阿里云
消息队列 Apache Kafka 版

快速入门

文档版本：20191012

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. **您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。**
2. **未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。**
3. **由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。**
4. **本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。**
5. **阿里云文档中所有内容，包括但不限于图片、架构设计、页面布局、文字描述，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。**
6. **如若发现本文档存在任何错误，请与阿里云取得直接联系。**

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告:** 重启操作将导致业务中断，恢复业务时间约十分钟。
	用于警示信息、补充说明等，是用户必须了解的内容。	 **注意:** 权重设置为0，该服务器不会再接受新请求。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明:** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	单击设置 > 网络 > 设置网络类型。
粗体	表示按键、菜单、页面名称等UI元素。	在结果确认页面，单击**确定**。
`Courier字体`	命令。	执行`cd /d C:/window`命令，进入Windows系统文件夹。
#####	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{}或者{a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1 概述

本文旨在以简单明了的方式引导您快速上手消息队列 for Apache Kafka。

操作步骤

具体操作步骤如下：

1. **步骤一：获取访问授权**
2. **步骤二：购买和部署实例**
   - *VPC* **接入**
   - **公网** + *VPC* **接入**
3. **步骤三：创建资源**
4. **步骤四：使用 SDK 收发消息**
   - *VPC* **接入**
   - **公网接入**

2 步骤一：获取访问授权

开通消息队列 for Apache Kafka 服务需要使用阿里云其他云产品中的资源，因此，您需要先授权消息队列 for Apache Kafka 访问您所拥有的其他阿里云资源。

背景信息

为了方便您快速开通服务，消息队列 for Apache Kafka 为您的账号创建了一个默认 RAM 角色，用于快捷访问阿里云资源。

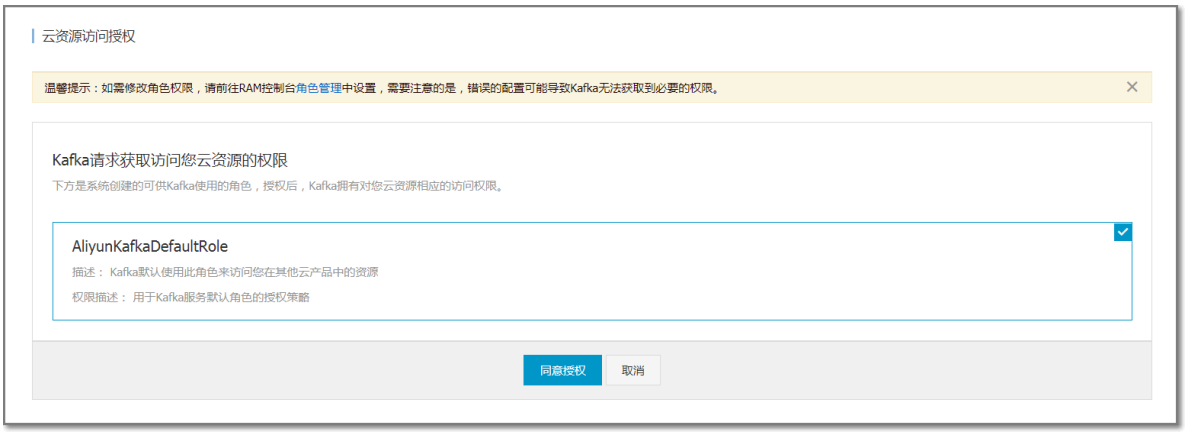
> **说明:**
> RAM 是阿里云的访问控制系统，您可以通过创建 RAM 用户和角色，并为其配置相应的权限来管控您的资源访问。关于 RAM 的更多信息，请参见 *#unique_10*。

前提条件

- 您已注册阿里云账号并完成实名认证。

操作步骤

1. 登录阿里云官网。
2. 进入云资源访问授权页面，单击同意授权。



同意授权后，页面跳转到消息队列 for Apache Kafka 的控制台。

3 步骤二：购买和部署实例

3.1 VPC 接入

本文介绍如何使用 VPC 网络接入消息队列 for Apache Kafka 服务。

前提条件

您已搭建了自己的阿里云 VPC 网络。

操作步骤

test

1. 购买实例

a. 登录消息队列 *for Apache Kafka* 控制台。

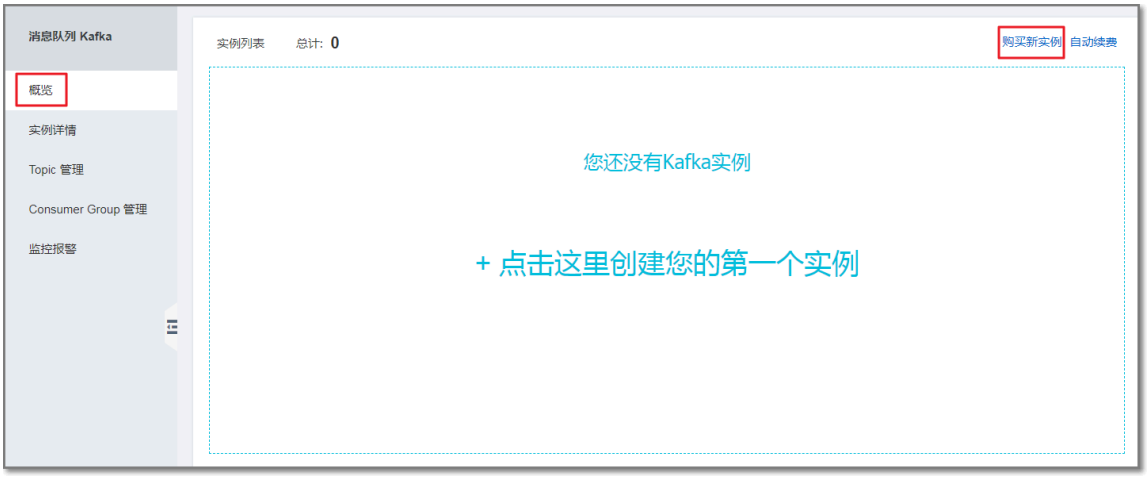
b. 在控制台的顶部导航栏，选择您将要购买和部署实例的地域（下图举例为华东1（杭州），您可以根据需求进行切换）。

注意:

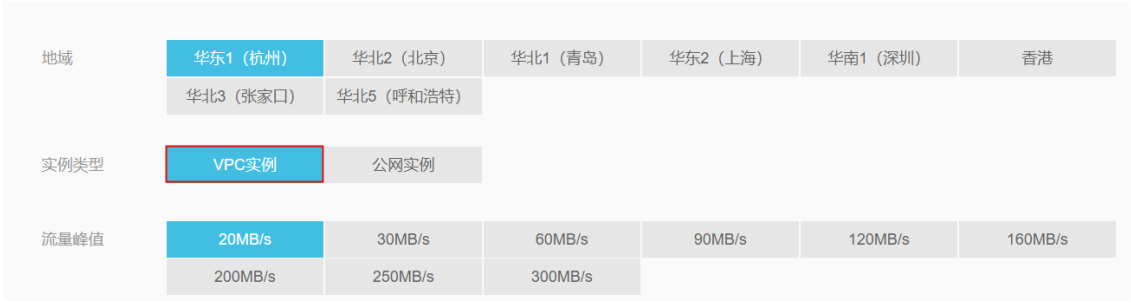
消息队列 for Apache Kafka 实例不支持跨地域部署，即您需在购买实例的地域进行部署。

c. **在消息队列 for Apache Kafka 控制台，单击左侧导航栏中的概览。**

d. **在实例列表页，单击购买新实例。**



e. **在实例购买页，根据自身业务需求选择相应的配置。注意，在实例类型一栏，选择VPC实例类型。**



f. 单击页面右侧的立即购买，按照提示完成购买流程。

2. 获取 VPC 信息

a. 登录 *VPC* 管理控制台。在左侧导航栏，单击交换机。

b. 在交换机页面，查看以下信息：

- VSwitch ID
- VPC ID
- 可用区

注意：

请根据该页面的可用区（A～G）在消息队列 for Apache Kafka 控制台中选择对应的可用区（A～G）。例如，某 VPC 交换机（VSwitch）显示在可用区E，那么在消息队列 for Apache Kafka 控制台中就相应地选择可用区E。

3. 部署实例

a. 在消息队列 for Apache Kafka 控制台，单击左侧导航栏的概览，查看已购买的实例。

b. 选择处于待部署状态的实例，单击右侧的部署，然后根据页面提示填写在步骤二中已获取的 VPC 信息。

完成后，实例会进入部署中状态。实例部署预计需要 10 分钟 ~ 30 分钟。

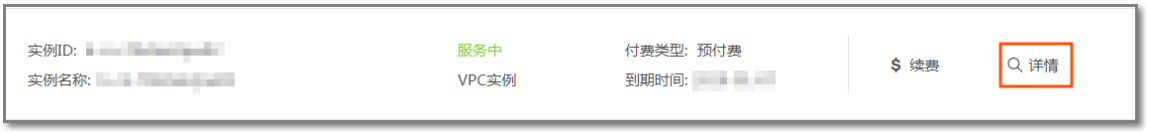
注意：

多接入点和重新设置用户名密码这两项功能，不建议在纯 VPC 场景中使用，请保持默认否。

c. 刷新控制台页面。实例状态显示服务中，即表示已部署成功。

4. 查看实例详情

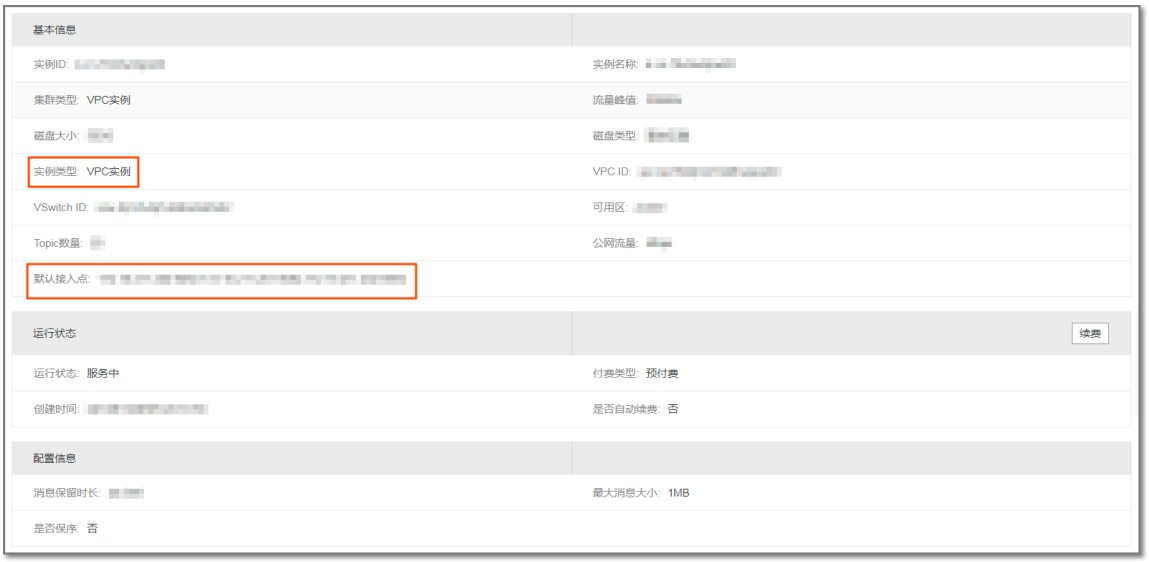
a. 实例部署成功后，单击该实例右侧的详情，查看实例详情。



说明:

您也可在左侧导航栏单击实例详情，然后直接在页面上方选择要查看的实例，单击查看其详情。

b. 在实例详情页，查看该实例的默认接入点。从 VPC 内接入消息队列 for Apache Kafka 服务，需使用默认接入点接入。



3.2 公网 + VPC 接入

本文介绍如何同时使用 VPC 网络和公网网络接入消息队列 for Apache Kafka 服务。在此场景下，您只需购买公网实例，采用公网和 VPC 的方式部署即可。

前提条件

您已搭建了自己的阿里云 VPC 网络。

操作步骤

1. **购买实例**

   a. **登录消息队列 *for Apache Kafka* 控制台。**

   b. **在控制台的顶部导航栏，选择您将要购买和部署实例的地域（下图举例为华东1（杭州），您可以根据需求进行切换）。**

   c. **在消息队列 for Apache Kafka 控制台，单击左侧导航栏中的概览。**

   d. **在实例列表页，单击购买新实例。**

   e. **在实例购买页，根据自身业务需求选择相应的配置。注意，在实例类型一栏，选择公网/VPC实例类型。**

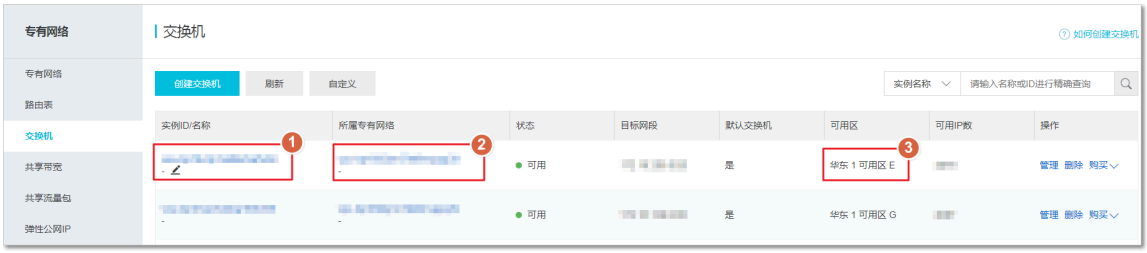
f. 单击页面右侧的立即购买，按照提示完成购买流程。

2. 获取 VPC 信息

a. 登录 *VPC* 管理控制台。在左侧导航栏，单击交换机。

b. 在交换机页面，查看以下信息：

- VSwitch ID
- VPC ID
- 可用区



注意：

请根据该页面的可用区（A～G）在消息队列 for Apache Kafka 控制台中选择对应的可用区（A～G）。例如，某 VPC 交换机（VSwitch）显示在可用区E，那么在消息队列 for Apache Kafka 控制台中就相应地选择可用区E。

3. 部署实例

a. 在消息队列 for Apache Kafka 控制台，单击左侧导航栏的概览，查看已购买的实例。

b. 选择处于待部署状态的实例，单击右侧的部署。

c. 在部署对话框中：

A. 在 VPC ID一栏，选择 VPC 的 ID。

B. 在 VSwitch ID 一栏，填写在步骤二中获取的 VPC 的交换机 ID。

C. 在重新设置用户名密码一栏：

- 否：若无多实例共享用户名和密码的需求，选择该项，即使用系统为每个实例分配的默认用户名和密码。
- 是：若您需要多实例共享同一用户名和密码，选择该项，然后自定义用户名和密码。

完成后，实例会进入部署中状态。实例部署预计需要 10~30 分钟。

d. 刷新控制台页面。实例状态显示服务中，即表示已部署成功。

u

4. 查看实例详情

a. 实例部署成功后，实例类型显示为公网/VPC实例。单击该实例右侧的详情，查看实例详情。

说明:

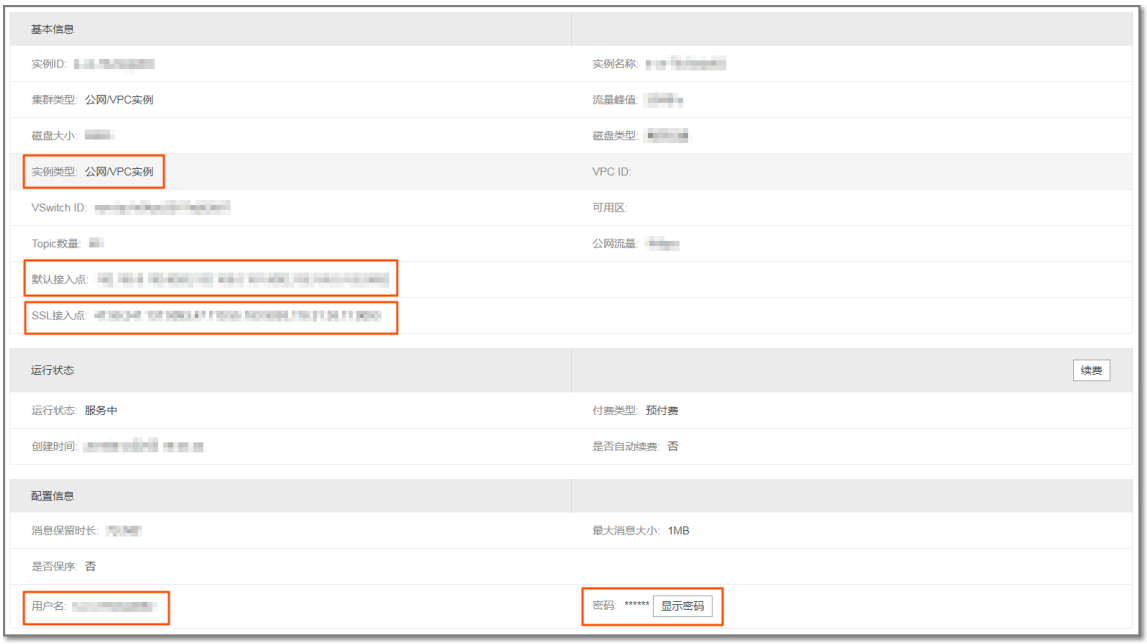
您也可在左侧导航栏单击实例详情，然后直接在页面上方选择要查看的实例，单击查看其详情。

b. 在实例详情页：

A. 查看该实例的默认接入点和SSL接入点 。

- **从 VPC 内接入消息队列 for Apache Kafka 服务，需使用默认接入点。**
- **从公网接入消息队列 for Apache Kafka 服务，需使用 SSL接入点。**

B. 若之前部署实例时重新设置了用户名和密码，查看重设的用户名和密码。



4 步骤三：创建资源

在使用消息队列 for Apache Kafka 进行消息收发之前，您需要先在消息队列 for Apache Kafka 控制台上创建资源，否则将无法通过鉴权认证及使用相关的管控运维功能。这里的资源概念是指 Topic 和 Consumer Group。

步骤一：创建 Topic

消息主题（Topic）是消息队列 for Apache Kafka 里对消息进行的一级归类，比如可以创建“Topic_Trade”这一主题用来识别交易类消息。使用消息队列 for Apache Kafka 的第一步就是先为您的应用创建 Topic。

请按照以下步骤创建 Topic：

1. **登录消息队列 *for Apache Kafka* 控制台。在顶部导航栏，选择地域，如华东1（杭州）。**
2. **在左侧导航栏，单击 Topic管理。**
3. **在 Topic管理页面，单击创建Topic 按钮。**

4. 填写的 Topic 信息与付费模式和实例版本相关：

   - 包年包月预付费模式下的标准版实例或是按量后付费实例：

     在 创建Topic 页面，输入 Topic 名称，选择 Topic 所在的实例，再输入备注和分区数，然后单击创建。

   - 包年包月预付费模式下的专业版实例：

     在创建Topic 页面，输入 Topic 名称，选择 Topic 所在的实例，再输入备注和分区数；单击高级设置，选择存储引擎和消息类型，然后单击创建。

     高级设置说明如下：

配置项	说明
存储引擎	目前，消息队列 for Apache Kafka 有以下两种存储引擎： - 云存储：底层接入阿里云云盘的多副本能力，在 Kafka 层面，每个分区只需要 1 个副本。如需 30 MB/s 峰值流量，900 GB 磁盘容量，则购买 30 MB/s 峰值流量，900 GB 磁盘容量即可。 - Local存储：使用原生 Kafka 的 ISR 复制算法，3 副本，且 `min.insync.replicas` = 2。如需 30 MB/s 峰值流量，900 GB 磁盘容量，由于 3 副本，则需购买 90 MB/s 峰值流量，2700 GB 磁盘容量。Topic 存储类型的更多信息请参见 *#unique_12*。 **注意:** 若需使用 Local存储，请**提交工单**

配置项	说明
cleanup.policy	当您选择了 Local存储时，才需配置日志清理策略（cleanup.policy）。目前，消息队列 for Apache Kafka 有以下两种日志清理策略： - delete：默认的消息清理策略。在磁盘容量充足的情况下，保留在最长保留时间范围内的消息；在磁盘容量不足时（一般磁盘使用率超过 85% 视为不足），将提前删除旧消息，以保证服务可用性。 - compact：使用 *Kafka Log Compaction* 日志清理策略。Log Compaction 清理策略保证相同 Key 的消息，最新的 value 值一定会被保留。主要适用于系统宕机后恢复状态，系统重启后重新加载缓存等场景。例如，在使用 Kafka Connect 或 Confluent Schema Registry 时，需要使用 Kafka Compact Topic 存储系统状态信息或配置信息。 注意： Compact Topic 一般只用在某些生态组件中，比如 Kafka Connect 或 Confluent Schema Registry，其他情况的消息收发请勿为 Topic 设置该属性。具体详情请仔细阅读各生态组件的 *Demo* 文档
消息类型	目前，消息队列 for Apache Kafka 有以下两种消息类型： - 普通消息：默认情况下，保证相同 Key 的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，可能会造成消息乱序。 - 分区顺序消息：默认情况下，保证相同 Key 的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，仍然保证分区内按照发送顺序存储。但是会出现部分分区发送消息失败，等到分区恢复后即可恢复正常。

完成后，您创建的 Topic 将出现在 Topic管理页面的列表中。

> **说明:**
>
> - Topic 也需要在应用程序所在的地域（即所部署的 ECS 的所在地域）进行创建。
> - Topic 不能跨地域使用。比如 Topic 创建在华北2（北京）这个地域，那么消息生产端和消费端也必须运行在华北2（北京）的 ECS 上。
> - 地域的详细介绍请参见 ECS 文档中的*#unique_13*

步骤二：创建 Consumer Group

创建完 Topic 后，请按以下步骤创建 Consumer Group：

1. 在消息队列 for Apache Kafka 控制台的左侧导航栏，单击 Consumer Group管理。
2. 在 Consumer Group 管理页面，单击 创建Consumer Group 按钮。
3. 在 创建Consumer Group 页面，填写 Consumer Group 的名称，并选择该 Consumer Group 所在的实例，然后单击 创建。

完成后，您创建的 Consumer Group 将出现在 Consumer Group管理页面的列表中。

> **注意:**
>
> Consumer Group 和 Topic 的关系是 N：N。 同一个 Consumer Group 可以订阅多个 Topic，同一个 Topic 也可以被多个 Consumer Group 订阅。

5 步骤四：使用 SDK 收发消息

5.1 VPC 接入

本文将通过 Java SDK 示例向您展示如何通过消息队列 for Apache Kafka 的 SDK 在 VPC 环境收发消息。

步骤一：准备配置

添加 Maven 依赖的示例代码如下：

1. 添加 Maven 依赖

添加 Maven 依赖的示例代码如下：

```
// 消息队列 for Apache Kafka 服务端版本为 0.10.0.0，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>
```

2. 客户端配置

a. 准备配置文件：kafka.properties

请参考以下示例代码进行修改：

```
## 配置接入点，即控制台的实例详情页显示的默认接入点
bootstrap.servers=xxxxxxxxxxxxxxxxxxxx
## 配置 Topic，可以在控制台上创建 Topic
topic=alikafka-topic-demo
## 配置 Consumer Group，可以在控制台创建 Consumer Group
group.id=CID-consumer-group-demo
```

b. 加载配置文件

请参考以下示例代码进行修改：

```
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.getClassLo
ader().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;
    }
}
```

步骤二：使用 Java SDK 发送消息

说明:

本节已 Java SDK 为例，关于其他语言示例，请参见消息队列 *for Apache Kafka Demo* 库。

示例代码如下：

```
// 加载 kafka.properties
Properties kafkaProperties =  JavaKafkaConfigurer.getKafkaProperties
();
Properties props = new Properties();
// 设置接入点，即控制台的实例详情页显示的“默认接入点”
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();
}
```

步骤三：使用 Java SDK 订阅消息

说明:

本节已 Java SDK 为例，关于其他语言示例，请参见消息队列 *for Apache Kafka Demo* 库。

示例代码如下：

```
        //加载kafka.properties

         Properties kafkaProperties =  JavaKafkaConfigurer.getKafkaPr
operties();

         Properties props = new Properties();

         //设置接入点，即控制台的实例详情页显示的“默认接入点”

         props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, kafkaPrope
rties.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");

         //当前消费实例所属的消费组，请在控制台申请之后填写

         //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) {
                }
                //参考常见报错：https://help.aliyun.com/knowledge_detail
/124136.html
                e.printStackTrace();
            }
        }
```

5.2 公网接入

本文将通过 Java SDK 示例向您展示如何通过消息队列 for Apache Kafka 的 SDK 在公网环境收发消息。

更多详情，请参见消息队列 *for Apache Kafka Demo* (公网版) 。

步骤一：准备配置

1. 添加 Maven 依赖

添加 Maven 依赖的示例代码如下：

```
//消息队列 for Apache Kafka 服务端版本为 0.10.0.0，建议客户端版本为
 0.10.2.2
<dependency>
        <groupId>org.apache.kafka</groupId>
        <artifactId>kafka-clients</artifactId>
        <version>0.10.2.2</version>
</dependency>
```

2. SASL 配置

   消息队列 for Apache Kafka 利用 SASL 机制对客户端进行身份验证。

   a. 创建文本文件 kafka_client_jaas.conf

      请参考以下示例代码进行修改：

```
KafkaClient {
  org.apache.kafka.common.security.plain.PlainLoginModule required
  username="xxxxxxxxxxxxxxxxxxxx"
  password="xxxxxxxxxxxxxxxxxxxx";
};
```

       注意：

      其中username和password可以从消息队列 for Apache Kafka 的实例信息中获取。

   b. 设置 kafka_client_jaas.conf 的路径

      kafka_client_jaas.conf 的路径是系统变量，有以下两种办法进行设置（这里假设 kafka_client_jaas.conf 放在/home/admin 下面，实际部署时请注意修改为自己的路径）。

      - 程序启动时，启动 JVM 参数：

        `-Djava.security.auth.login.config=/home/admin/kafka_client_jaas.conf`

      - 或者在代码中设置参数（需要保证在 Kafka Producer 和 Consumer 启动之前）：

        `System.setProperty("java.security.auth.login.config", “/home/admin/kafka_client_jaas.conf");`

3. SSL配置

   下载根证书，下载后放入某个目录下，其路径需要直接配置在代码中。

4. 客户端配置

   a. 准备配置文件：kafka.properties

      请参考以下示例代码进行修改：

```
## 接入点，即控制台的实例详情页显示的 SSL 接入点
bootstrap.servers=xxxxxxxxxxxxxxxxxxxx

## 您在控制台创建的 Topic
topic=alikafka-topic-demo

## 您在控制台创建的 Consumer Group
```

```
group.id=CID-alikafka-consumer-group-demo

## ssl 根证书的路径，demo 中有，请拷贝到自己的某个目录下，不能被打包到 jar
 中
## 这里假设您的目录为/home/admin，请记得修改为自己的实际目录
ssl.truststore.location=/home/admin/kafka.client.truststore.jks

## sasl 路径，demo 中有，请拷贝到自己的某个目录下，不能被打包到 jar 中
## 这里假设您的目录为/home/admin，请记得修改为自己的实际目录
java.security.auth.login.config=/home/admin/kafka_client_jaas.conf
```

b. 加载配置文件

```
private static Properties properties;

public static void configureSasl() {
   //如果用-D 或者其它方式设置过，这里不再设置
   if (null == System.getProperty("java.security.auth.login.config
")) {
       //请注意将 XXX 修改为自己的路径
       //这个路径必须是一个文件系统可读的路径，不能被打包到 jar 中
       System.setProperty("java.security.auth.login.config",
getKafkaProperties().getProperty("java.security.auth.login.config
"));
   }
}

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;
}
```

步骤二：使用 Java SDK 发送消息

示例代码如下：

```
//设置 sasl 文件的路径
JavaKafkaConfigurer.configureSasl();

//加载 kafka.properties
Properties kafkaProperties =  JavaKafkaConfigurer.getKafkaProperties
();

Properties props = new Properties();
//设置接入点，即控制台的实例详情页显示的“SSL接入点”
props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, kafkaProperties.
getProperty("bootstrap.servers"));
//设置 SSL 根证书的路径，请记得将 XXX 修改为自己的路径
```

```
//与 sasl 路径类似，该文件也不能被打包到 jar 中
props.put(SslConfigs.SSL_TRUSTSTORE_LOCATION_CONFIG, kafkaProperties.
getProperty("ssl.truststore.location"));
//根证书 store 的密码，保持不变
props.put(SslConfigs.SSL_TRUSTSTORE_PASSWORD_CONFIG, "KafkaOnsClient
");
//接入协议，目前支持使用 SASL_SSL 协议接入
props.put(CommonClientConfigs.SECURITY_PROTOCOL_CONFIG, "SASL_SSL");
//SASL 鉴权方式，保持不变
props.put(SaslConfigs.SASL_MECHANISM, "PLAIN");
//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(String.format("Produce ok: topic:%s partition:%
d offset:%d value:%s",
            recordMetadata.topic(), recordMetadata.partition(),
recordMetadata.offset(), value));
} catch (Exception e) {
    //要考虑重试
    //参考常见报错：https://help.aliyun.com/document_detail/124136.html
    System.out.println("error occurred");
    e.printStackTrace();
}
```

注意：

如果 kafka-client 的版本为 2.0.0 及以上，需额外设置以下参数：

```
props.put("ssl.endpoint.identification.algorithm", "");
```

步骤三：使用 Java SDK 订阅消息

示例代码如下：

```
//设置 sasl 文件的路径
JavaKafkaConfigurer.configureSasl();

//加载kafka.properties
Properties kafkaProperties =  JavaKafkaConfigurer.getKafkaProperties
();

Properties props = new Properties();
//设置接入点，即控制台的实例详情页显示的“SSL接入点”
props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, kafkaProperties.
getProperty("bootstrap.servers"));
//设置 SSL 根证书的路径，请记得将 XXX 修改为自己的路径
//与 sasl 路径类似，该文件也不能被打包到jar中
props.put(SslConfigs.SSL_TRUSTSTORE_LOCATION_CONFIG, kafkaProperties.
getProperty("ssl.truststore.location"));
//根证书 store 的密码，保持不变
props.put(SslConfigs.SSL_TRUSTSTORE_PASSWORD_CONFIG, "KafkaOnsClient
");
//接入协议，目前支持使用 SASL_SSL 协议接入
props.put(CommonClientConfigs.SECURITY_PROTOCOL_CONFIG, "SASL_SSL");
//SASL 鉴权方式，保持不变
props.put(SaslConfigs.SASL_MECHANISM, "PLAIN");
//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");
//当前消费实例所属的消费组，请在控制台申请之后填写
//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_TI
MEOUT_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 topic:%s
partition:%d offset:%d value:%s",
                    record.topic(), record.partition(), record.offset
(), record.value()));
        }
    } catch (Exception e) {
        try {
            Thread.sleep(1000);
        } catch (Throwable ignore) {

        }
        //参考常见报错：https://help.aliyun.com/knowledge_detail/124136.
html
        e.printStackTrace();
    }
}
```

注意：

如果 kafka-client 的版本为 2.0.0 及以上，需额外设置以下参数：

```
props.put("ssl.endpoint.identification.algorithm", "");
```