贴首先打开配置文件：
nano /etc/docker/daemon.json
{
    "registry-mirrors": [
        "https://docker.m.daocloud.io",
        "https://docker.imgdb.de",
        "https://docker-0.unsee.tech",
        "https://docker.hlmirror.com",
        "https://docker.1ms.run",
        "https://func.ink",
        "https://lispy.org",
        "https://docker.xiaogenban1993.com"
    ]
}

## 3.重启docker服务
systemctl daemon-reload && sudo systemctl restart docker
```

1.1.3.启动

```
[root@localhost ~]# systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service
to /usr/lib/systemd/system/docker.service.
[root@localhost ~]# systemctl start docker
```

```
[root@localhost ~]# systemctl restart docker
[root@localhost ~]# docker info
Client:
...
Server:
...
 Server Version: 20.10.9
...
 Registry Mirrors:
  https://dsdpf8q4.mirror.aliyuncs.com/
  https://registry.docker-cn.com/
  https://docker.mirrors.ustc.edu.cn/
  http://hub-mirror.c.163.com/
  https://cr.console.aliyun.com/
 Live Restore Enabled: false
```

1.1.4.数据库配置

1.1.4.1.数据库的安装

> 方式一：选择直接在物理机部署数据库，如：mysql、mariadb（推荐）
>
> 方式二：选择数据库容器（不推荐，数据容易丢失）
>
> 注意：
>
> /etc/mysql/mariadb.conf.d/50-server.cnf 配置中的端口和允许远程连接的IP
>
> port = 3306
>
> bind-address = 0.0.0.0

```
# 安装mariadb数据库
apt install mariadb-server

# 初始化
mysql_secure_installation

# 进入数据库
mysql -u  root

# 创建数据库
CREATE DATABASE xblms DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;


# 创建数据库用户，允许本地登录
create user xblms@localhost identified by '123456';

# 设置用户权限为全部权限
```

```
grant all privileges on xblms.* to xblms@localhost identified by '123456';

# 刷新权限
flush privileges;

## 扩展，如果没有远程连接的权限，你可以使用以下命令授权：
GRANT ALL PRIVILEGES ON *.* TO 'xblms'@'%' IDENTIFIED BY '123456' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

1.1.4.2.数据库的备份与恢复

1.1.4.2.1. 数据库备份（远程）

```
cat << \EOF > dbbackup.sh
#!/bin/bash

# 云服务器地址
REMOTE_SERVER="你的公网IP"

# 远程服务器登录账号
REMOTE_USER="root"

# 远程服务器登录密码
REMOTE_PASSWORD="rxr642cj"

# 数据库端口
DB_PORT="3306"

# 数据库信息
DB_USER="xblms"
DB_PASSWORD="123456"
DB_NAME="xblms"

# 备份目录
BACKUP_DIR="/opt/DBbackup"

# 远程服务器上的指定文件夹
REMOTE_UPLOAD_DIR="/var/www/wwwroot/sitefiles/upload"

# 日期信息
DATE=$(date +"%Y%m%d%H%M%S")

# 创建备份目录（如果不存在）
mkdir -p $BACKUP_DIR

# 备份文件名
BACKUP_FILE="$BACKUP_DIR/$DB_NAME-$DATE.sql"

# 执行备份操作
echo "正在备份数据库 $DB_NAME..."
```

```
mysqldump -h $REMOTE_SERVER -P $DB_PORT -u $DB_USER -p$DB_PASSWORD $DB_NAME > $BACKUP_FILE

# 检查备份是否成功
if [ $? -eq 0 ]; then
    echo "备份成功：$BACKUP_FILE"
else
    echo "备份失败"
fi

# 清理旧备份（保留最近7天的备份）
find $BACKUP_DIR -type f -name "$DB_NAME-*.sql" -mtime +7 -delete

# 打包并备份远程服务器的 tool 文件夹到本地
LOCAL_BACKUP_FILE="$BACKUP_DIR/upload_backup_$DATE.tar.gz"
echo "正在打包并备份远程服务器的 $REMOTE_TOOL_DIR 文件夹到本地 $LOCAL_BACKUP_FILE..."

# 检查 sshpass 是否安装
if ! command -v sshpass &> /dev/null; then
    echo "sshpass 未安装，请先安装 sshpass 工具。"
    exit 1
fi

echo "正在备份 $REMOTE_UPLOAD_DIR..."
sshpass -p "$REMOTE_PASSWORD" scp -r $REMOTE_USER@$REMOTE_SERVER:$REMOTE_UPLOAD_DIR  $BACKUP_DIR/uploadbak
if [ $? -eq 0 ]; then
    # 进入临时目录
    cd $BACKUP_DIR/uploadbak
    # 打包时去掉路径前缀
    tar -zcvf $LOCAL_BACKUP_FILE --strip-components=1 .
    if [ $? -eq 0 ]; then
        echo "远程文件夹 $REMOTE_UPLOAD_DIR 备份到 $LOCAL_BACKUP_FILE 成功。"
        rm -rf $BACKUP_DIR/uploadbak
    else
        echo "打包远程文件夹 $REMOTE_UPLOAD_DIR 失败，请检查错误信息。"
    fi
else
    echo "从远程服务器复制 $REMOTE_UPLOAD_DIR 文件夹到本地失败，请检查网络、账号密码或权限。"
fi
EOF
```

1.1.4.2.2. 数据库备份（本地）

```
cat << \EOF > dbbackup_local.sh
#!/bin/bash

# 本地服务器地址
LOCAL_SERVER="127.0.0.1"

# 本地服务器登录账号
```

```
LOCAL_USER="root"

# 本地服务器登录密码
LOCAL_PASSWORD="123456"

# 数据库端口
DB_PORT="3306"

# 数据库信息
DB_USER="xblms"
DB_PASSWORD="123456"
DB_NAME="xblms"

# 备份目录
LOCAL_BACKUP_DIR="/opt/DBbackup_local"

# 本地服务器上的指定文件夹
LOCAL_UPLOAD_DIR="/var/www/wwwroot/sitefiles/upload"

# 日期信息
DATE=$(date +"%Y%m%d%H%M%S")

# 创建备份目录（如果不存在）
mkdir -p $LOCAL_BACKUP_DIR

# 备份文件名
LOCAL_BACKUP_DB_FILE="$LOCAL_BACKUP_DIR/$DB_NAME-$DATE.sql"

# 执行备份操作
echo "正在备份数据库 $DB_NAME..."
mysqldump -h $LOCAL_SERVER -P $DB_PORT -u$DB_USER -p$DB_PASSWORD $DB_NAME > $LOCAL_BACKUP_DB_FILE

# 检查备份是否成功
if [ $? -eq 0 ]; then
    echo "备份成功：$LCOAL_BACKUP_DB_FILE"
else
    echo "备份失败"
fi

# 清理旧备份（保留最近7天的备份）
find $LCOAL_BACKUP_DIR -type f -name "$DB_NAME-*.sql" -mtime +7 -delete

# 打包并备份本地服务器的 upload 文件夹到本地
LOCAL_BACKUP_FILES="$LOCAL_BACKUP_DIR/upload_backup_$DATE.tar.gz"
echo "正在打包并备份本地服务器的 $LOCAL_UPLOAD_DIR 文件夹到本地 $LOCAL_BACKUP_FILES..."

echo "正在备份 $LOCAL_UPLOAD_DIR..."
cp -r $LOCAL_UPLOAD_DIR  $LOCAL_BACKUP_DIR/uploadbak
if [ $? -eq 0 ]; then
    # 进入临时目录
    cd $LOCAL_BACKUP_DIR/uploadbak
    # 打包时去掉路径前缀
    tar -zcvf $LOCAL_BACKUP_FILES --strip-components=1 .
    if [ $? -eq 0 ]; then
```

```
        echo "本地文件夹 $LOCAL_UPLOAD_DIR 备份到 $LOCAL_BACKUP_FILES 成功。"
        rm -rf $LOCAL_BACKUP_DIR/uploadbak
    else
        echo "打包本地文件夹 $LOCAL_UPLOAD_DIR 失败，请检查错误信息。"
    fi
else
    echo "从本地服务器复制 $LOCAL_UPLOAD_DIR 文件夹到本地失败，请检查网络、账号密码或权限。"
fi
EOF
```

1.1.4.2.3. 数据恢复

```
cat << \EOF > dbupdate.sh
#!/bin/bash

# 服务器地址
LOCAL_SERVER="localhost"
# 数据库端口
DB_PORT="3306"
# 数据库信息  \ 是转义字符
DB_USER="xblms"
DB_PASSWORD="123456"
DB_NAME="xblms"

# 备份文件路径，可根据实际情况修改
BACKUP_FILE="/opt/DBbackup_local"

# 检查备份文件夹是否存在
if [ ! -d "$BACKUP_FILE" ]; then
    echo "备份文件 $BACKUP_FILE 不存在，请检查路径。"
    exit 1
fi

# 查找备份目录下所有以 .sql 结尾的文件
BACKUP_FILES=("$BACKUP_FILE"/*.sql)
FOUND_FILES=false

for BACKUP_FILE in "${BACKUP_FILES[@]}"; do
    if [ -f "$BACKUP_FILE" ]; then
        FOUND_FILES=true
        break
    fi
done

if [ "$FOUND_FILES" = false ]; then
    echo "备份目录 $BACKUP_FILE 中未找到有效的 .sql 备份文件，请检查。"
    exit 1
fi

# */停止 Nginx 和 xblms 服务
echo "正在停止 Nginx 服务..."
systemctl stop nginx
```

```
if [ $? -eq 0 ]; then
    echo "Nginx 服务已停止。"
else
    echo "停止 Nginx 服务失败，请检查错误信息。"
fi

echo "正在停止 xblms 服务..."
systemctl stop xblms
if [ $? -eq 0 ]; then
    echo "xblms 服务已停止。"
else
    echo "停止 xblms 服务失败，请检查错误信息。"
fi

# 删除指定数据库内的所有表
echo "正在删除 $DB_NAME 数据库内的所有表..."
TABLES=$(mysql -h $LOCAL_SERVER -P $DB_PORT -u $DB_USER -p$DB_PASSWORD -N -s -e "SELECT table_name FROM information_schema.tables WHERE table_schema = '$DB_NAME'")
for table in $TABLES; do
    mysql -h $LOCAL_SERVER -P $DB_PORT -u $DB_USER -p$DB_PASSWORD -e "DROP TABLE IF EXISTS $DB_NAME.$table"
done
echo "数据库 $DB_NAME 内的所有表已删除。"

# 恢复数据库
echo "正在恢复数据库 $DB_NAME..."
for BACKUP_FILE in "${BACKUP_FILES[@]}"; do
    if [ -f "$BACKUP_FILE" ]; then
        echo "正在恢复备份文件 $BACKUP_FILE 到数据库 $DB_NAME..."
        mysql -h $LOCAL_SERVER -P $DB_PORT -u $DB_USER -p$DB_PASSWORD $DB_NAME < "$BACKUP_FILE"
        # 检查恢复是否成功
        if [ $? -eq 0 ]; then
            echo "备份文件 $BACKUP_FILE 恢复到数据库 $DB_NAME 成功。"
        else
            echo "备份文件 $BACKUP_FILE 恢复到数据库 $DB_NAME 失败，请检查错误信息。"
        fi
    fi
done

# 解压文件路径
UNZIP_FILE="/opt/DBbackup_local"
UNZIP_DIR="/var/www/wwwroot/sitefiles/upload"

# 查找备份目录下所有以upload_backup_*.tar.gz 结尾的文件
UNZIP_FILES=("$UNZIP_FILE"/upload_backup_*.tar.gz)
FOUND_DIRS=false

for UNZIP_FILE in "${UNZIP_FILES[@]}"; do
    if [ -f "$UNZIP_FILE" ]; then
        FOUND_DIRS=true
        break
    fi
done
```

```
92
93  if [ "$FOUND_DIRS" = false ]; then
94      echo "备份目录 $UNZIP_FILE 中未找到有效的upload_backup_*.tar.gz 备份文件，请检
    查。"
95      exit 1
96  fi
97
98  # 检查解压文件是否存在
99  if [ -f "$UNZIP_FILE" ]; then
100     # 解压指定文件到指定文件夹
101     echo "正在解压 $UNZIP_FILE 到 $UNZIP_DIR..."
102     mkdir -p "$UNZIP_DIR"
103     tar -xzvf "$UNZIP_FILE" -C "$UNZIP_DIR"
104     if [ $? -eq 0 ]; then
105         echo "文件 $UNZIP_FILE 解压到 $UNZIP_DIR 成功。"
106     else
107         echo "文件 $UNZIP_FILE 解压到 $UNZIP_DIR 失败，请检查错误信息。"
108     fi
109 else
110     echo "解压文件 $UNZIP_FILE 不存在，请检查路径。"
111 fi
112
113 # 启动 Nginx 和 xblms 服务
114 echo "正在启动 Nginx 服务..."
115 systemctl start nginx
116 if [ $? -eq 0 ]; then
117     echo "Nginx 服务已启动。"
118 else
119     echo "启动 Nginx 服务失败，请检查错误信息。"
120 fi
121
122 echo "正在启动 xblms 服务..."
123 systemctl start xblms
124 if [ $? -eq 0 ]; then
125     echo "xblms 服务已启动。"
126 else
127     echo "启动 xblms 服务失败，请检查错误信息。"
128 fi
129 EOF
```

1.2.镜像构建

1.2.1.拉取基础镜像

```
1  # 自己的阿里镜像源
2  docker pull registry.cn-hangzhou.aliyuncs.com/tools-docker/aspnet:8.0
3  docker pull registry.cn-hangzhou.aliyuncs.com/tools-docker/sdk:8.0
```

1.2.2.开始构建

1.2.1.1.拉取源代码

```
cd /opt
git clone https://gitee.com/xblms/xblmes.git
```

1.2.1.2.修改Dockerfile文件

nano ./Dockerfile

Ctrl + X 退出提示是否保存时选择是

```
# 1.构建阶段
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build

# 配置工作目录
WORKDIR /app

# 复制项目文件并还原依赖
COPY ["src/XBLMS.Web/XBLMS.Web.csproj", "src/XBLMS.Web/"]
COPY ["src/XBLMS.Core/XBLMS.Core.csproj", "src/XBLMS.Core/"]
COPY ["src/XBLMS/XBLMS.csproj", "src/XBLMS/"]
COPY ["src/Datory/Datory.csproj", "src/Datory/"]
RUN dotnet restore "src/XBLMS.Web/XBLMS.Web.csproj"

# 复制源代码并构建
COPY . .
WORKDIR "/app/src/XBLMS.Web"
RUN dotnet build "XBLMS.Web.csproj" -c Release -o /app/build

# 2.发布阶段
FROM build AS publish
RUN dotnet publish "XBLMS.Web.csproj" -c Release -o /app/publish

# 3.运行阶段
FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS base
LABEL maintainer="xblm"
# docker旧版时，maintainer使用此格式(使用时需要将#号去掉)
# MAINTAINER xblm

WORKDIR /app
EXPOSE 8888

# 复制发布文件并设置权限
COPY --from=publish /app/publish .

# 设置时间为中国上海，默认为UTC时间
ENV TZ=Asia/Shanghai
```

```
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

# 配置程序运行端口，如果程序不使用默认的80端口这里一定要设置（程序运行端口）
ENV ASPNETCORE_URLS=http://+:8888
# docker旧版使用此格式(使用时需要将#号去掉)
# ENV ASPNETCORE_URLS http://+:8888

# 健康检查
HEALTHCHECK --interval=30s --timeout=3s --start-period=60s \
  CMD wget -q --spider http://localhost:8888/health || exit 1

# 启动应用
ENTRYPOINT ["dotnet", "XBLMS.Web.dll"]
```

1.2.1.3.开始构建

```
# 创建一个名为xblmsexam的8.0版本镜像
docker build -t xblmsexam:8.0 .
```

1.2.1.4. 启动容器

```
docker run -itd --name xblms --restart always  -p 8888:8888 xblmsexam:8.0
```

1.2.1.5.开始安装

> 访问 http://localhost:8888/admin/install 安装系统

1.2.3. 关于持久化

```
#前面的步骤删除容器后数据会丢失，以下是数据的持久化（存在每次更新都要重新安装系统！）
# 1.使用Docker Volume（推荐）
# 创建名为 xblms-data 的Volume（命名Volume便于管理）
docker volume create xblms-data
docker run -itd --name xblms --restart always -v xblms:/app  -p 8888:8888
xblmsexam:8.0

# 查看Volume列表
docker volume ls

# 查看xblms-data的实际存储路径（宿主机上，一般无需手动修改）
docker volume inspect xblms-data

# 删除Volume（谨慎！数据会丢失）
docker volume rm xblms-data

```

```
# 2.使用绑定挂载 Bind Mount（不推荐）
# 临时创建容器
docker run -d --rm --name temp-xblms  xblmsexam:v9  /bin/bash -c "sleep infinity"

# 拷贝文件到当前目录
docker cp  temp-xblms:/app/ .

# 将复制出来的文件夹内的文件全部复制到宿主机你要映射的目录下，如：/opt/xblms_data
cp -r ./app/*   /opt/xblms_data

# 重新运行正式的容器
docker run -itd --name xblms --restart always -v /opt/xblms_data:/app  -p 8888:8888  xblmsexam:8.0
```

二、Docker compose部署

2.1. 镜像准备

方式一：可以自己构建

方式二：直接用我构建的，默认版本为最新版本（此处以此种方法为主）

```
# 数据库镜像（mariadb）
docker  pull  registry.cn-hangzhou.aliyuncs.com/tools-docker/mariadb:v11.7.2

# 星期八在线考试系统镜像（xblms_v8.1.8）
docker  pull  registry.cn-hangzhou.aliyuncs.com/tools-docker/xblms:v8.1.8
```

2.2.编写yml文件

xblms_app_data　用于 xblms 容器 /app 目录的持久化卷

mariadb_data　用于 mariadb 数据的持久化卷

/opt/data/mariadb_conf　物理机上配置存放数据库的配置文件，需要手动创建

```
# 创建相关目录
mkdir -pv /opt/data

# xblms考试系统的yml文件
cat << \EOF > /opt/data/xblms.yml
version: '3.1'
```

```
7
8  volumes:
9    xblms_app_data:
10     driver: local
11   mariadb_data:
12     driver: local
13
14 services:
15   mariadb:
16     image: registry.cn-hangzhou.aliyuncs.com/tools-docker/mariadb:v11.7.2
17     container_name: mariadb
18     environment:
19       - MARIADB_ROOT_PASSWORD=123456
20       - MARIADB_USER=xblms
21       - MARIADB_PASSWORD=123456
22       - MARIADB_DATABASE=xblms
23       - TZ=Asia/Shanghai
24     restart: always
25     ports:
26       - "33306:3306"
27     volumes:
28       - mariadb_data:/var/lib/mysql
29       - /opt/data/mariadb_conf:/etc/mysql/conf.d
30     healthcheck:
31       test: ["CMD", "mysqladmin", "ping", "-h", "localhost", "-u", "root", "-
   p123456"]
32       interval: 10s
33       timeout: 5s
34       retries: 3
35       start_period: 30s
36
37   xblms:
38     image: registry.cn-hangzhou.aliyuncs.com/tools-docker/xblms:v8.1.8
39     container_name: xblms
40     restart: always
41     ports:
42       - "8888:8888"
43     depends_on:
44       - mariadb
45     environment:
46       - DB_HOST=mariadb
47       - DB_PORT=3306
48       - DB_USER=xblms
49       - DB_PASS=123456
50       - DB_NAME=xblms
51       - TZ=Asia/Shanghai
52     volumes:
53       - xblms_app_data:/app
54     healthcheck:
55       test: ["CMD", "curl", "-f", "http://localhost:8888/health"]
56       interval: 15s
57       timeout: 5s
58       retries: 3
59       start_period: 60s
60 EOF
```

```
# 创建数据库配置目录
mkdir -pv /opt/data/mariadb.conf

# 数据库配置文件
cat << \EOF >  /opt/data/mariadb.conf/my.cnf
[mysqld]
character-set-server = utf8mb4
collation-server = utf8mb4_unicode_ci
init_connect = 'SET NAMES utf8mb4'

[client]
default-character-set = utf8mb4
EOF
```

2.3. 运行

```
cd /opt/data

docker compose -f ./xblms.yml up -d
```

2.4.开始安装

在选择数据库时一定要填写物理机的IP，端口为yml文件中对应的自定义数据库端口，此处为33306

http://IP/admin/install