目录

目录 1
为集群添加SSH密钥 2
使用SSH访问集群 2
Kafka集群公网访问 3
  模式一：使用SDK模式收发消息 3
    绑定公网IP 3
    开放安全组端口 3
    示例：操作指南 3
  模式二：本地客户端命令行形式 4
    在本地安装kafka 4
    绑定公网IP 4
    开放安全组端口 4
    示例：操作指南 5

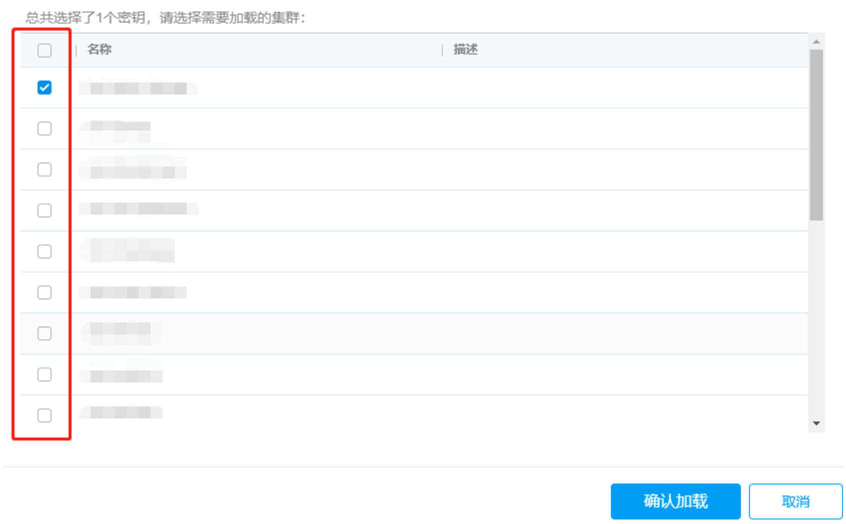
为集群添加SSH密钥

KMR 集群仅支持 SSH 密钥认证方式，在使用 SSH 之前，您需要为集群添加SSH 密钥。

1. 使用密钥生成工具生成 SSH-2 RSA 密钥，保存好私钥和公钥。
    - windows用户可以使用 PuTTYgen.exe 工具，下载可参阅Download PuTTY。
    - Linux 用户可以通过 ssh-keygen－t rsa 来生成，默认生成在~/.ssh/目录下，公钥文件是~/.ssh/id_rsa.pub，用户使用 ssh-keygen 时也可以自己指定公钥目录。

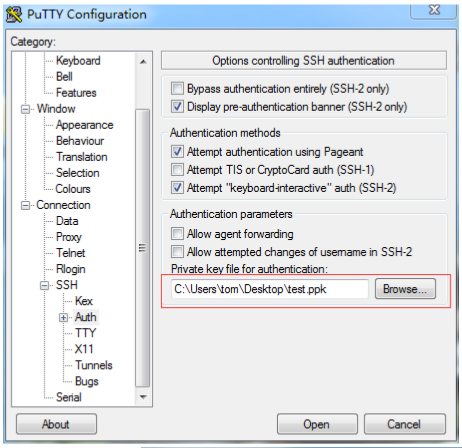
2. 打开KMR控制台，点击SSH 密钥。

3. 点击**创建密钥**，把第1步生成的公钥文件内容粘贴到对话框，并点击**添加**。
4. 选择刚刚创建好的密钥，点击**加载到集群**，把密钥加载到集群，您可以把密钥同时加载到多个集群。至此，密钥加载完成。

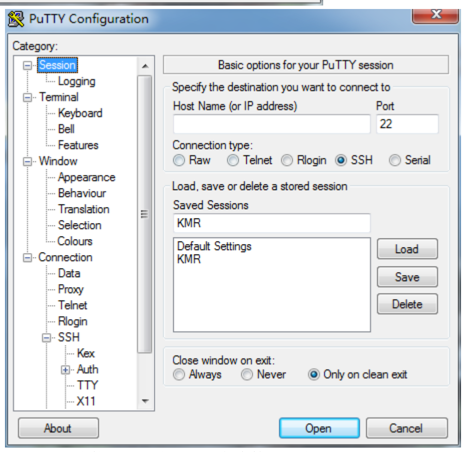


使用SSH访问集群

1.打开KMR控制台，进入集群详情，展开主节点的详细信息，您可以通过公网（如果绑定EIP）或者内网IP地址访问集群。



2. 导入3.2.1为集群添加SSH密钥中产生的私钥。



3. 指定IP和端口连接KMR主节点，连接到集群，登陆账户名是root。
4. 主节点和核心节点已配置了SSH互信，可以在控制台查看核心节点的IP地址，直接从主节点登录到各个核心节点。

Kafka集群公网访问

本文介绍Kafka集群公网访问的两种模式。

模式一：使用SDK模式收发消息

绑定公网IP

将集群所有节点绑定公网IP，具体操作是在云服务控制台绑定公网IP，在云服务器控制台根据主机名搜索所有节点，设置方法详见设置弹性IP。

开放安全组端口

选择创建集群时所选的安全组，并在该安全组内配置外网访问入站的ACL。开放对应客户端外网IP的访问，端口为6667。

示例：操作指南

1. 在客户端主机上配置/etc/hosts文件 需要将master1上的/etc/hosts文件内容添加至客户端主机上，并将每个ip替换成公网IP地址。

```
# kafka
120.92.33.214 kmr-0ba43848-gn-73607e20-master-1-001.ksc.com kmr-0ba43848-gn-73607e20-master-1-001
120.92.93.108 kmr-0ba43848-gn-73607e20-core-1-001.ksc.com kmr-0ba43848-gn-73607e20-core-1-001
120.131.13.117 kmr-0ba43848-gn-73607e20-core-1-002.ksc.com kmr-0ba43848-gn-73607e20-core-1-002
```

2. 添加maven依赖。

```
// 消息队列 Kafka 版服务端版本为 0.10.1.0，建议的客户端版本为 0.10.2.2
<dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka-clients</artifactId>
<version>0.10.2.2</version>
</dependency>
```

3.客户端添加配置文件。

```
##配置节点列表，即控制台中集群详情页中，节点配置项所有节点的内网ip列表，如120.92.33.214:6667,120.92.93.108:6667,120.131.13.117:6667
bootstrap.servers=xxxxxxxxxxxxxxxxxxxxx
##配置 Topic，可以在Kafka
topic=alikafka-topic-demo
##配置 Consumer Group
group.id=group-demo
```

加载配置文件：

```
public class JavaKafkaConfigurer {
    private static Properties properties;
    public synchronized static Properties getKafkaProperties() {
       if (null != properties) {
           return properties;
       }
       //获取配置文件 kafka.properties 的内容
       Properties kafkaProperties = new Properties();
       try {
           kafkaProperties.load(KafkaProducerDemo.class.getClassLoader().getResourceAsStream("kafka.properties"));
       } catch (Exception e) {
           // 没加载到文件，程序要考虑退出
           e.printStackTrace();
       }
       properties = kafkaProperties;
       return kafkaProperties;
    }
}
```

4.使用Java SDK发送消息。

```
    // 加载 kafka.properties
    Properties kafkaProperties =  JavaKafkaConfigurer.getKafkaProperties();
    Properties props = new Properties();props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,       kafkaProperties.getProperty("bootstrap.servers"));
    // 接入协议
    props.put(CommonClientConfigs.SECURITY_PROTOCOL_CONFIG, "PLAINTEXT");
    // Kafka 消息的序列化方式
    props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, "org.apache.kafka.common.serialization.StringSerializer");
    props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, "org.apache.kafka.common.serialization.StringSerializer");
    // 请求的最长等待时间
    props.put(ProducerConfig.MAX_BLOCK_MS_CONFIG, 30 * 1000);
    // 构造 Producer 对象，注意，该对象是线程安全的，一般来说，一个进程内一个 Producer 对象即可；// 如果想提高性能，可以多构造几个对象，但不要太多，最好不要超过 5 个
    KafkaProducer<String, String> producer = new KafkaProducer<String, String>(props);
    // 构造一个 Kafka 消息
    String topic = kafkaProperties.getProperty("topic");
    //消息所属的 Topic，请在控制台申请之后，填写在这里String value = "this is the message's value";
    //消息的内容
    ProducerRecord<String, String>  kafkaMessage =  new ProducerRecord<String, String>(topic, value);try {
      // 发送消息，并获得一个 Future 对象
      Future<RecordMetadata> metadataFuture = producer.send(kafkaMessage);
      // 同步获得 Future 对象的结果
      RecordMetadata recordMetadata = metadataFuture.get();
      System.out.println("Produce ok:" + recordMetadata.toString());
    } catch (Exception e) {
      // 要考虑重试
      System.out.println("error occurred");
      e.printStackTrace();
    }
```

5.使用Java SDK订阅消息。

```
//加载kafka.properties
    Properties kafkaProperties =  JavaKafkaConfigurer.getKafkaProperties();
    Properties props = new Properties();
    //设置server
    props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, kafkaProperties.getProperty("bootstrap.servers"));
    //两次poll之间的最大允许间隔
    //请不要改得太大，服务器会掐掉空闲连接，不要超过30000
    props.put(ConsumerConfig.SESSION_TIMEOUT_MS_CONFIG, 25000);
    //每次poll的最大数量
    //注意该值不要改得太大，如果poll太多数据，而不能在下次poll之前消费完，则会触发一次负载均衡，产生卡顿
    props.put(ConsumerConfig.MAX_POLL_RECORDS_CONFIG, 30);
    //消息的反序列化方式
    props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, "org.apache.kafka.common.serialization.StringDeserializer");
    props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, "org.apache.kafka.common.serialization.StringDeserializer");

    //属于同一个组的消费实例，会负载消费消息
    props.put(ConsumerConfig.GROUP_ID_CONFIG, kafkaProperties.getProperty("group.id"));

    //构造消息对象，也即生成一个消费实例
    KafkaConsumer<String, String> consumer = new org.apache.kafka.clients.consumer.KafkaConsumer<String, String>(props);
    //设置消费组订阅的Topic，可以订阅多个
    //如果GROUP_ID_CONFIG是一样，则订阅的Topic也建议设置成一样
    List<String> subscribedTopics =  new ArrayList<String>();
    //如果需要订阅多个Topic，则在这里add进去即可
    //每个Topic需要先在控制台进行创建
    subscribedTopics.add(kafkaProperties.getProperty("topic"));
    consumer.subscribe(subscribedTopics);
    //循环消费消息
    while (true){
        try {

            ConsumerRecords<String, String> records = consumer.poll(1000);
            //必须在下次poll之前消费完这些数据, 且总耗时不得超过SESSION_TIMEOUT_MS_CONFIG
            //建议开一个单独的线程池来消费消息，然后异步返回结果
            for (ConsumerRecord<String, String> record : records) {
                System.out.println(String.format("Consume partition:%d offset:%d", record.partition(), record.offset()));
            }
        } catch (Exception e) {
            try {
                Thread.sleep(1000);
            } catch (Throwable ignore) {
            }
            e.printStackTrace();
        }
    }
```

模式二：本地客户端命令行形式

在本地安装kafka

需要安装kafka集群一致版本的kafka。

绑定公网IP

参照使用SDK模式收发消息绑定公网IP。

开放安全组端口

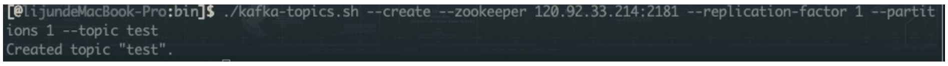
1.选择创建集群时所选的安全组，并在该安全组内配置外网访问入站的ACL。开放对应客户端外网IP的访问，端口为2181、6667。

2. 验证所有节点绑定公网IP的端口号是否通：telnet 120.92.33.214 2181。

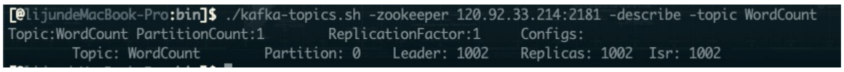
示例：操作指南

1. 查看有哪些主题： ./kafka-topics.sh --list --zookeeper zk服务IP:2181。

2. 创建topic：./kafka-topics.sh --create --zookeeper zk服务IP:2181 --replication-factor 1 --partitions 1 --topic WordCount。



3. 查看topic的详细信息：./kafka-topics.sh -zookeeper zk服务IP:2181 -describe -topic WordCount。



4. kafka生产者客户端命令：./kafka-console-producer.sh --broker-list kafka服务IP:6667 --topic WordCount。

5. kafka消费者客户端命令：./kafka-console-consumer.sh --bootstrap-server 120.92.33.214:6667 --from-beginning --topic WordCount。