Kafka实验

1.创建集群

注意选择流式集群以及勾选Kafka组件即可，并绑定公网IP。

2.安装客户端

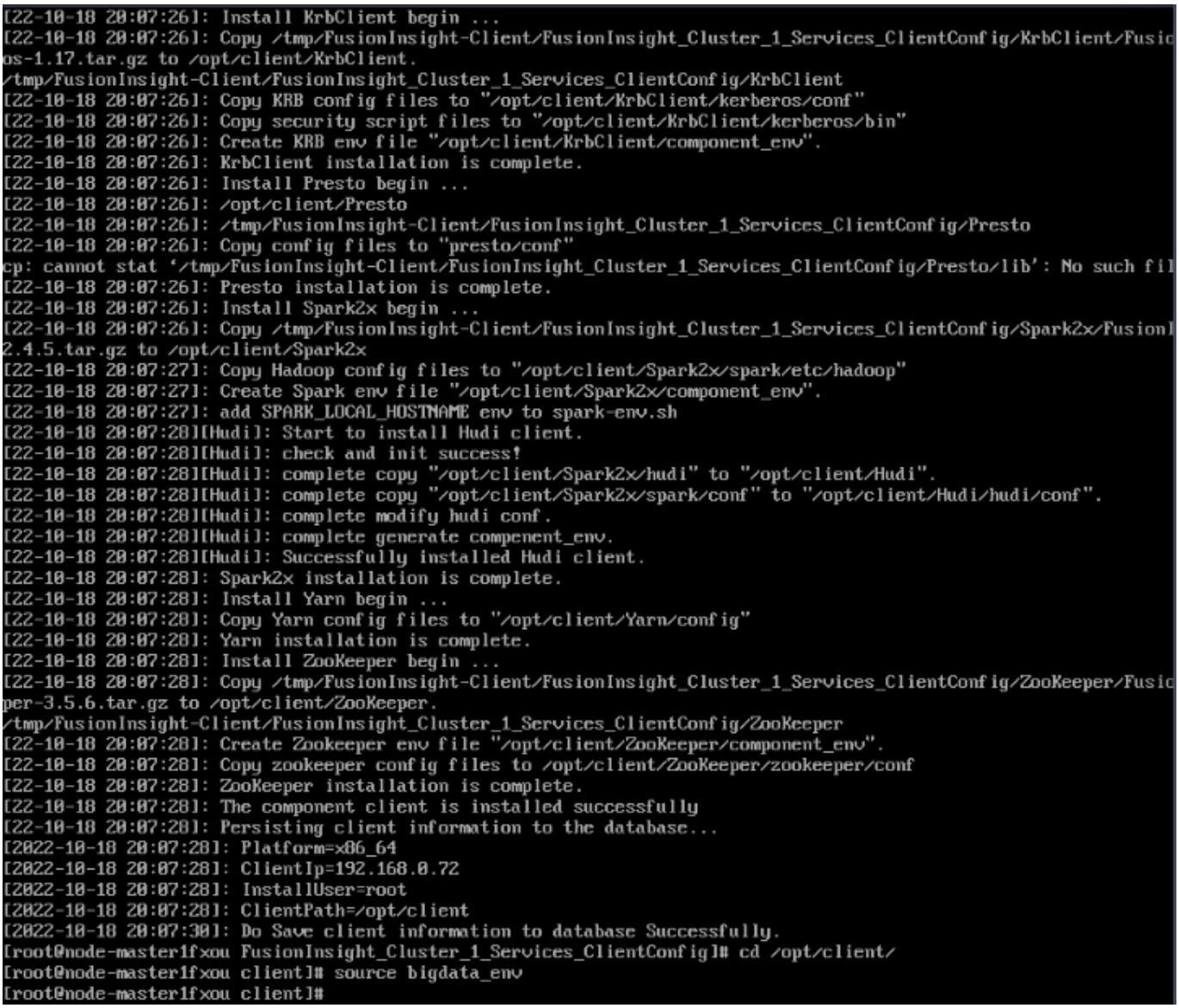
进入Manager界面

登录对应的服务器（192.168.0.3），安装客户端。

输入指令：

```
tar -xvf FusionInsight_Cluster_1_Services_Client.tar

sha256sum -c FusionInsight_Cluster_1_Services_ClientConfig.tar.sha256

tar -xvf FusionInsight_Cluster_1_Services_ClientConfig.tar

cd /opt/Bigdata/client/FusionInsight_Cluster_1_Services_ClientConfig

./install.sh /opt/hadoopclient #安装到/opt/hadoopclient目录

cd /opt/hadoopclient

source bigdata_env
```

最终效果：



3. 登录Master节点

在“集群列表 > 现有集群”列表中，单击名称“Kafka_test”，在“节点管理”页签中找到类型为“Master1”的节点，并单击其名称，跳转至云服务器控制台上的该弹性云服务器详情页面。

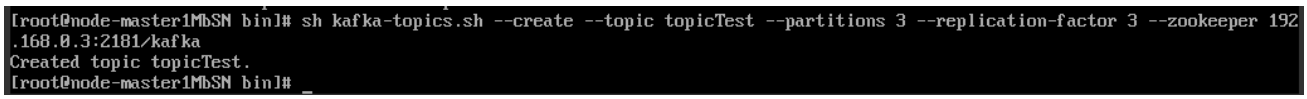
4.使用Kafka客户端创建topic

1. 在“集群列表 > 现有集群”列表中，单击名称“Kafka_test”，进入集群“概览”页面。
   在“概览”页面单击“IAM用户同步”后的“同步”等待同步完成。

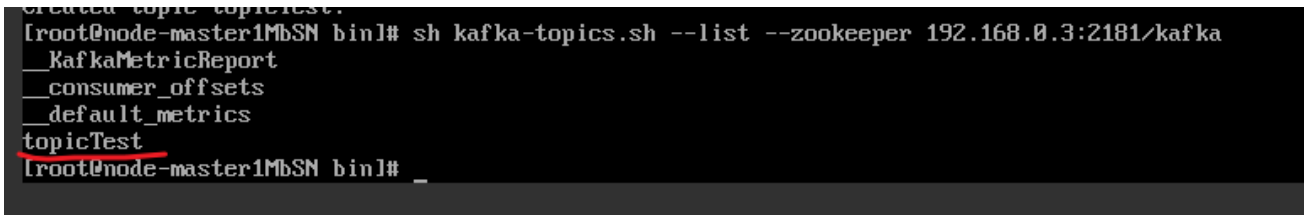
2. 选择“组件管理 > ZooKeeper > 实例”，查看ZooKeeper角色实例的IP地址。 记录ZooKeeper角色实例中任意一个的IP地址即可。这里记下`192.168.0.3`。

3. 执行如下命令，创建kafka topic。
   kafka-topics.sh --create --zookeeper <ZooKeeper角色实例所在节点IP:2181/kafka> --partitions 3 --replication-factor 3 --topic <Topic名称>

   相应的，我们输入`sh kafka-topics.sh --create --zookeeper 192.168.0.3:2181/kafka --partitions 3 --replication-factor 3 --topic topicTest`



   输入`sh kafka-topics.sh --list --zookeeper 192.168.0.3:2181/kafka`查看当前topic



5.管理Kafka主题中的消息

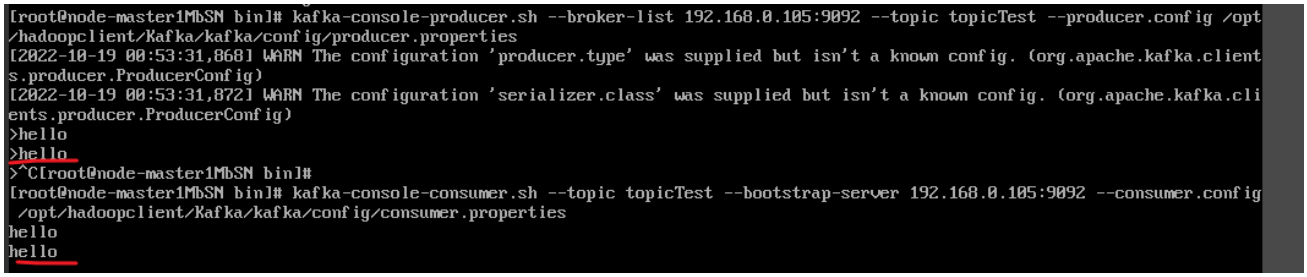
1. 选择“组件管理 > Kafka > 实例”，查看Kafka角色实例的IP地址。记录Kafka角色实例中任意一个的IP地址。如`192.168.0.105`。

2. 登录Master节点，在topic test中产生消息。 首先执行命令`kafka-console-producer.sh --broker-list <Kafka角色实例所在节点IP:9092> --topic <Topic名称> --producer.config /opt/hadoopclient/Kafka/kafka/config/producer.properties`其中<Topic名称>为Step7创建的Topic名称。 然后输入指定的内容作为生产者产生的消息，输入完成后按回车发送消息。如果需要结束产生消息，使用“Ctrl + C”退出任务。

输入`kafka-console-producer.sh --broker-list 192.168.0.105:9092 --topic topicTest --producer.config /opt/hadoopclient/Kafka/kafka/config/producer.properties`

3. 消费topic test中的消息。 输入`kafka-console-consumer.sh --topic topicTest --bootstrap-server 192.168.0.105:9092 --consumer.config /opt/hadoopclient/Kafka/kafka/config/consumer.properties`

   结果：



6.Python使用Kafka

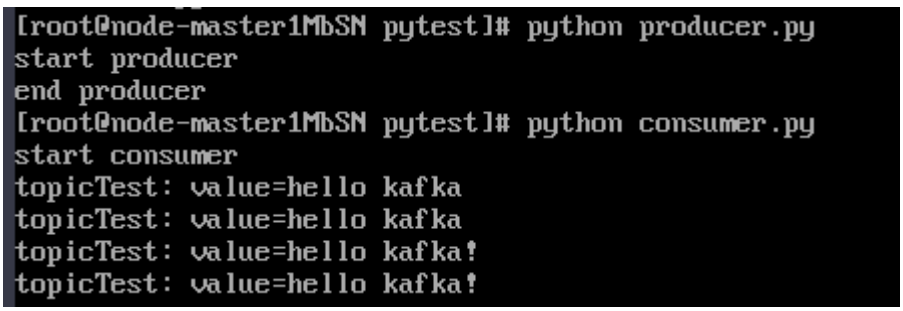
准备Python环境，导入kafka-python包（这里使用2.0.1版本），注意绑定公网IP下载。

导入producer.py以及consumer.py文件，注意相应的本机信息。其中`consumer_id`可查找`/opt/hadoopclient/Kafka/kafka/config/consumer.properties`获取。

```
#!/usr/bin/python3

from kafka import KafkaProducer

conf = {
    'bootstrap_servers': ["192.168.0.105:9092"],
    'topic_name': 'topicTest',
}

print('start producer')
producer = KafkaProducer(bootstrap_servers=conf['bootstrap_servers'])

data = bytes("hello kafka!", encoding="utf-8")
producer.send(conf['topic_name'], data)
producer.close()
print('end producer')
```

```
#!/usr/bin/python3

from kafka import KafkaConsumer

conf = {
    'bootstrap_servers':["192.168.0.105:9092"],
    'topic_name': 'topicTest',
    'consumer_id': 'example-group1'
}

print('start consumer')
consumer = KafkaConsumer(conf['topic_name'],
                         bootstrap_servers=conf['bootstrap_servers'],
                         group_id=conf['consumer_id'])

for message in consumer:
    print("%s: value=%s" % (message.topic, message.value))

print('end consumer')y运行producer.py
```

依次运行producer.py和consumer.py，结果如图：



附：Kafka学习笔记

1.介绍

Kafka是Apache旗下的一款分布式流媒体平台，Kafka是一种高吞吐量、持久性、分布式的发布订阅的消息队列系统。 它最初由LinkedIn(领英)公司发布，使用Scala语言编写，与2010年12月份开源，成为Apache的顶级子项目。 它主要用于处理消费者规模网站中的所有动作流数据。动作指(网页浏览、搜索和其它用户行动所产生的数据)。

活动流数据是几乎所有站点在对其网站使用情况做报表时都要用到的数据中最常规的部分。活动数据包括页面访问量（Page View）、被查看内容方面的信息以及搜索情况等内容。这种数据通常的处理方式是先把各种活动以日志的形式写入某种文件，然后周期性地对这些文件进行统计分

析。**运营数据**指的是服务器的性能数据（CPU、IO 使用率、请求时间、服务日志等等数据)。运营数据的统计方法种类繁多。

Kafka三大特点：

1.**高吞吐量**：可以满足每秒百万级别消息的生产和消费。

2.**持久性**：有一套完善的消息存储机制，确保数据高效安全且持久化。

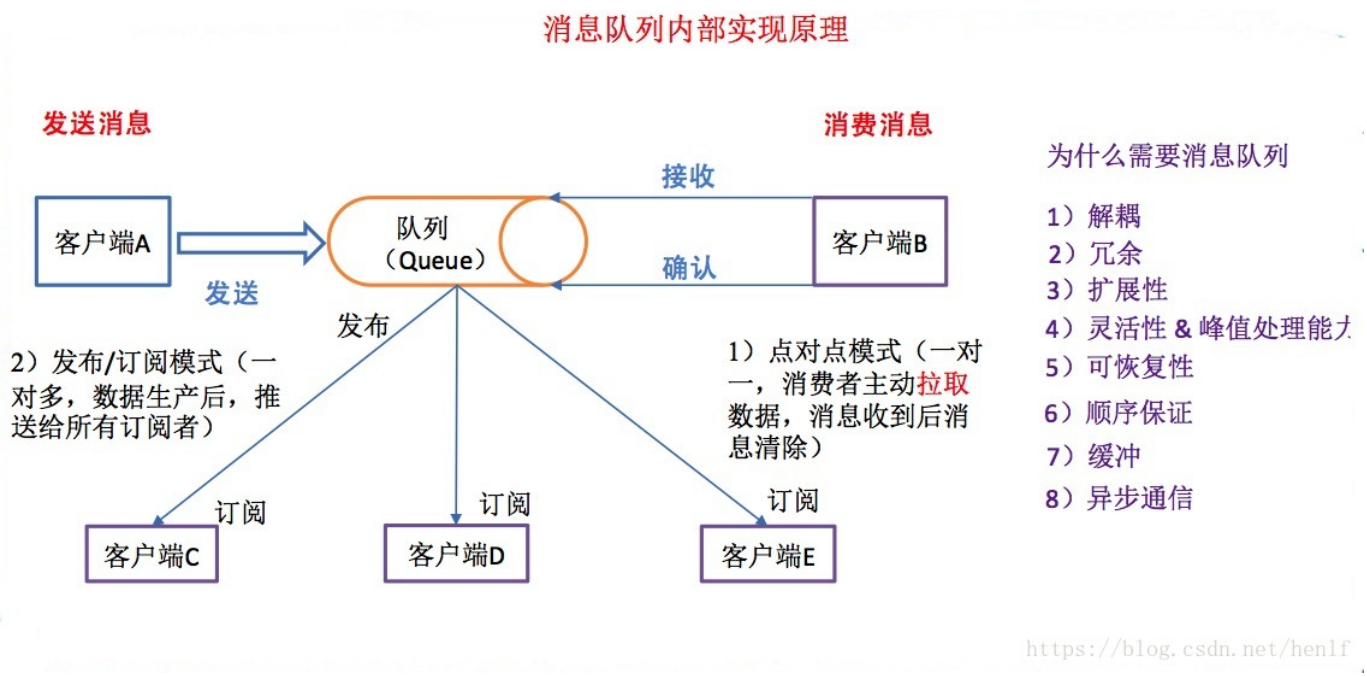
3.**分布式**：基于分布式的扩展；Kafka的数据都会复制到几台服务器上，当某台故障失效时，生产者和消费者转而使用其它的Kafka。

1.关键概念

- topic kafka 把收到的消息按 topic 进行分类，因此可以理解为 topic 是一种类别
- producer 往 kafka 发送消息的用户
- consumer 接收 kafka 消息的用户
- borker kafka 集群可以由多个 kafka 实例组成，每个实例（server）称为 broker

无论是 kafka broker 本身，还是 producer 或者 consumer，都依赖于 zookeeper 集群保存一些 meta 信息，保证系统可用性，以及使用 zookeeper 的选举机制。

2.消息队列实现原理



点对点模式 一对一，**消费者主动拉取数据，消息收到后消息清除。**点对点模式通常是一个基于拉取或轮询的消息发送模型。此模型中，消费者从队列主动拉取信息，而不是消息系统推送消息给消费者，并且，**消息只能被一个且只有一个消费者接收处理**，即使有多个消息监听者也是如此。
发布订阅模式 一对多，数据生产后，**推送给所有订阅者。**发布订阅模型则是一个**基于推送的**消息传送模型。发布订阅模型可以有多种不同的订阅者，临时订阅者只在主动监听主题时才接收消息，而持久订阅者则监听主题的所有消息，即使当前订阅者不可用，处于离线状态。