**1.Docker安装**

搭建gcc环境（gcc是编程语言译器）

yum -y install gcc

yum -y install gcc-c++

安装需要的软件包

yum install -y yum-utils

装镜像仓库

因为docker的服务器是在国外，所以有时候从仓库中下载镜像的时候会连接被拒绝或者连接超时的情况！

因此可以使用阿里云镜像仓库

yum-config-manager --add-repo <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

新yum软件包索引

yum makecache fast

安装docker引擎

yum install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin

启动docker

systemctl start docker

查看docker版本

docker version

配置阿里云镜像加速

为了提高镜像的拉取、发布的速度，可以配置阿里云镜像加速（推荐）

[获取镜像加速器地址](https://cr.console.aliyun.com/cn-hangzhou/instances/mirrors)

在CentOS下配置镜像加速器

mkdir -p /etc/docker

tee /etc/docker/daemon.json <<-'EOF'

{

  "registry-mirrors": ["你个人的阿里云镜像加速器地址"]

}

EOF

systemctl daemon-reload

systemctl restart docker

完成

**2.Docker的前端部署（nginx）**

步骤：

创建所需的文件夹 ----> 创建见nginx镜像和容器 ----> 将nginx容器内容cp到文件夹中 ----> 删除容器 ----> 创建nginx新容器与cp文件夹绑定

----> 修改配置文件 ----> 将前端项目放入html ---->重启

创建文件夹

#需要一个conf文件存放目录，和html文件目录,及日志存放目录

mkdir -p /data/applications/nginx/conf

mkdir -p /data/applications/nginx/html

mkdir -p /data/logs

创建见nginx镜像和容器

#创建镜像

docker pull nginx:1.18.0

#创建容器

docker run --name nginx -p 85:80 -d nginx:1.18.0

#创建了一个镜像名为ningx 对外开放85端口对内80端口

将nginx容器内容cp到文件夹中

docker cp nginx:/etc/nginx/nginx.conf /data/applications/nginx/conf/nginx.conf

docker cp nginx:/etc/nginx/conf.d /data/applications/nginx/conf/conf.d

docker cp nginx:/usr/share/nginx/html /data/applications/nginx/

删除容器

docker rm -f nginx #nginx为镜像名称

或

docker stop nginx

docker rm nginx

创建nginx新容器与cp文件夹绑定

docker run -d \

-p 85:80 \

-p 443:443 \ #ssl证书要用没有不写

--name nginx \

--restart=always \

-e TZ="Asia/Shanghai" \

-v /data/applications/nginx/conf/nginx.conf:/etc/nginx/nginx.conf \

-v /data/applications/nginx/conf/conf.d:/etc/nginx/conf.d \

-v /data/logs/nginx:/var/log/nginx \

-v /data/applications/nginx/html:/usr/share/nginx/html \

-v /data/applications/nginx/cert:/etc/nginx/cert \ #ssl证书要用没有不写

nginx:1.18.0

#-p与之前容器的nginx一致

到此时就可以访问是否成功

修改配置文件**default.conf改成如下**

server {

listen 80; #这个对内的，（我的是对你80对外85）

listen [::]:80;

server\_name localhost; #有域名可以换成域名

#charset koi8-r;

#access\_log /var/log/nginx/host.access.log main;

location / {

root /usr/share/nginx/html;

index index.html index.htm;

}

#没部署可以不要写，写上也没错就是访问不到

location /prod-api/{ #访问后端格式

#proxy\_redirect off;

proxy\_set\_header Host $http\_host;

proxy\_set\_header REMOTE-HOST $remote\_addr;

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

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

proxy\_pass http://xx.xx.xx.xx:8080/; #后端端口

}

}

将前端项目放入html

将前端项目打包，将打包好的dist下的文件全部复制到/data/applications/nginx/html下

重启

docker restart nginx

#重启可以在修改为配置文件后执行

**3.Docker的后端部署(boot,mysql,redis)**

步骤：

redis: 创建镜像 ----> 创建容器

mysql: 创建镜像 ----> 创建容器 ----> 存入数据

boot: 修改代码 ----> 打包 ----> 编写dockerfile ----> 创建java镜像----> 创建boot镜像 ----> 创建容器

创建redis镜像

docker pull redis:6.0.8

创建redis容器

docker run -d --name redis --restart=always -p 6379:6379 redis:6.0.8 --requirepass "123456"

#如果6379被占用，且没有redis没有密码

docker run -d --name redis --restart=always -p 6378:6379 redis:6.0.8

redis完成

创建mysql镜像

docker pull mysal:8.0.19

创建mysql容器

docker run --name mysql --restart=always -v /myapp/mysql:/var/lib/mysql -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=root -d mysql:8.0.19

#创建/myapp/mysql

存入数据

Navcat连接docker数据库域名服务器ip端口号3306（具体看你容器中设置）账号密码root

boot项目的修改

将访问本地的mysql和redis改为服务器的

打包

clean+package 打出jar包

编写dockerfile

在服务器中创建docker文件夹(名称随意)，进入文件夹，将打好的jar放入，在创建Dockerfile文件进入编写（编写如下）

#基础镜像使用jdk1.8

FROM java:8

#作者

MAINTAINER pan

# VOLUME 指定临时文件目录为/tmp，在主机/var/lib/docker目录下创建了一个临时文件并链接到容器的/tmp

VOLUME /tmp

# 将jar包添加到容器中并更名 ruoyi-admin.jar给我你的jar包名

ADD ruoyi-admin.jar app.jar

# 运行jar包

RUN bash -c 'touch /app.jar'

# 为了缩短 Tomcat 启动时间，添加一个系统属性指向 “/dev/./urandom” 作为 Entropy Source

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]

#暴露8080端口 根据自己需求填写

EXPOSE 8080

以上操作在xftp操作

创建java镜像

docker pull java:8

创建boot镜像

cd到你创建的文件夹下即jar同级

#创建创建boot镜像名字为ruoyi,根据自己需求改名

docker build -t ruoyo:1.0 .

创建容器

docker  run -d --name ruoyi -p 8080:8080 --restart=always ruoyi:1.0

完成

查看项目是否启动

docker logs -f -t --tail 1000 ruoyi(容器名称)