## 环境搭建

### 安装Jdk

#官网下载安装包并上传到服务器

#解压

tar -xzf jdk-17.0.10\_linux-x64\_bin.tar.gz

#打开这2个配置文件

vim ~/.profile

vim ~/.bashrc

#都增加以下内容

export JAVA\_HOME=/home/PANTA/Jdk/jdk-17.0.10

export PATH=$JAVA\_HOME/bin:$PATH

#刷新配置并生效

source ~/.profile

source ~/.bashrc

#检查安装是否成功

Java - version

### 安装mysql

#安装mysql服务

sudo yum install mysql-server

#启动服务

sudo systemctl start mysqld

#设置开机自启

sudo systemctl enable mysqld

#登录mysql(初始没密码)

mysql -u root -p

#修改密码

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'Panta1023';

#执行flush privileges; 刷新权限

flush privileges;

#开启远程连接

update mysql.user set host='%' where user='root';

#执行flush privileges; 刷新权限

flush privileges;

#退出mysql窗口，在linux窗口下重启mysql服务

service mysqld restart

### 安装Docker

#下载docker包 版本可以选择较新的

wget https://download.docker.com/linux/static/stable/x86\_64/docker-18.09.9.tgz

#解压

tar zxf docker-18.09.9.tgz

#移动解压后的文件夹到/usr/bin （ps！！只移动docker目录下的，所以加\\*）

mv docker/\* /usr/bin

#写入docker.service

cat >/usr/lib/systemd/system/docker.service <<EOF

[Unit]

Description=Docker Application Container Engine

Documentation=https://docs.docker.com

After=network-online.target firewalld.service

Wants=network-online.target

[Service]

Type=notify

ExecStart=/usr/bin/dockerd --

ExecReload=/bin/kill -s HUP $MAINPID

LimitNOFILE=infinity

LimitNPROC=infinity

TimeoutStartSec=0

Delegate=yes

KillMode=process

Restart=on-failure

StartLimitBurst=3

StartLimitInterval=60s

[Install]

WantedBy=multi-user.target

EOF

#启动docker

systemctl start docker

#设置开机自启动

systemctl enable docker

#查看docker版本

docker version

#运行hello-world 检验安装是否成功

docker run hello-world

### 安装Nginx

#下载压缩包

wget -c http://nginx.org/download/nginx-1.24.0.tar.gz

#解压压缩包

tar -zxvf nginx-1.24.0.tar.gz

#初始化Nginx(进入nginx的根目录执行下面的命令)

./configure

#进行make(依次执行下面的命令)

make

make install

#查看Nginx是否成功安装

whereis nginx

### 安装Redis

安装方式采用docker

#### 普通安装方式

#拉取镜像(最新版本)

docker pull redis

#设置redis外部挂载配置

## 创建目录

mkdir -p /home/PANTA/Redis/conf

mkdir -p /home/PANTA/Redis/data

## 创建文件

touch /home/PANTA/Redis/conf/redis.conf

#创建容器并启动

# Docker 创建 Redis 容器命令

docker run \

--restart=always \

--log-opt max-size=100m \

--log-opt max-file=2 \

-p 6379:6379 \

--name panta-redis \

-v /opt/redis/conf/redis.conf:/etc/redis/redis.conf \

-v /opt/redis/data:/data \

-d redis redis-server /etc/redis/redis.conf \

--appendonly yes \

--requirepass panta1023

### 安装Kafka

#创建一个文件夹目录(随意)

mkdir Kafka

#在文件夹里下载kafka安装包（下载前可以先看看地址能不能在浏览器打开，也可以选择一个合适的版本。如果linux里无法下载，就在windos下载好上传到linux）

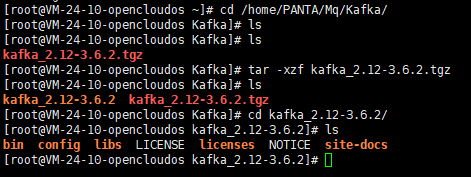
wget <https://dlcdn.apache.org/kafka/3.6.2/kafka_2.12-3.6.2.tgz>

#解压

tar -xzf kafka\_2.12-3.6.2.tgz

#进入解压的文件夹查看内容

cd kafka\_2.12-3.6.2/



#配置远程访问

# 修改 kraft 里面的配置文件

vim config/kraft/server.properties

把 advertised.listeners 的 localhost 修改为当前服务器的公网 IP

#启动

#要进入解压的文件夹里面执行 格式化 kraft 文件夹（新安装后只需执行一次）

KAFKA\_CLUSTER\_ID="$(bin/kafka-storage.sh random-uuid)"

bin/kafka-storage.sh format -t $KAFKA\_CLUSTER\_ID -c config/kraft/server.properties

# 启动

bin/kafka-server-start.sh -daemon config/kraft/server.properties &

----------------------------------------------------

## Docker配套镜像安装

### 安装Portainer

#Portainer是Docker的图形化管理工具

# 查询当前有哪些Portainer镜像

docker search portainer

# 下载镜像

docker pull docker.io/portainer/portainer

#个人学习选择单机运行

#该语句用宿主机9000端口关联容器中的9000端口，并给容器起名为portainer-test。执行完该命令之后，使用该机器IP:PORT即可访问Portainer。

#访问方式：http://IP:9000

docker run -d -p 9000:9000 \

--restart=always \

-v /var/run/docker.sock:/var/run/docker.sock \

--name prtainer-test \

docker.io/portainer/portainer

### Docker内运行JDK

---------制作JDK17的镜像文件Dockerfile---------

FROM centos:7

MAINTAINER panta

WORKDIR /usr/local/java

ADD jdk-17\_linux-x64\_bin.tar.gz /usr/local/java/

ENV JAVA\_HOME=/usr/local/java/jdk-17.0.8

ENV CLASSPATH=.:$JAVA\_HOME/lib/jrt-fs.jar

ENV PATH=$PATH:$JAVA\_HOME/bin

#ps:要add的文件要和Dockerfile放在一个目录下。不然build后名称和tag会出现nono

----------------运行镜像-----------

#构建Docker镜像

docker build -t='jdk17' . //其中，-t参数指定镜像的名字和标签-t name:tag)，.表示Dockerfile在当前目录下

#运行Docker镜像

docker run -di --name jdk17 -d jdk17

#进入容器查看是否成功

docker exec -it 容器ID /bin/bash

#退出容器

Ctrl+P+Q

### Docker运行java项目

--------制作项目的镜像文件Dockerfile---------

# 继承 java:17 这个镜像

FROM jdk17

# 作者

MAINTAINER panta

# 设置当前生成的镜像工作目录

WORKDIR /app

# 将宿主机中的jar包 copy到容器中 /app目录下, 并改名为 panta-boot-link.jar

COPY panta-boot-link.jar /app/panta-boot-link.jar

# 运行JAR文件

ENTRYPOINT ["java", "-jar", "panta-boot-link.jar", "--server.port=9001"]

# 暴露端口号

EXPOSE 9001

--------构建和运行镜像----------

#构建Docker镜像

docker build -t panta-boot-link:1.0 . //其中，-t参数指定镜像的名字和标签，.表示Dockerfile在当前目录下。

#启动SpringBoot项目坐在的容器

docker run -d --name panta-boot-link -p 9001:9001 panta-boot-link:1.0

ps:!!可以用自己制作的jdk镜像，嫌麻烦也可以直接引用docker的openjk镜像

## Nginx里运行vue项目

#vue项目先打包好(打包成dist文件夹)

#将dist文件夹放在服务器上任意文件夹下面

#修改nginx的配置文件(我的在 /usr/local/nginx/conf/nginx.conf)

#添加如下内容(作用为监听9002端口，映射到对应的vue项目文件夹路径，/api的调用重定向到对应的后台地址)

server {

listen 9002;

server\_name localhost;

location / {

root /home/PANTA/Vue/zhuque\_dist;

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

}

location /api {

rewrite /api/(.\*)$ /$1 break;

proxy\_pass http://175.178.1.5:9001;

}

}

## 常用方法

### 编辑sh文件来执行繁琐的重复命令

#创建sh文件

vim xxx.sh

#编辑sh文件例子

#!/bin/sh

cd /home/PANTA/Dockerfile/Panta\_Link/

docker build -t panta-boot-link:1.0 .

docker run -d --name panta-boot-link -p 9001:9001 panta-boot-link:1.0

#授权 777=最高权限

chmod 777 xxx.sh

#执行sh文件

sh xxx.sh

### Nginx的一些常用命令

以下命令都是在进入安装Nginx的sbin目录中操作，我这里是 /usr/local/nginx/sbin

启动：./nginx

关闭：./nginx -s stop

重启：./nginx -s reload

----------------------------------------------------

修改配置文件 vim /usr/local/nginx/conf/nginx.conf（！！一定要先把默认的server删掉！！！）

首行添加 user root

模块里添加(测试映射一个图片文件夹路径，使其可以在浏览器直接访问)

server {

listen 80;

server\_name localhost;

location /Image/ {

# 存放图片的路径

root /home/PANTA/FileServer/;

}

}

### Docker的一些常用命令

#更换镜像源

#编辑/etc/docker目录下的文件daemon.json，加入以下内容（没有就创建）

{

"registry-mirrors": [

"https://dockerproxy.com",

"https://hub-mirror.c.163.com",

"https://mirror.baidubce.com",

"https://ccr.ccs.tencentyun.com"

]

}

#刷新配置文件

systemctl daemon-reload

#重启docker服务

systemctl restart docker.service

#查看配置

docker info

#重启容器

docker restart 容器name