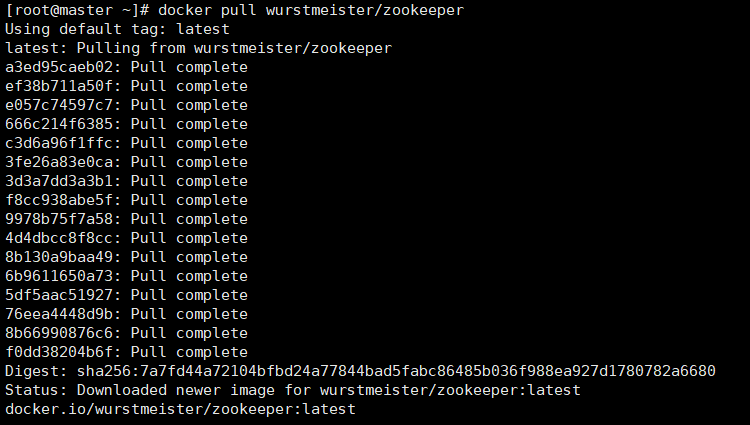
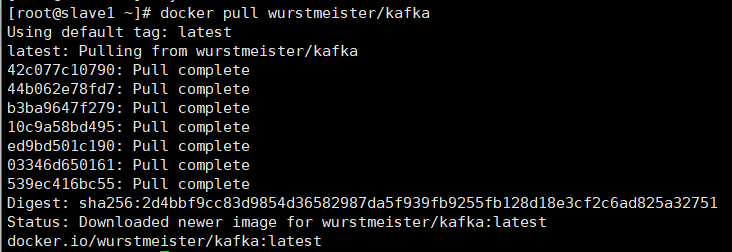
# 实现两台虚拟机zookeeper、kafka应用测试

## 拉取镜像

docker pull wurstmeister/zookeeper 拉取zookeeper镜像

docker pull wurstmeister/kafka 拉取kafka镜像





## 创建zookeeper容器

docker create -p 2181:2181 --name zookeeper --privileged=true \

-v /home/data/zookeeper/data/:/var/lib/zookeeper \

-v /home/data/zookeeper/log:/datalog \

--network my-macvlan-net --ip 192.168.9.10 \

zookeeper

## 创建kafka容器

docker run -d --name kafka --publish 9092:9092 \

--env KAFKA\_ZOOKEEPER\_CONNECT=192.168.28.10:2181 \

--env KAFKA\_ADVERTISED\_HOST\_NAME=192.168.28.11 \

--env KAFKA\_ADVERTISED\_PORT=9092 \

--env KAFKA\_LOG\_DIRS=/kafka/kafka-logs-1 \

-v /usr/local/kafka/data/:/var/lib/zookeeper \

-v /usr/local/kafka/logs:/kafka/kafka-logs-1 \

--network my-macvlan --ip 192.168.28.11 \

wurstmeister/kafka

参数解释

·--env KAFKA\_ZOOKEEPER\_CONNECT：指定 ZooKeeper 的地址和端口。

·指定 Kafka 的广（advertised）主机名或 IP 地址。这是用于在生产者和消费 者之间传递正确的连接信息。

·--env KAFKA\_ADVERTISED\_PORT：指定Kafka 的广告端口号。

·--env KAFKA\_LOG\_DIRS：设置 Kafka 的日志目录路径。

·-v /usr/local/kafka/data/:/var/lib/zookeeper：将主机的目录挂载到容器的目录， 用于持久化存储 ZooKeeper 数据。

·-v /usr/local/kafka/logs:/kafka/kafka-logs-1：将主机的目录挂载到容器的目录， 用于持久化存储 Kafka 的日志文件。

·--network my-macvlan-net：将容器连接到名为 "my-macvlan-net" 的自定义网 络。

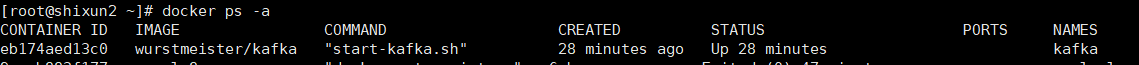
·--ip 192.168.9.9：为容器指定 IP 地址。

## 启动容器

命令：docker start kafka

命令：docker start zookeeper

容器状态为UP表示容器启动成功



容器状态为UP表示容器启动成功



## 创建查看 Topic

创建一个名为 test 的 Topic，该 Topic 包含一个分区和一个 Replica。

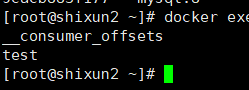
docker exec kafka kafka-topics.sh --create --zookeeper 192.168.9.10:2181 --replication-factor 1 --partitions 1 --topic test

·--replication-factor 1：指定了主题的副本因子。副本因子定义了每个分区的副 本数量。在这里，副本因子为 1，表示每个分区只有一个副本。

·--partitions 1：指定了主题的分区数。分区是 Kafka 中消息的逻辑单元，它允 许消息在集群中进行并行处理。在这里，主题只有一个分区。

创建后可以执行如下命令查看当前的 Topics：

docker exec kafka kafka-topics.sh --list --zookeeper 192.168.9.10:2181



Test为使用命令创建的容器，可以进行进行消息的生产和消费操作。

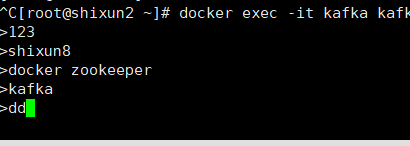
\_\_consumer\_offsets 是 Kafka 内部使用的特殊主题，用于存储消费者组的偏移量信息。

## 创建消息

注意：如果事先没有使用 kafka-topics 命令来手工创建 Topic，直接使用下面的内容进行消息创建时也会自动创建 Topics。

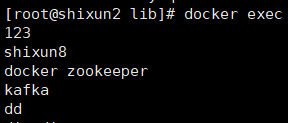
首先创建消息生产者。执行如下命令启动 Kafka 基于命令行的消息生产客户端，启动后可以直接在控制台中输入消息来发送，控制台中的每一行数据都会被视为一条消息来发送。

docker exec -it kafka kafka-console-producer.sh --broker-list 192.168.9.9:9092 --topic test



接着创建消息消费者。我们打开另一个命令窗口执行如下执行命令启动 Kafka 基于命令行的消息消费客户端，启动之后，马上可以在控制台中看到输出了之前我们在消息生产客户端中发送的消息。

docker exec -it kafka kafka-console-consumer.sh --bootstrap-server 192.168.9.9:9092 --topic test --from-beginning



# 搭建zookeeper集群

**创建 macvlan 创建自定义网络**

docker network create -d macvlan --subnet=192.168.28.0/24 --gateway=192.168.28.2 -o parent=ens33 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 192.168.28.21 \

-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 192.168.28.22 \

-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 192.168.28.23 \

-p 2183:2181 \

-e ZOO\_MY\_ID=3 \

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

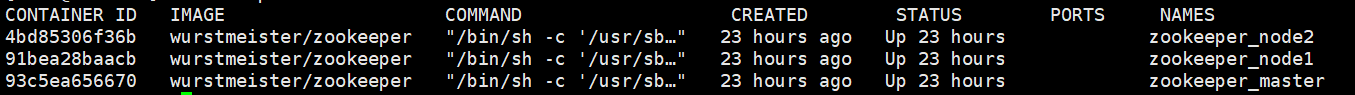
-v /mydata/zookeeper/node2/data:/data \

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

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

wurstmeister/zookeeper

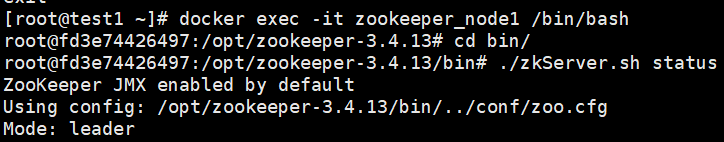
**查看容器状态**



**进入容器查看zookeeper状态**

docker exec -it zookeeper\_node1 /bin/bash

**查看状态**

./bin/zkServer.sh status

# 搭建kafka集群

**创建 macvlan 创建自定义网络**

docker network create -d macvlan --subnet=192.168.28.0/24 --gateway=192.168.28.2 -o parent=ens33 zookeeper\_kafka\_network

**创建kafka节点1**

docker run -d \

--name kafka1 \

--network zookeeper\_kafka\_network \

--ip 192.168.28.24 \

-p 9092:9092 \

-e KAFKA\_BROKER\_ID=1 \

-e KAFKA\_ZOOKEEPER\_CONNECT=192.168.28.21:2181,192.168.28.22:2181,192.168.28.23:2181 \

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

-e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://192.168.28.24:9092 \

wurstmeister/kafka

**创建kafka节点2**

docker run -d \

--name kafka2 \

--network zookeeper\_kafka\_network \

--ip 192.168.28.25 \

-p 9093:9093 \

-e KAFKA\_BROKER\_ID=2 \

-e KAFKA\_ZOOKEEPER\_CONNECT=192.168.28.21:2181,192.168.28.22:2181,192.168.28.23:2181 \

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

-e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://192.168.28.25:9093 \

wurstmeister/kafka

**创建kafka节点3**

docker run -d \

--name kafka3 \

--network zookeeper\_kafka\_network \

--ip 192.168.28.26 \

-p 9094:9094 \

-e KAFKA\_BROKER\_ID=3 \

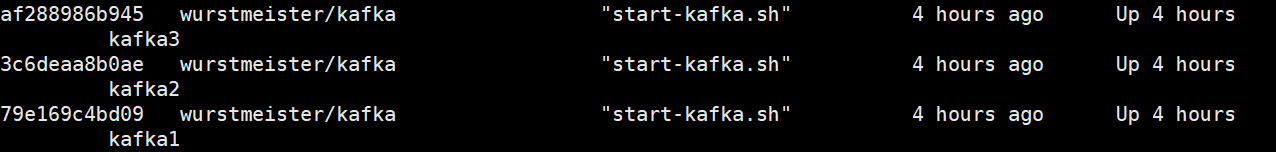
-e KAFKA\_ZOOKEEPER\_CONNECT=192.168.28.21:2181,192.168.28.22:2181,192.168.28.23:2181 \

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

-e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://192.168.28.26:9094 \

wurstmeister/kafka

查看状态

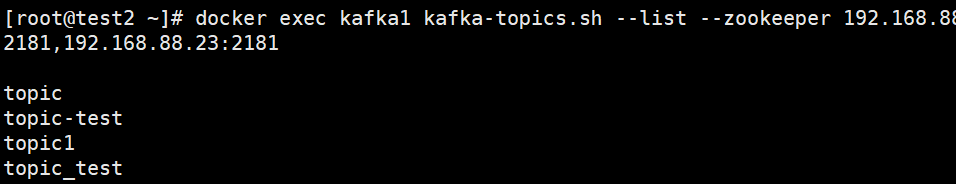


**创建topic主题**

docker exec kafka1 kafka-topics.sh --topic topic-test --create --zookeeper 192.168.28.21:2181,192.168.28.22:2181,192.168.28.23:2181 --replication-factor 3 --partitions 3

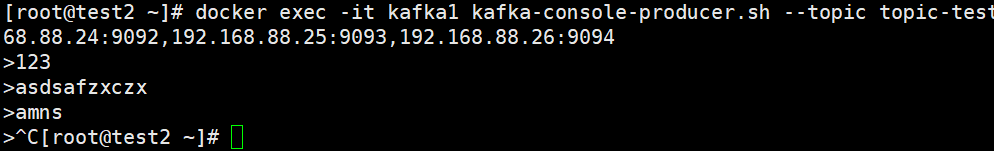
**查看已创建的主题**

docker exec kafka1 kafka-topics.sh --list --zookeeper 192.168.28.21:2181,192.168.28.22:2181,192.168.28.23:2181



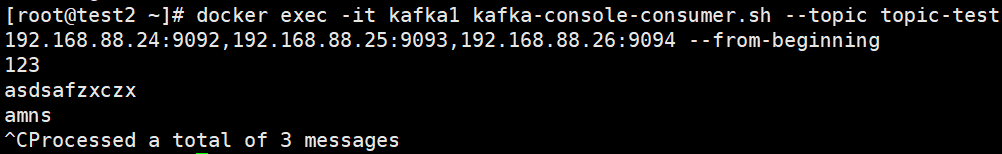
**创建生产者消息**

docker exec -it kafka1 kafka-console-producer.sh --topic topic-test --broker-list 192.168.28.24:9092,192.168.28.25:9093,192.168.28.26:9094



**创建消费者消息**

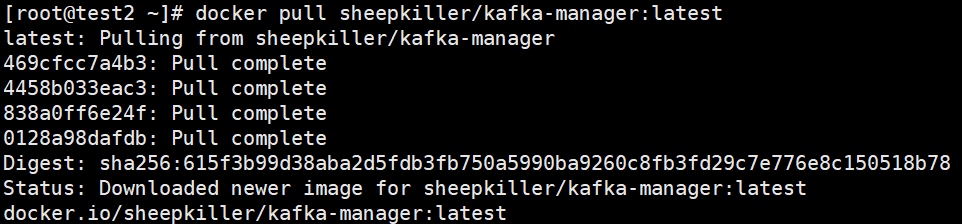
docker exec -it kafka1 kafka-console-consumer.sh --topic topic-test --bootstrap-server 192.168.28.24:9092,192.168.28.25:9093,192.168.28.26:9094 --from-beginning



# 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 \

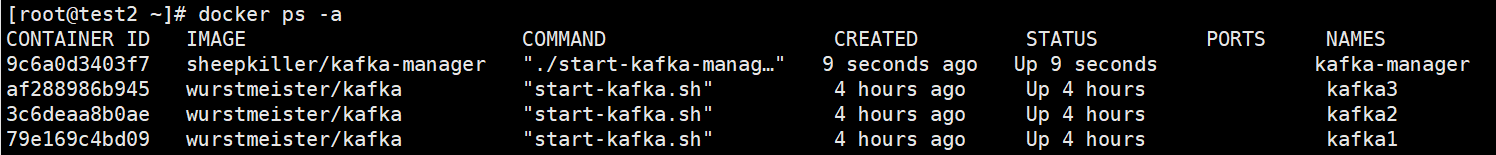
--ip 192.168.28.30 \

-p 9000:9000 \

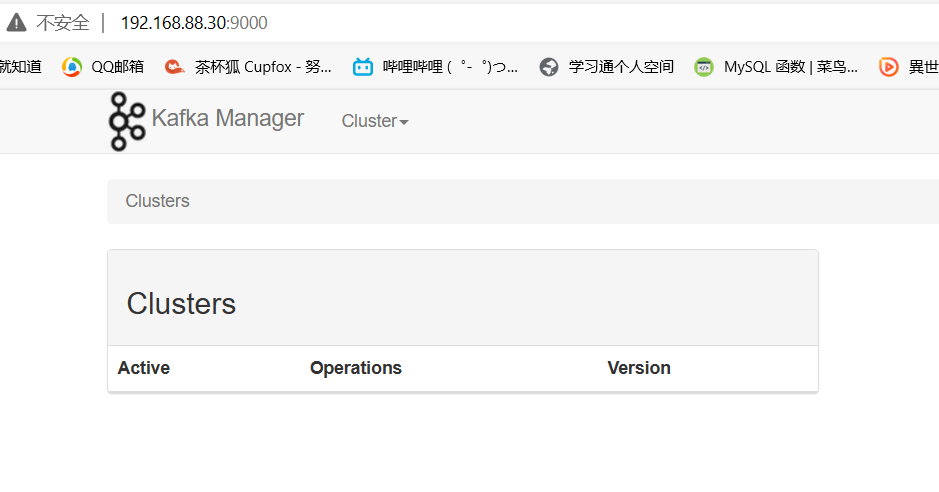
-e ZK\_HOSTS="192.168.28.21:2181,192.168.28.22:2181,192.168.28.23:2181" \

sheepkiller/kafka-manager

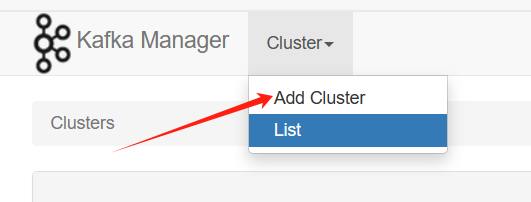
**查看容器状态**



**浏览器访问kafka-manager**



点击Add Cluster 添加集群

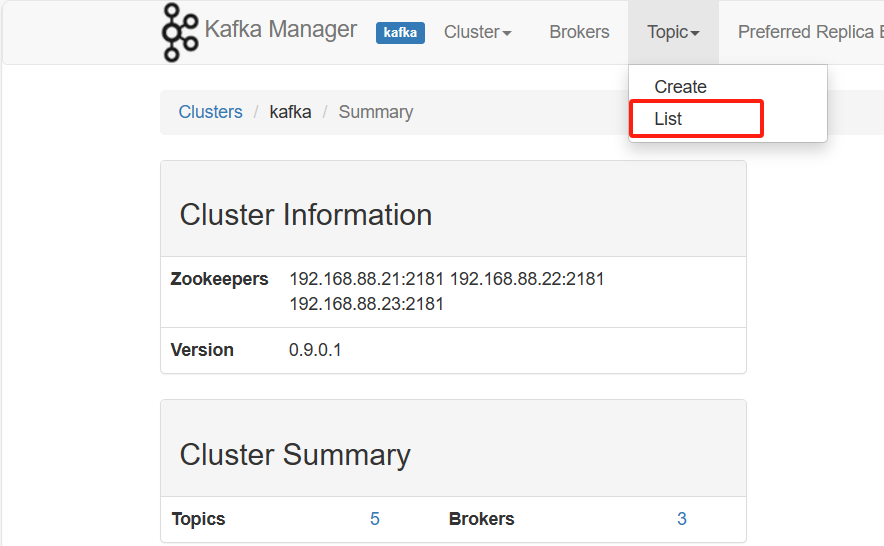






设置完点击save进行保存

查看列表



查看topics主题

