项目八

docker24.0.7+zookeeper(wurstmeister/zookeeper)latest+kafka(wurstmeister/kafka)

使用docker实现同网段的一台虚拟机内的容器拉取zookeeper，另一台虚拟机容器拉取kafka，并基于单台kafka容器的主题(topic)实现以控制台消息生产(produce)和消费(consume)为主的应用测试

# 安装docker

**安装gcc**

yum -y install gcc

**以及gcc-c++**

yum -y install gcc-c++

**需要的安装包**

sudo yum install -y yum-utils

**设置镜像仓库**

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

**更新yum软件包索引（安装前的工作，非必须）**

yum makecache fast -y

**配置阿里云镜像加速**

sudo mkdir -p /etc/docker

sudo tee /etc/docker/daemon.json <<-'EOF'

{

"registry-mirrors": ["https://pak31uuv.mirror.aliyuncs.com"]

}

EOF

**安装docker docker-ce：社区版的，docker-ee：企业版的**

sudo yum install docker-ce docker-ce-cli containerd.io

**启动docker服务**

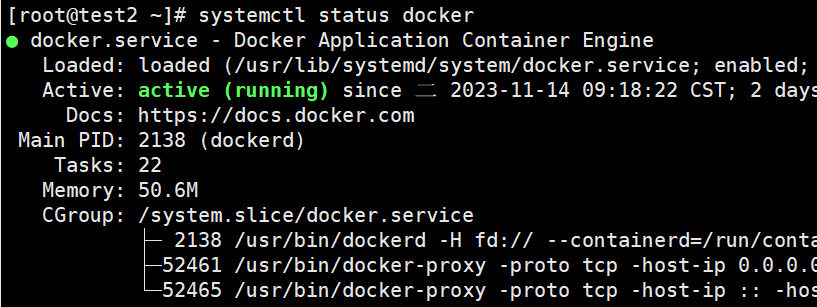
systemctl start docker

**docker开机自启动**

systemctl enable docker

**查看docker服务状态**

systemctl status docker



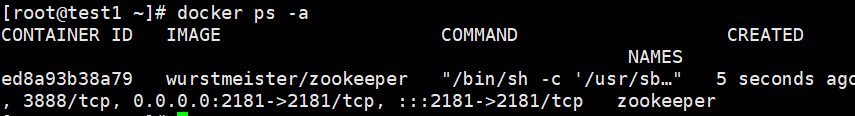
# 部署单台zookeeper容器

**拉取 ZooKeeper 镜像：**

docker pull wurstmeister/zookeeper

**创建并运行 ZooKeeper 容器：**

sudo docker run --restart=always -d --name zookeeper01 --network app-tier -p 2181:2181 -e ALLOW\_ANONYMOUS\_LOGIN=yes wurstmeister/zookeeper



# 部署单台kafka容器

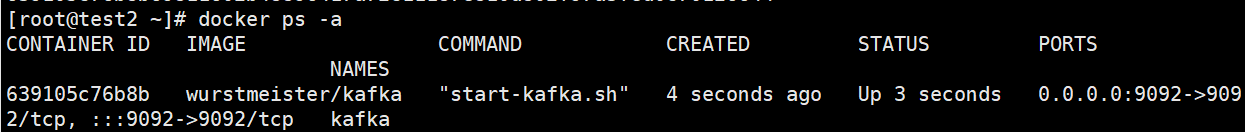
确保安装了docker

**在另一台虚拟机上拉取Kafka 镜像：**

docker pull wurstmeister/kafka

**创建并运行 Kafka 容器：使用以下命令创建并运行 Kafka 容器：**

docker run -d --name kafka -p 9092:9092 -e KAFKA\_ADVERTISED\_HOST\_NAME=192.168.88.167 -e KAFKA\_ZOOKEEPER\_CONNECT=192.168.88.166:2181 -e KAFKA\_ADVERTISED\_PORT=9092 wurstmeister/kafka



**运行以下命令进入Kafka容器的命令行界面：**

docker exec -it kafka /bin/bash

**在Kafka容器的命令行界面中，执行以下命令来创建一个名为"topic"的主题：**

kafka-topics.sh --create --topic topic --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

**运行以下命令进入 Kafka 容器的命令行界面：**

docker exec -it kafka /bin/bash

**在 Kafka 容器的命令行界面中，执行以下命令来生产消息到 "topic" 主题：**

kafka-console-producer.sh --topic topic --bootstrap-server localhost:9092

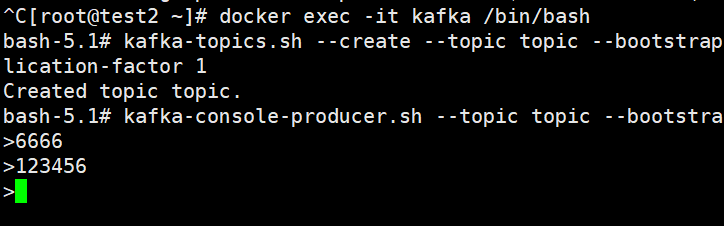
**运行以下命令进入 Kafka 容器的命令行界面：**

docker exec -it kafka /bin/bash

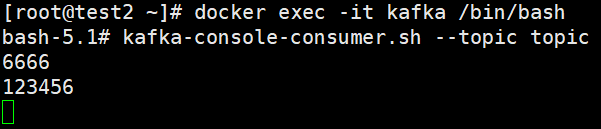
**在 Kafka 容器的命令行界面中，执行以下命令来消费 "topic" 主题的消息：**

kafka-console-consumer.sh --topic topic --bootstrap-server localhost:9094-from-beginning

**生产消息**：



**消费消息**：



# 搭建zookeeper集群

**创建 macvlan 创建自定义网络**

docker network create -d macvlan --subnet=192.168.88.0/24 --gateway=192.168.88.2 -o parent=ens32 zookeeper\_kafka\_network

**创建主节点挂载目录**

# 创建 zookeeper 主节点配置存放目录

mkdir -p /mydata/zookeeper/master/conf

# 创建 zookeeper 主节点数据存放目录

mkdir -p /mydata/zookeeper/master/data

# 创建 zookeeper 主节点数据日志存放目录

mkdir -p /mydata/zookeeper/master/datalog

# 创建 zookeeper 主节点日志存放目录

mkdir -p /mydata/zookeeper/master/logs

**创建从节点1挂载目录**

# 创建 zookeeper 节点1 配置存放目录

mkdir -p /mydata/zookeeper/node1/conf

# 创建 zookeeper 节点1 数据存放目录

mkdir -p /mydata/zookeeper/node1/data

# 创建 zookeeper 节点1 数据日志存放目录

mkdir -p /mydata/zookeeper/node1/datalog

# 创建 zookeeper 节点1 日志存放目录

mkdir -p /mydata/zookeeper/node1/logs

**创建从节点2挂载目录**

# 创建 zookeeper 节点2 配置存放目录

mkdir -p /mydata/zookeeper/node2/conf

# 创建 zookeeper 节点2 数据存放目录

mkdir -p /mydata/zookeeper/node2/data

# 创建 zookeeper 节点2 数据日志存放目录

mkdir -p /mydata/zookeeper/node2/datalog

# 创建 zookeeper 节点2 日志存放目录

mkdir -p /mydata/zookeeper/node2/logs

**创建主配置文件**

# zookeeper 主节点配置存放目录

cd /mydata/zookeeper/master/conf

# 编辑配置文件

vim zoo.cfg

dataDir=/data

dataLogDir=/datalog

tickTime=2000

initLimit=5

syncLimit=2

autopurge.snapRetainCount=3

autopurge.purgeInterval=0

maxClientCnxns=60

clientPort=2181

server.1=192.168.88.21:2888:3888

server.2=192.168.88.22:2888:3888

server.3=192.168.88.23:2888:3888

**把zoo.cfg配置文件拷贝到其他目录下**

cp zoo.cfg /mydata/zookeeper/node1/conf

cp zoo.cfg /mydata/zookeeper/node2/conf

**启动主节点**

docker run -d --restart always \

--name zookeeper\_master \

--network zookeeper\_kafka\_network \

--ip 192.168.88.21 \

-p 2181:2181 \

-e ZOO\_MY\_ID=1 \

-v /mydata/zookeeper/master/conf/zoo.cfg:/opt/zookeeper-3.4.13/conf/zoo.cfg \

-v /mydata/zookeeper/master/data:/data \

-v /mydata/zookeeper/master/datalog:/datalog \

-v /mydata/zookeeper/master/logs:/logs \

wurstmeister/zookeeper

**启动从节点1**

docker run -d --restart always \

--name zookeeper\_node1 \

--network zookeeper\_kafka\_network \

--ip 192.168.88.22 \

-p 2182:2181 \

-e ZOO\_MY\_ID=2 \

-v /mydata/zookeeper/node1/conf/zoo.cfg:/opt/zookeeper-3.4.13/conf/zoo.cfg \

-v /mydata/zookeeper/node1/data:/data \

-v /mydata/zookeeper/node1/datalog:/datalog \

-v /mydata/zookeeper/node1/logs:/logs \

wurstmeister/zookeeper

**启动从节点2**

docker run -d --restart always \

--name zookeeper\_node2 \

--network zookeeper\_kafka\_network \

--ip 192.168.88.23 \

-p 2183:2181 \

-e ZOO\_MY\_ID=3 \

-v /mydata/zookeeper/node2/conf/zoo.cfg:/opt/zookeeper-3.4.13/conf/zoo.cfg \

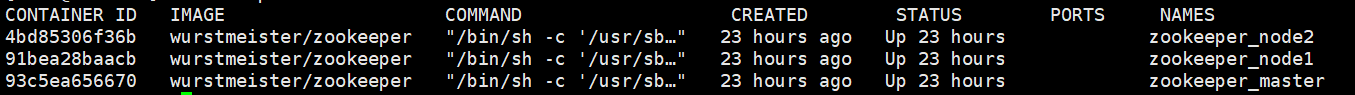
-v /mydata/zookeeper/node2/data:/data \

-v /mydata/zookeeper/node2/datalog:/datalog \

-v /mydata/zookeeper/node2/logs:/logs \

wurstmeister/zookeeper

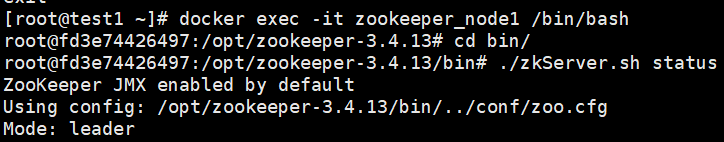
**查看容器状态**



**进入容器查看zookeeper状态**

docker exec -it zookeeper\_node1 /bin/bash

**查看状态**

./bin/zkServer.sh status

状态为leader

为了测试zookeeper集群是否搭建成功，把node1挂掉查看是否集群是否会自动选举一个新的leader

**停止node1**

docker stop zookeeper\_node1

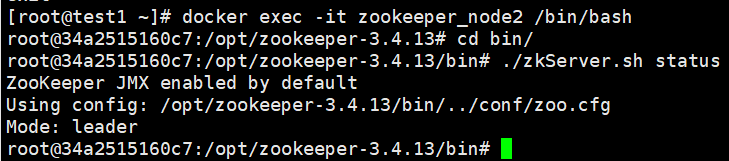
**进入另一个zookeeper容器里面**

docker exec -it zookeeper\_node2 /bin/bash

**进入bin目录**

cd bin

**查看状态**

./bin/zkServer.sh status

发现集群自动选举了一个leader，说明zookeeper集群搭建成功

# 搭建kafka集群

**创建 macvlan 创建自定义网络**

docker network create -d macvlan --subnet=192.168.88.0/24 --gateway=192.168.88.2 -o parent=ens32 zookeeper\_kafka\_network

**创建kafka节点1**

docker run -d \

--name kafka1 \

--network zookeeper\_kafka\_network \

--ip 192.168.88.24 \

-p 9092:9092 \

-e KAFKA\_BROKER\_ID=1 \

-e KAFKA\_ZOOKEEPER\_CONNECT=192.168.88.21:2181,192.168.88.22:2181,192.168.88.23:2181 \

-e KAFKA\_LISTENERS=PLAINTEXT://0.0.0.0:9092 \

-e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://192.168.88.24:9092 \

wurstmeister/kafka

**创建kafka节点2**

docker run -d \

--name kafka2 \

--network zookeeper\_kafka\_network \

--ip 192.168.88.25 \

-p 9093:9093 \

-e KAFKA\_BROKER\_ID=2 \

-e KAFKA\_ZOOKEEPER\_CONNECT=192.168.88.21:2181,192.168.88.22:2181,192.168.88.23:2181 \

-e KAFKA\_LISTENERS=PLAINTEXT://0.0.0.0:9093 \

-e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://192.168.88.25:9093 \

wurstmeister/kafka

**创建kafka节点3**

docker run -d \

--name kafka3 \

--network zookeeper\_kafka\_network \

--ip 192.168.88.26 \

-p 9094:9094 \

-e KAFKA\_BROKER\_ID=3 \

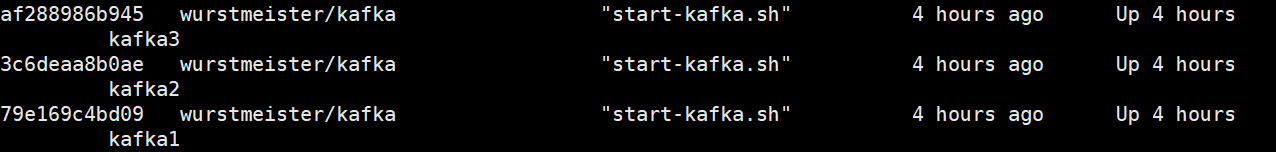
-e KAFKA\_ZOOKEEPER\_CONNECT=192.168.88.21:2181,192.168.88.22:2181,192.168.88.23:2181 \

-e KAFKA\_LISTENERS=PLAINTEXT://0.0.0.0:9094 \

-e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://192.168.88.26:9094 \

wurstmeister/kafka

查看状态

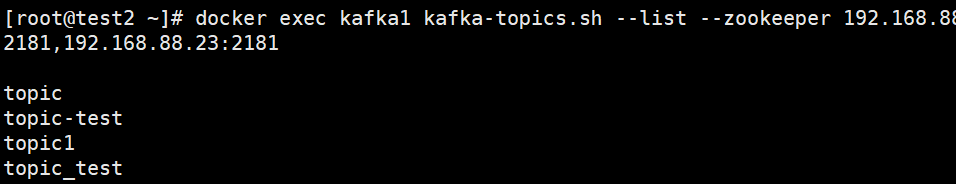


**创建topic主题**

docker exec kafka1 kafka-topics.sh --topic topic-test --create --zookeeper 192.168.88.21:2181,192.168.88.22:2181,192.168.88.23:2181 --replication-factor 3 --partitions 3

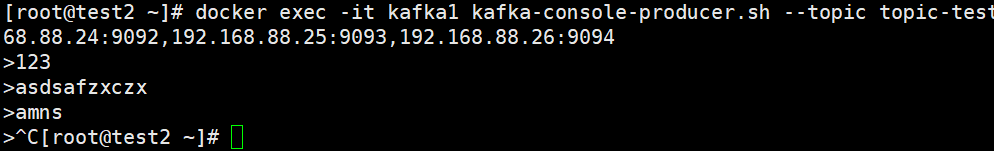
**查看已创建的主题**

docker exec kafka1 kafka-topics.sh --list --zookeeper 192.168.88.21:2181,192.168.88.22:2181,192.168.88.23:2181



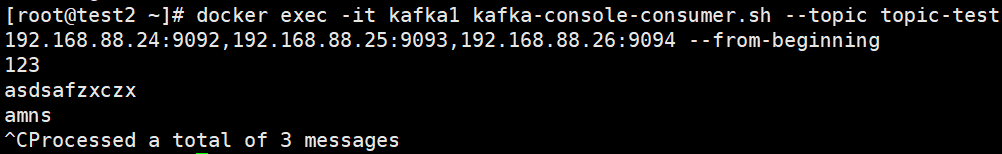
**创建生产者消息**

docker exec -it kafka1 kafka-console-producer.sh --topic topic-test --broker-list 192.168.88.24:9092,192.168.88.25:9093,192.168.88.26:9094



**创建消费者消息**

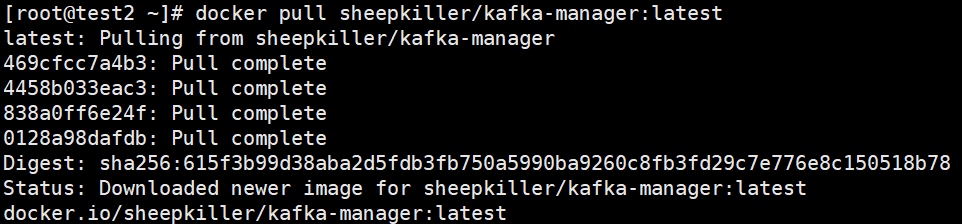
docker exec -it kafka1 kafka-console-consumer.sh --topic topic-test --bootstrap-server 192.168.88.24:9092,192.168.88.25:9093,192.168.88.26:9094 --from-beginning



# 部署kafka-manager

**拉取镜像**

docker pull sheepkiller/kafka-manager:latest



**创建kafka-manager容器**

docker run -it -d \

--name kafka-manager \

--network zookeeper\_kafka\_network \

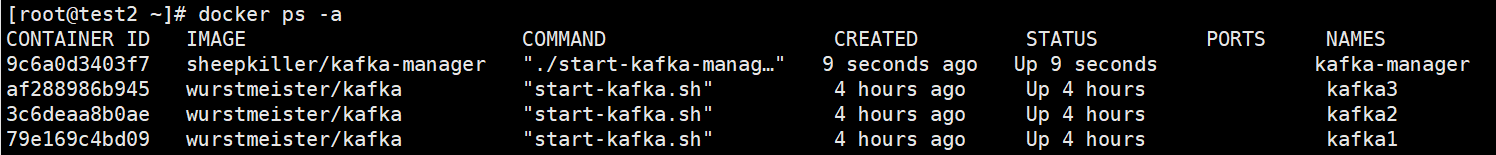
--ip 192.168.88.30 \

-p 9000:9000 \

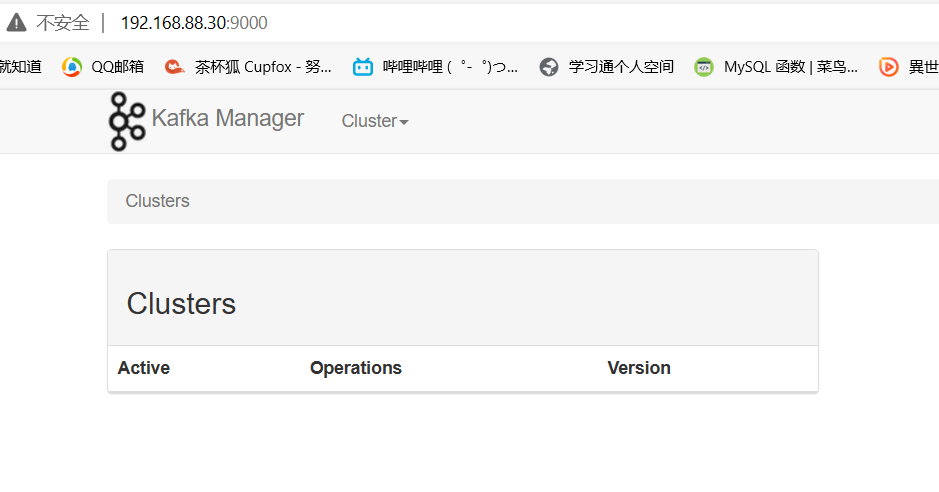
-e ZK\_HOSTS="192.168.88.21:2181,192.168.88.22:2181,192.168.88.23:2181" \

sheepkiller/kafka-manager

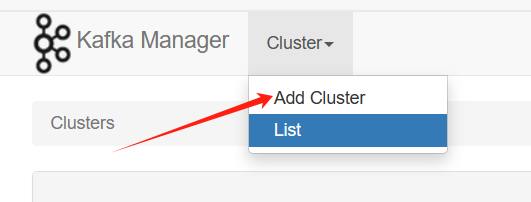
**查看容器状态**



**浏览器访问kafka-manager**



点击Add Cluster 添加集群

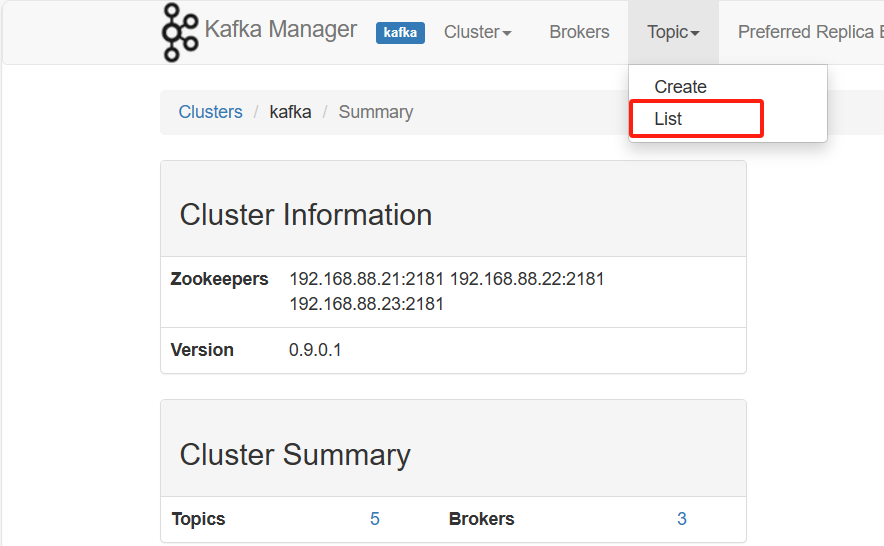






设置完点击save进行保存

查看列表



查看topics主题

