ALIBABA CLOUD

Alibaba Cloud

消息队列 Kafka 版

生态对接

文档版本：20201010

阿里云

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. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，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
危险	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。	危险 重置操作将丢失用户配置数据。
警告	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。
注意	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。
说明	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`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	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.生态对接概述

本文介绍消息队列Kafka版支持对接的生态。

开源生态

Logstash

- 接入Logstash

Filebeat

- 接入Filebeat

MySQL

- 使用Kafka Connect同步MySQL

SQL Server

- 使用Kafka Connect同步SQL Server

2.开源生态

2.1. Logstash

2.1.1. 接入Logstash

本文介绍如何将消息队列Kafka版接入Logstash。
kafka logstash

Logstash

Logstash是开源的服务器端数据处理管道，能够同时从多个数据源采集数据，然后对数据进行转换，并将数据写入指定的存储中。Logstash的数据处理流程如下：

1. 输入：采集各种格式、大小和来源的数据。在实际业务中，数据往往以各种各样的形式分散或集中地存储在多个系统中，Logstash支持多种数据输入方式，可以在同一时间从多种数据源采集数据。Logstash能够以连续的流式传输方式从日志、Web应用、数据存储等采集数据。
2. 过滤：实时解析和转换数据。数据从源传输到目标存储的过程中，Logstash过滤器能够解析各个事件，识别已命名的字段来构建结构，并将它们转换成通用格式，通过更轻松、快速的方式分析数据来实现商业价值。
3. 输出：导出数据。Logstash提供多种数据输出方向，灵活解锁众多下游用例。

更多关于Logstash的介绍，请参见Logstash简介。

接入优势

消息队列Kafka版接入Logstash可以带来以下优势：

- 异步处理：提高运行效率，防止突发流量影响用户体验。
- 应用解耦：当应用上下游中有一方存在异常情况，另一方仍能正常运行。
- 减少开销：减少Logstash的资源开销。

接入方案

消息队列Kafka版支持以下方式接入Logstash：

- 作为Input接入
- 作为Output接入

2.1.2. VPC

2.1.2.1. 作为Input接入

消息队列Kafka版可以作为Input接入Logstash。本文说明如何在VPC环境下通过Logstash从消息队列Kafka版消费消息。

前提条件

在开始本教程前，请确保您已完成以下操作：

- 购买并部署消息队列Kafka版实例。详情请参见VPC接入。
- 下载并安装Logstash。详情请参见Download Logstash。
- 下载并安装JDK 8。详情请参见Download JDK 8。

步骤一：获取接入点

Logstash通过消息队列Kafka版的接入点与消息队列Kafka版建立连接。

1. 登录消息队列Kafka版控制台。
2. 在左侧导航栏，单击**实例详情**。
3. 在**实例详情**页面，选择要作为Input接入Logstash的实例。
4. 在**基本信息**区域，获取实例的接入点。

> **说明** 不同接入点的差异，请参见接入点对比。

步骤二：创建Topic

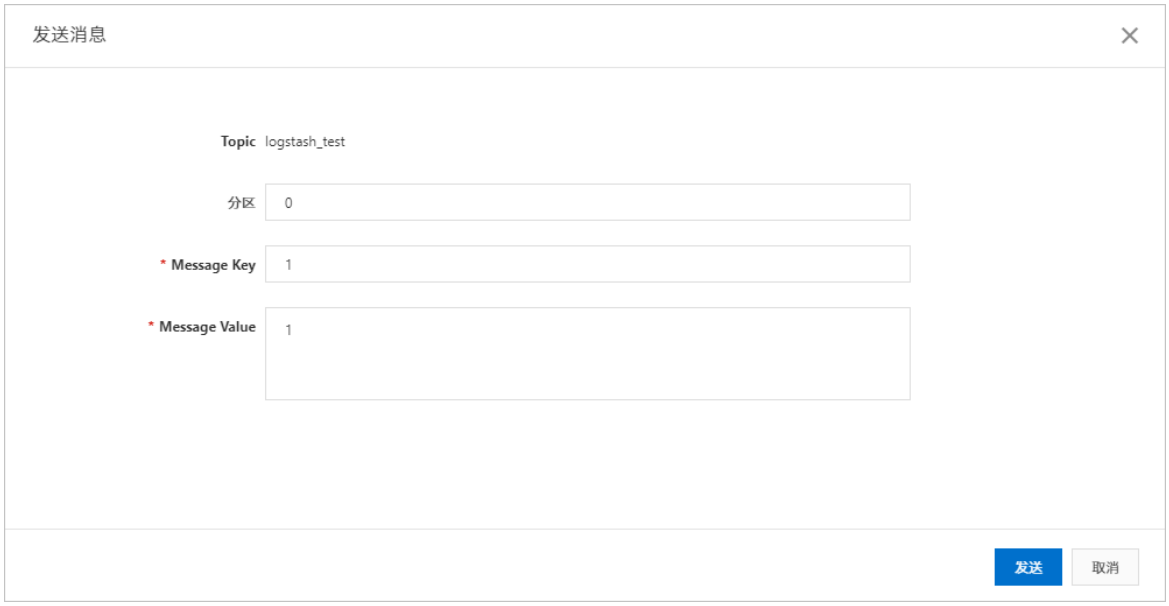
创建用于存储消息的Topic。

1. 在消息队列Kafka版控制台的左侧导航栏，单击**Topic管理**。
2. 在**Topic管理**页面，单击**创建Topic**。
3. 在**创建Topic**对话框，输入Topic信息，然后单击**创建**。

步骤三：发送消息

向创建的Topic发送消息。

1. 在消息队列Kafka版控制台的Topic管理页面，找到创建的Topic，在其右侧操作列，单击发送消息。
2. 在发送消息对话框，输入消息信息，然后单击发送。



步骤四：创建Consumer Group

创建Logstash所属的Consumer Group。

1. 在消息队列Kafka版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。
3. 在创建Consumer Group页面，输入Consumer Group信息，然后单击创建。

步骤五：Logstash消费消息

在安装了Logstash的机器上启动Logstash，从创建的Topic中消费消息。

1. 执行cd命令切换到logstash的*bin*目录。
2. 创建*input.conf*配置文件。
   i. 执行命令 `vim input.conf` 创建空的配置文件。
   ii. 按*i*键进入插入模式。

iii. 输入以下内容。

```
input {
 kafka {
    bootstrap_servers => "192.168.XXX.XXX:9092,192.168.XXX.XXX:9092,192.168.XXX.XXX:9092"
    group_id => "logstash_group"
    topics => ["logstash_test"]
    consumer_threads => 12
    auto_offset_reset => "earliest"
 }
}
output {
 stdout{codec=>rubydebug}
}
```

参数	描述	示例值
bootstrap_servers	消息队列Kafka版提供以下VPC接入点： ■ 默认接入点 ■ SASL接入点	192.168.XXX.XXX:9092,192.168.XXX.XXX:9092,192.168.XXX.XXX:9092
group_id	Cosumer Group的名称。	logstash_group
topics	Topic的名称。	logstash_test
consumer_threads	消费线程数。建议与Topic的分区数保持一致。	12
auto_offset_reset	重置偏移量。取值： ■ earliest：读取最早的消息。 ■ latest：读取最新的消息。	earliest

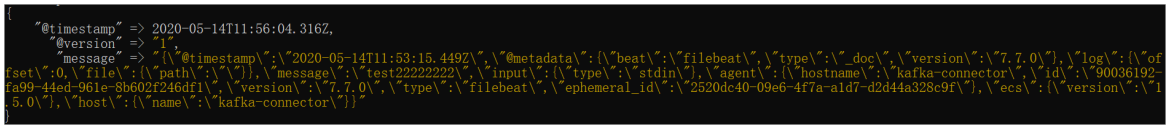
iv. 按*Esc*键回到命令行模式。

v. 按*：*键进入底行模式，输入*wq*，然后按回车键保存文件并退出。

3. 执行以下命令消费消息。

```
./logstash -f input.conf
```

返回结果如下。



更多信息

更多参数设置，请参考Kafka input plugin。

2.1.2.2. 作为Output接入

消息队列Kafka版可以作为Output接入Logstash。本文说明如何在VPC环境下通过Logstash向消息队列Kafka版发送消息。

前提条件

在开始本教程前，请确保您已完成以下操作：

- 购买并部署消息队列Kafka版实例。详情请参见VPC接入。
- 下载并安装Logstash。详情请参见Download Logstash。
- 下载并安装JDK 8。详情请参见Download JDK 8。

步骤一：获取接入点

Logstash通过消息队列Kafka版的接入点与消息队列Kafka版建立连接。

1. 登录消息队列Kafka版控制台。
2. 在左侧导航栏，单击**实例详情**。
3. 在**实例详情**页面，选择要作为Output接入Logstash的实例。
4. 在**基本信息**区域，获取实例的接入点。

> **说明** 不同接入点的差异，请参见接入点对比。

步骤二：创建Topic

创建用于存储消息的Topic。

1. 在消息队列Kafka版控制台的左侧导航栏，单击**Topic管理**。
2. 在**Topic管理**页面，单击**创建Topic**。
3. 在**创建Topic**页面，输入Topic信息，然后单击**创建**。

步骤三：Logstash发送消息

在安装了Logstash的机器上启动Logstash，向创建的Topic发送消息。

1. 执行cd命令切换到logstash的*bin*目录。
2. 创建*output.conf*配置文件。
   i. 执行命令 `vim output.conf` 创建空的配置文件。
   ii. 按*i*键进入插入模式。

iii. 输入以下内容。

```
input {
  input {
    stdin{}
 }
}

output {
  kafka {
     bootstrap_servers => "192.168.XXX.XXX:9092,192.168.XXX.XXX:9092,192.168.XXX.XXX:9092"
     topic_id => "logstash_test"
    }
}
```

参数	描述	示例值
bootstrap_servers	消息队列Kafka版提供以下VPC接入点： ■ 默认接入点 ■ SASL接入点	192.168.XXX.XXX:9092,192.168.XXX.XXX:9092,192.168.XXX.XXX:9092
topic_id	Topic的名称。	logstash_test

iv. 按*Esc*键回到命令行模式。

v. 按*：*键进入底行模式，输入*wq*，然后按回车键保存文件并退出。

3. 向创建的Topic发送消息。

i. 执行 `./logstash -f output.conf` 。

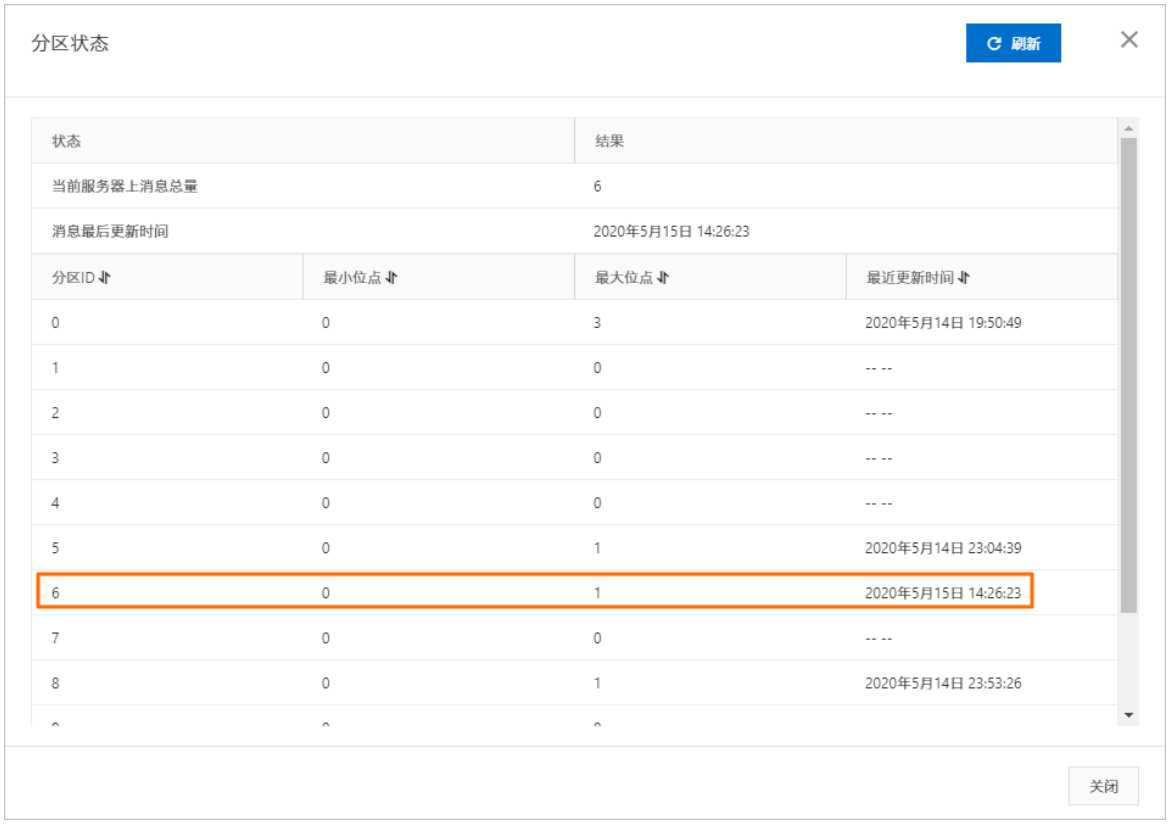
ii. 输入*test*，然后按回车键。
返回结果如下。

```
[2020-05-15T14:26:13,576][INFO ][logstash.javapipeline    ][main] Starting pipeline {:pipeline_id=>"main", "pipeline.workers"=>2, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>50, "pipeline.max_inflight"=>250, "pipeline.sources"=>["/home/logstash-7.6.2/bin/output.conf"], :thread=>"#<Thread:0x7be16bf0 run>"}
[2020-05-15T14:26:13,974][INFO ][org.apache.kafka.clients.Metadata][main] [Producer clientId=producer-1] Cluster ID: TsUt5To9Q4-f9YX_5yz28Q
[2020-05-15T14:26:15,241][INFO ][logstash.javapipeline    ][main] Pipeline started {"pipeline.id"=>"main"}
The stdin plugin is now waiting for input:
[2020-05-15T14:26:15,408][INFO ][logstash.agent           ] Pipelines running {:count=>1, :running_pipelines=>[:main], :non_running_pipelines=>[]}
[2020-05-15T14:26:15,801][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>9601}
test
```

步骤四：查看Topic分区

查看消息发送到Topic的情况。

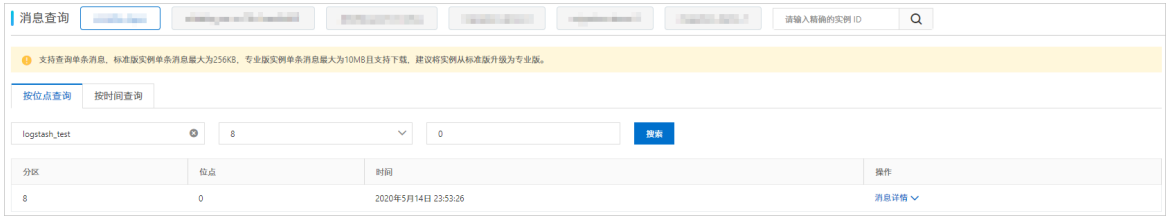
1. 在消息队列Kafka版控制台的左侧导航栏，单击**Topic管理**。
2. 在**Topic管理**页面，选择作为Output接入Logstash的实例，找到发送消息的Topic，在其右侧**操作**列单击**分区状态**。
3. 在**分区状态**页面，单击**刷新**。
发送的消息的分区ID和位点信息如下图所示。



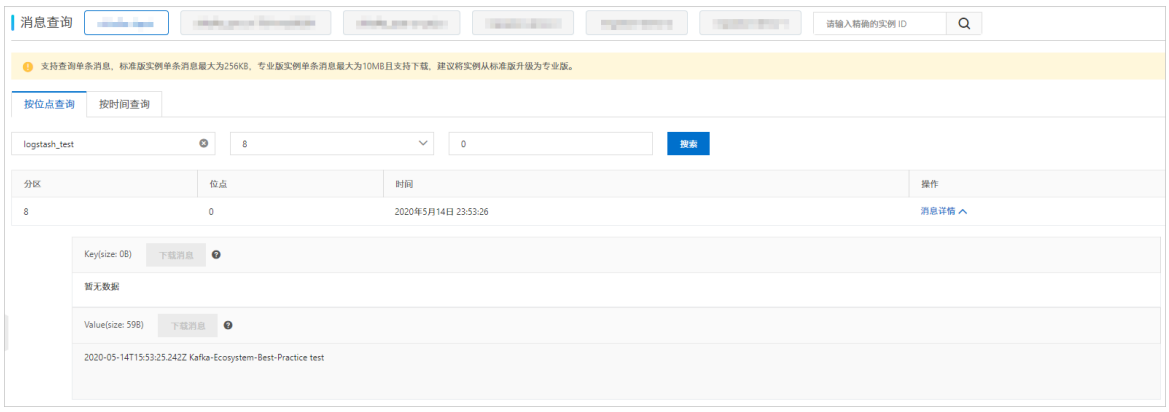
步骤五：按位点查询消息

您可以根据发送的消息的分区ID和位点信息查询该消息。

1. 在消息队列Kafka版控制台的左侧导航栏，单击**消息查询**。
2. 在**消息查询**页面，单击**按位点查询**页签。
3. 选择发送了消息的Topic，选择发送的消息的分区ID，选择发送的消息的位点，然后单击**搜索**。



4. （可选）在搜索结果右侧的**操作**列，单击**消息详情**。



更多信息
更多参数设置，请参考Kafka output plugin。

2.2. Filebeat

2.2.1. 接入Filebeat

本文介绍如何将消息队列Kafka版接入Filebeat。
kafka filebeat

Filebeat
Filebeat是用于转发和集中日志数据的轻量级传输程序。Filebeat可以监听指定的日志文件或位置，从中收集日志事件并将其转发到Elasticsearch或Logstash进行索引。Filebeat的工作原理如下：

1. Filebeat启动一个或多个Input，Input在指定的位置中查找日志数据。
2. Filebeat为每个找到的日志启动Harvester，Harvester读取日志并将日志数据发送到libbeat。
3. libbeat聚集数据，然后将聚集的数据发送到配置的Output。

接入优势
消息队列Kafka版接入Filebeat可以带来以下优势：

- 异步处理：防止突发流量。
- 应用解耦：当下游异常时，不会影响上游工作。
- 减少开销：减少Filebeat的资源开销。

接入方案
消息队列Kafka版支持以下方式接入Filebeat：

- 作为Input接入
- 作为Output接入

2.2.2. VPC

2.2.2.1. 作为Input接入

消息队列Kafka版可以作为Input接入Filebeat。本文说明如何在VPC环境下通过Filebeat从消息队列Kafka版消费消息。

背景信息
在开始本教程前，请确保您已完成以下操作：

- 购买并部署消息队列Kafka版实例。详情请参见VPC接入。
- 下载并安装Filebeat。详情请参见Download Filebeat。
- 下载并安装JDK 8。详情请参见Download JDK 8。

步骤一：获取接入点
Filebeat通过消息队列Kafka版的接入点与消息队列Kafka版建立连接。

1. 登录消息队列Kafka版控制台。
2. 在左侧导航栏，单击**实例详情**。

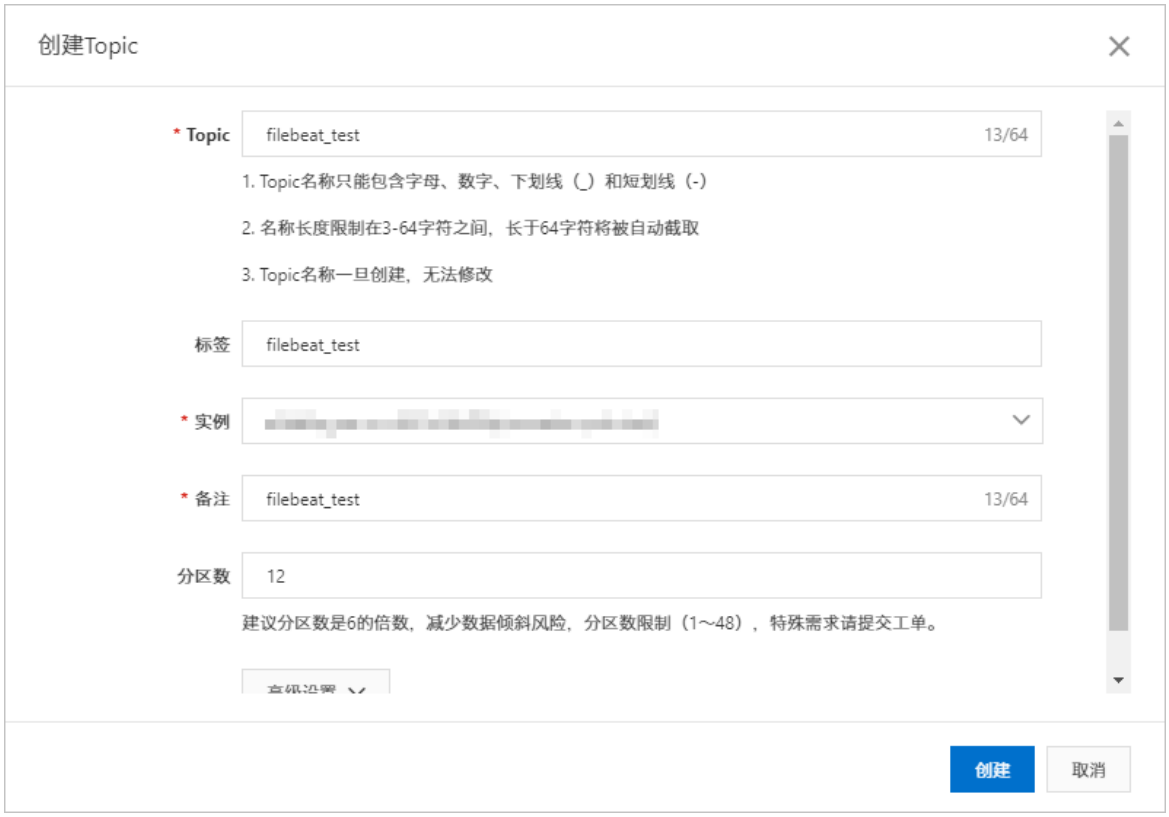
3. 在**实例详情**页面，选择要作为Input接入Filebeat的实例。
4. 在**基本信息**区域，获取实例的接入点。

> **说明** 不同接入点的差异，请参见**接入点对比**。

步骤二：创建Topic

创建用于存储消息的Topic。

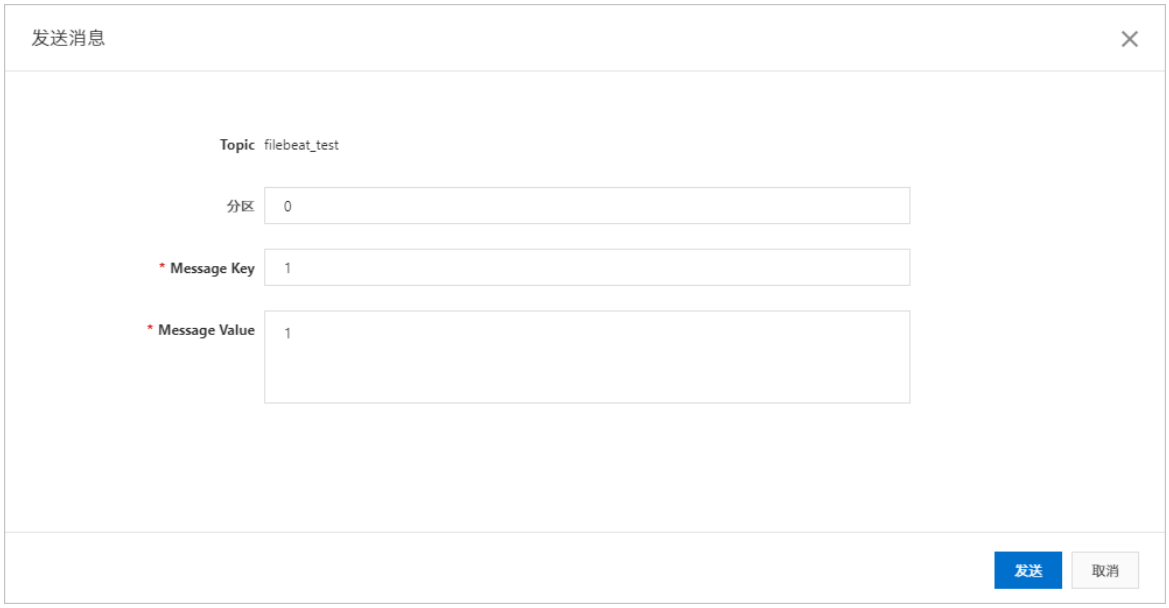
1. 在消息队列Kafka版控制台的左侧导航栏，单击**Topic管理**。
2. 在**Topic管理**页面，单击**创建Topic**。
3. 在**创建Topic**页面，输入Topic信息，然后单击**创建**。



步骤三：发送消息

向创建的Topic发送消息。

1. 在消息队列Kafka版控制台的**Topic管理**页面，找到创建的Topic，在其右侧**操作**列，单击**发送消息**。
2. 在**发送消息**对话框，输入消息信息，然后单击**发送**。



步骤四：创建Consumer Group

创建Filebeat所属的Consumer Group。

1. 在消息队列Kafka版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。
3. 在创建Consumer Group页面，输入Consumer Group信息，然后单击创建。

步骤五：Filebeat消费消息

在安装了Filebeat的机器上启动Filebeat，从创建的Topic中消费消息。

1. 执行cd命令切换到Filebeat的安装目录。
2. 创建*input.yml*配置文件。
   i. 执行命令 `vim input.yml` 创建空的配置文件。
   ii. 按*i*键进入插入模式。

iii. 输入以下内容。

```
filebeat.inputs:
- type: kafka
  hosts:
    - 192.168.XX.XX:9092
    - 192.168.XX.XX:9092
    - 192.168.XX.XX:9092
  topics: ["filebeat_test"]
  group_id: "filebeat_group"

output.console:
  pretty: true
```

参数	描述	示例值
type	Filebeat的Input类型。	kafka
hosts	消息队列Kafka版提供以下VPC接入点： ■ 默认接入点 ■ SASL接入点	- 192.168.XX.XX:9092 - 192.168.XX.XX:9092 - 192.168.XX.XX:9092
topics	Topic的名称。	filebeat_test
group_id	Consumer Group的名称。	filebeat_group

更多参数说明，请参见Kafka input plugin。

iv. 按*Esc*键回到命令行模式。

v. 按*：*键进入底行模式，输入*wq*，然后按回车键保存文件并退出。

3. 执行以下命令消费消息。

```
./filebeat -c ./input.yml
```

```
{
  "@timestamp": "2020-05-18T12:48:32.782Z",
  "@metadata": {
    "beat": "filebeat",
    "type": "_doc",
    "version": "7.7.0"
  },
  "agent": {
    "hostname": "kafka-connector",
    "id": "90036192-fa99-44ed-961e-8b602f246df1",
    "version": "7.7.0",
    "type": "filebeat",
    "ephemeral_id": "26b5c40d-4696-4b83-a5f8-ecc4b1213b04"
  },
  "message": "1",
  "kafka": {
    "headers": [],
    "topic": "filebeat_test",
    "partition": 0,
    "offset": 0,
    "key": "1"
  },
  "input": {
    "type": "kafka"
  },
  "ecs": {
    "version": "1.5.0"
  },
  "host": {
    "name": "kafka-connector"
  }
```


2.2.2.2. 作为Output接入

消息队列Kafka版可以作为Output接入Filebeat。本文说明如何在公网环境下通过Filebeat向消息队列Kafka版发送消息。

前提条件

在开始本教程前，请确保您已完成以下操作：

- 购买并部署消息队列Kafka版实例。详情请参见VPC接入。
- 下载并安装Filebeat。详情请参见Download Filebeat。
- 下载并安装JDK 8。详情请参见Download JDK 8。

步骤一：获取接入点

Filebeat通过消息队列Kafka版的接入点与消息队列Kafka版建立连接。

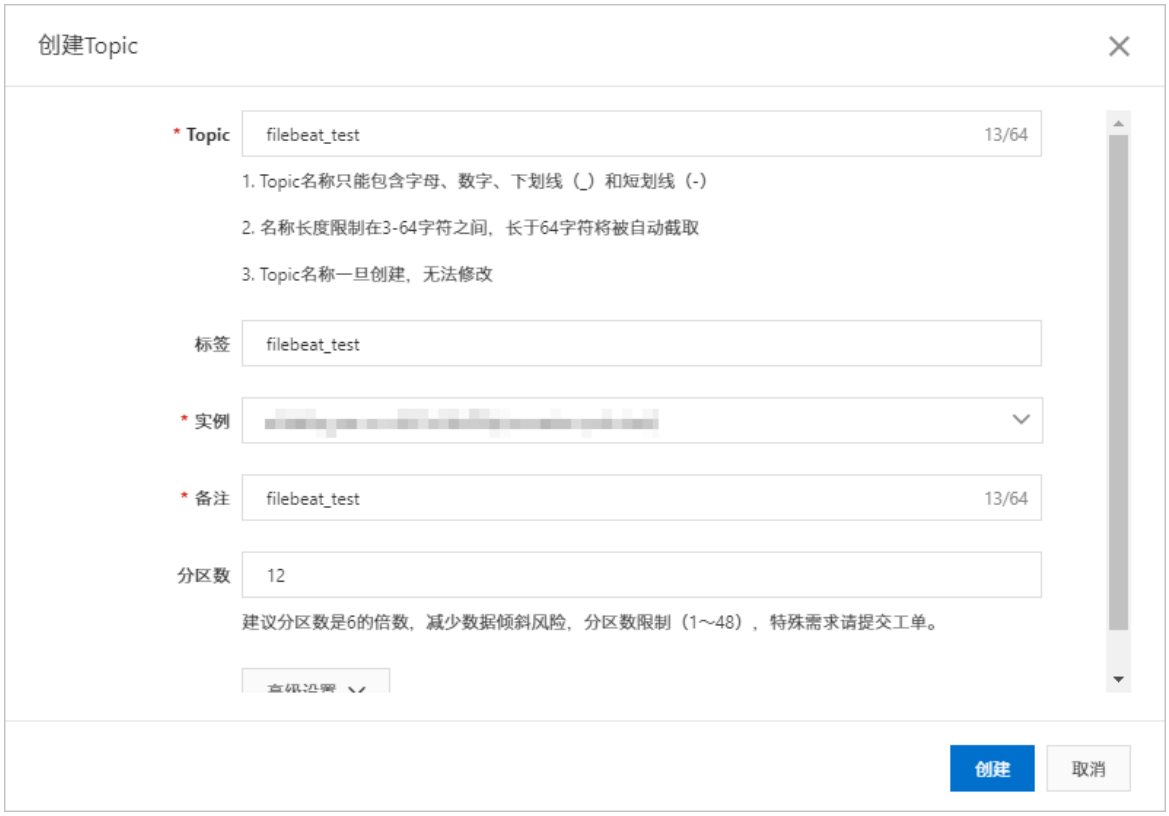
1. 登录消息队列Kafka版控制台。
2. 在左侧导航栏，单击**实例详情**。
3. 在**实例详情**页面，选择要作为Output接入Filebeat的实例。
4. 在**基本信息**区域，获取实例的接入点。

> **说明** 不同接入点的差异，请参见接入点对比。

步骤二：创建Topic

创建用于存储消息的Topic。

1. 在消息队列Kafka版控制台的左侧导航栏，单击Topic管理。
2. 在Topic管理页面，单击创建Topic。
3. 在创建Topic页面，输入Topic信息，然后单击创建。



步骤三：Filebeat发送消息

在安装了Filebeat的机器上启动Filebeat，向创建的Topic发送消息。

1. 执行cd命令切换到Filebeat的安装目录。
2. 创建*output.conf*配置文件。
   i. 执行命令 `vim output.conf` 创建空的配置文件。
   ii. 按*i*键进入插入模式。
   iii. 输入以下内容。

```
filebeat.inputs:
- type: stdin

output.kafka:
 hosts: ["192.XX.XX.XX:9092", "192.XX.XX.XX:9092", "192.XX.XX.XX:9092"]

 topic: 'filebeat_test'

 required_acks: 1
 compression: none
 max_message_bytes: 1000000
```

参数	描述	示例值
hosts	消息队列Kafka版提供以下VPC接入点： ■ 默认接入点 ■ SASL接入点	192.168.XXX.XXX:9092,192.168.XXX.XXX:9092,192.168.XXX.XXX:9092
topic	Topic的名称。	filebeat_test
required_acks	ACK可靠性。取值： ■ 0：无响应 ■ 1：等待本地提交 ■ -1：等待所有副本提交 默认值为1。	1
compression	数据压缩编译码器。默认值为gzip。取值： ■ none：无 ■ snappy：用来压缩和解压缩的C++开发包 ■ lz4：着重于压缩和解压缩速度的无损数据压缩算法 ■ gzip：GNU自由软件的文件压缩程序	none
max_message_bytes	最大消息大小。单位为字节。默认值为1000000。该值应小于您配置的消息队列Kafka版最大消息大小。	1000000

更多参数说明，请参见Kafka output plugin。

iv. 按*Esc*键回到命令行模式。

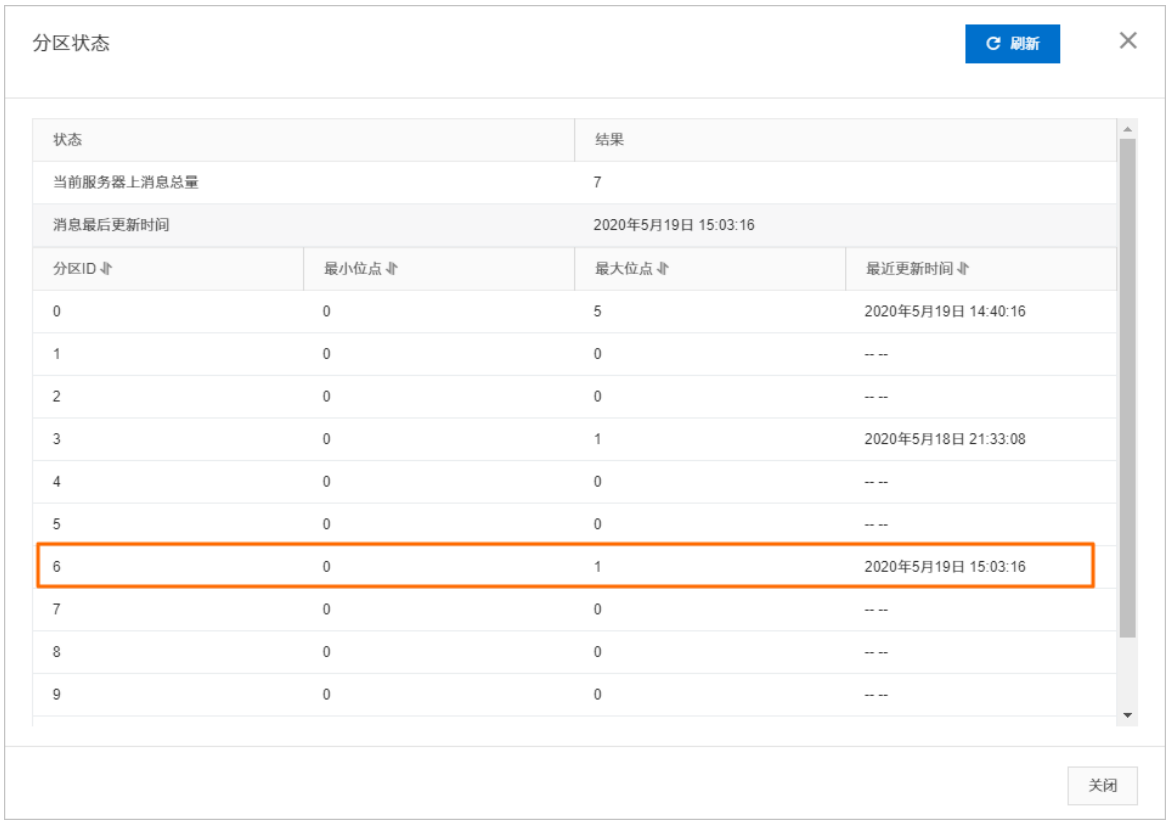
v. 按*：*键进入底行模式，输入*wq*，然后按回车键保存文件并退出。

3. 向创建的Topic发送消息。
   i. 执行 `./filebeat -c ./output.yml` 。
   ii. 输入*test*，然后按回车键。

步骤四：查看Topic分区

查看消息发送到Topic的情况。

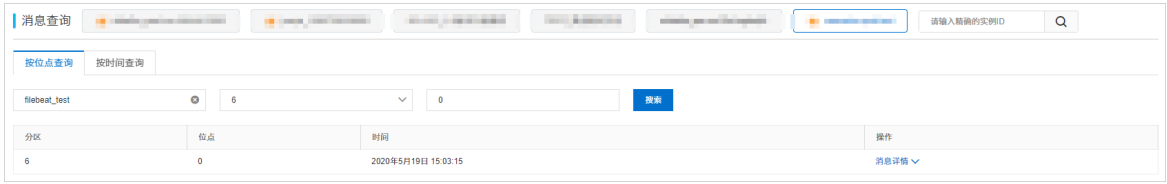
1. 在消息队列Kafka版控制台的左侧导航栏，单击**Topic管理**。
2. 在**Topic管理**页面，选择作为Output接入Filebeat的实例，找到发送消息的Topic，在其右侧**操作**列单击**分区状态**。
3. 在**分区状态**页面，单击**刷新**。
   发送的消息的分区ID和位点信息如下图所示。



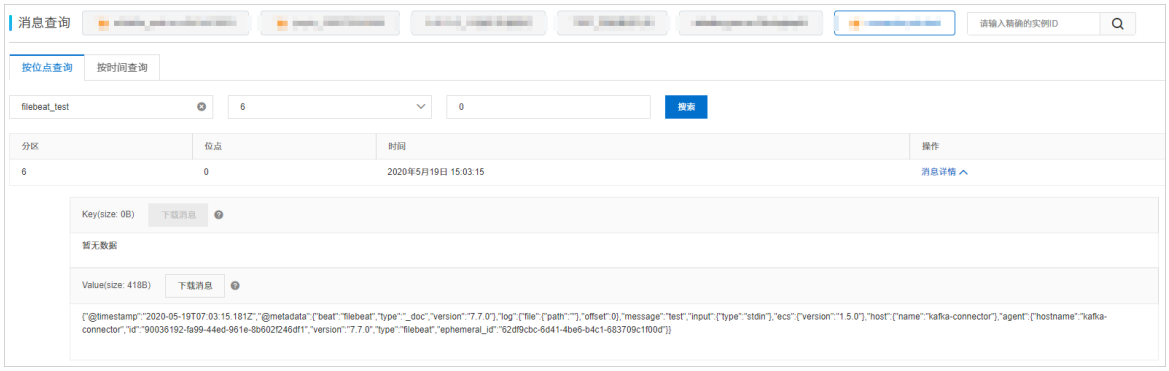
步骤五：按位点查询消息

您可以根据发送的消息的分区ID和位点信息查询该消息。

1. 在消息队列Kafka版控制台的左侧导航栏，单击**消息查询**。
2. 在**消息查询**页面，单击**按位点查询**页签。
3. 选择发送了消息的Topic，选择发送的消息的分区ID，选择发送的消息的位点，然后单击**搜索**。



4. （可选）在搜索结果右侧的**操作**列，单击**消息详情**。

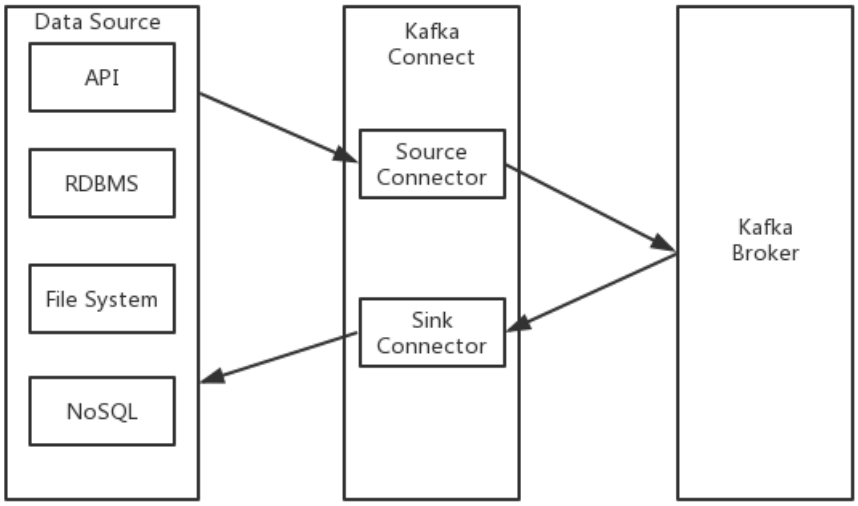


2.3. 使用Kafka Connect将MySQL数据同步至消息队列Kafka版

本教程介绍如何使用Kafka Connect的Source Connector将MySQL的数据同步至消息队列Kafka版。

背景信息

Kafka Connect主要用于将数据流输入和输出消息队列Kafka版。Kafka Connect主要通过各种Source Connector的实现，将数据从第三方系统输入到Kafka broker，通过各种Sink Connector实现，将数据从Kafka broker中导入到第三方系统。



前提条件

在开始本教程前，请确保您已完成以下操作：

- 已下载MySQL Source Connector。

  > 说明 本教程以0.5.2版本的MySQL Source Connector为例。

- 已下载Kafka Connect。

> 说明 本教程以0.10.2.2版本的Kafka Connect为例。

- 已安装docker。

步骤一：配置Kafka Connect

1. 将下载完成的MySQL Connector解压到指定目录。
2. 在Kafka Connect的配置文件*connect-distributed.properties*中配置插件安装位置。

```
plugin.path=/kafka/connect/plugins
```

> 注意
> Kafka Connect的早期版本不支持配置*plugin.path*，您需要在*CLASSPATH*中指定插件位置。
>
> ```
> export CLASSPATH=/kafka/connect/plugins/mysql-connector/*
> ```

步骤二：启动Kafka Connect

在配置好*connect-distributed.properties*后，执行以下命令启动Kafka Connect。

- VPC接入
  执行命令 `bin/connect-distributed.sh config/connect-distributed.properties` 启动Kafka Connect。

步骤三：安装MySQL

1. 下载docker-compose-mysql.yaml。
2. 执行以下命令安装MySQL。

```
export DEBEZIUM_VERSION=0.5
docker-compose -f docker-compose-mysql.yaml up
```

步骤四：配置MySQL

1. 执行以下命令开启MySQL的binlog写入功能，并配置binlog模式为row。

```
[mysqld]
log-bin=mysql-bin
binlog-format=ROW
server_id=1
```

2. 执行以下命令设置MySQL的User权限。

```
GRANT SELECT, RELOAD, SHOW DATABASES, REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'de
bezium' IDENTIFIED BY 'dbz';
```

> 说明 示例中MySQL的User为*debezium*，密码为*dbz*。

步骤五：启动MySQL Connector

1. 下载register-mysql.json。
2. 编辑*register-mysql.json*。
   - VPC接入

     ```
     ## Kafka接入点，通过控制台获取
     ## 您在控制台获取的默认接入点
     "database.history.kafka.bootstrap.servers" : "kafka:9092",
     ## 需要提前在控制台创建同名Topic，在本例中创建topic：server1
     ## 所有Table的变更数据，会记录在server1.$DATABASE.$TABLE的Topic中，如 server1.inventory.pro
     ducts
     ## 因此用户需要提前在控制台中创建所有相关 Topic
     "database.server.name": "server1",
     ## 记录schema变化信息将记录在这个Topic中
     ## 需要提前在控制台创建
     "database.history.kafka.topic": "schema-changes-inventory"
     ```

3. 配置好register-mysql.json后，您需要根据配置在控制台创建相应的Topic，相关操作步骤请参见步骤一：创建Topic。
   按照本教程中的方式安装的MySQL，您可以看到MySQL中已经提前创建好了*database：inventory*。其中有四张表：
   - *customers*
   - *orders*
   - *products*
   - *products_on_hand*

   根据以上配置，您需要使用OpenAPI创建Topic：
   - *server1*
   - *server1.inventory.customers*
   - *server1.inventory.orders*
   - *server1.inventory.products*
   - *server1.inventory.products_on_hand*

   在*register-mysql.json*中，配置了将schema变化信息记录在*schema-changes-testDB*，因此您还需要使用OpenAPI创建Topic：*schema-changes-inventory*。 使用OpenAPI创建Topic，请参见CreateTopic。
4. 执行以下命令启动MySQL Connector。

   ```
   > curl -i -X POST -H "Accept:application/json" -H  "Content-Type:application/json" http://localhost
   :8083/connectors/ -d @register-mysql.json
   ```

结果验证

按照以下步骤操作确认消息队列Kafka版能否接收到MySQL的变更数据。

1. 变更MySQL Table中的数据。
2. 在控制台的**消息查询**页面，查询变更数据。

2.4. 使用Kafka Connect将SQL Server数据同步至消息队列Kafka版

本教程介绍如何使用Kafka Connect的Source Connector将SQL Server的数据同步至消息队列Kafka版。

前提条件

在开始本教程前，请确保您已完成以下操作：

- 已下载SQL Server Source Connector。详情请参见SQL Server Source Connector。
- 已下载Kafka Connect。详情请参见Kafka Connect。

> 说明 SQL Server Source Connector目前只支持2.1.0及以上版本的Kafka Connect。

- 已下载Docker。详情请参见Docker。

步骤一：配置Kafka Connect

1. 将下载完成的SQL Server Connector解压到指定目录。
2. 在Kafka Connect的配置文件*connect-distributed.properties*中配置插件安装位置。

```
## 指定插件解压后的路径。
plugin.path=/kafka/connect/plugins
```

> 注意
> Kafka Connect的早期版本不支持配置*plugin.path*，您需要在*CLASSPATH*中指定插件位置。
>
> ```
> export CLASSPATH=/kafka/connect/plugins/sqlserver-connector/*
> ```

步骤二：启动Kafka Connect

配置好*connect-distributed.properties*后，执行以下命令启动Kafka Connect。

```
## 启动Kafka Connect。
bin/connect-distributed.sh config/connect-distributed.properties
```

步骤三：安装SQL Server

> 注意 SQL Server 2016 SP1以上版本支持CDC，因此您的SQL Server版本必须高于该版本。

1. 下载docker-compose-sqlserver.yaml。
2. 执行以下命令安装SQL Server。

```
docker-compose -f docker-compose-sqlserver.yaml up
```

步骤四：配置SQL Server

1. 下载inventory.sql。

2. 执行以下命令初始化SQL Server中的测试数据。

```
cat inventory.sql | docker exec -i tutorial_sqlserver_1 bash -c '/opt/mssql-tools/bin/sqlcmd -U sa -P $SA_PASSWORD'
```

3. （可选）如果您需要监听SQL Server中已有的数据表，请完成以下配置：
   i. 执行以下命令开启CDC配置。

```
## 开启CDC模板数据库
USE testDB
GO
EXEC sys.sp_cdc_enable_db
GO
```

   ii. 执行以下命令开启指定Table的CDC配置。

```
## Enable a Table Specifying Filegroup Option Template
USE testDB
GO

EXEC sys.sp_cdc_enable_table
@source_schema = N'dbo',
@source_name   = N'MyTable',
@role_name     = N'MyRole',
@filegroup_name = N'MyDB_CT',
@supports_net_changes = 1
GO
```

   iii. 执行以下命令确认是否有权限访问CDC Table。

```
EXEC sys.sp_cdc_help_change_data_capture
GO
```

> 说明 如果返回结果为空，您需要确认是否有权限访问该表。

   iv. 执行以下命令确认SQL Server Agent已开启。

```
EXEC master.dbo.xp_servicecontrol N'QUERYSTATE',N'SQLSERVERAGENT'
```

> 说明 如果返回结果为 `Running`，则说明SQL Server Agent已开启。

步骤五：启动SQL Server Connector

1. 下载register-sqlserver.json。
2. 编辑*register-sqlserver.json*。

- VPC接入

```
## 消息队列Kafka版实例的默认接入点，您可以在消息队列Kafka版控制台获取。
"database.history.kafka.bootstrap.servers" : "kafka:9092",
## 您需要提前在消息队列Kafka版控制台创建同名Topic，在本例中创建topic：server1。
## 所有table的变更数据，会记录在server1.$DATABASE.$TABLE的topic中，例如server1.testDB.products。
## 因此您需要提前在消息队列Kafka版控制台中创建所有相关Topic。
"database.server.name": "server1",
## 记录schema变化信息将记录在该Topic中。
## 您需要提前在消息队列Kafka版控制台创建该Topic。
"database.history.kafka.topic": "schema-changes-inventory"
```

3. 完成*register-sqlserver.json*配置后，您需要根据配置在控制台创建相应的Topic，相关操作步骤请参见步骤一：创建Topic。
   按照本教程中的方式安装的SQL Server，您可以看到SQL Server中已经提前创建*db name：testDB*。其中有四张表：
   - *customers*
   - *orders*
   - *products*
   - *products_on_hand*

   根据以上*register-sqlserver.json*的配置，您需要使用OpenAPI创建Topic：
   - *server1*
   - *server1.testDB.customers*
   - *server1.testDB.orders*
   - *server1.testDB.products*
   - *server1.testDB.products_on_hand*

   在*register-sqlserver.json*中，配置了将schema变化信息记录在*schema-changes-testDB*，因此您还需要使用OpenAPI创建Topic：*schema-changes-inventory*，相关操作请参见CreateTopic。
4. 执行以下命令启动SQL Server。

```
> curl -i -X POST -H "Accept:application/json" -H  "Content-Type:application/json" http://localhost:8083/connectors/ -d @register-sqlserver.json
```

结果验证

确认消息队列Kafka版能否接收到SQL Server的变更数据：

1. 变更监听SQL Server中的数据。
2. 在控制台的**消息查询**页面，查询变更消息。具体操作步骤，请参见查询消息。