

数据接入服务

用户指南

文档版本 01
发布日期 2024-10-28



版权所有 © 华为云计算技术有限公司 2024。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目录

1 IAM 权限管理	1
1.1 创建用户并授权使用 DIS	1
2 入门	3
2.1 DIS 使用流程简介	3
2.2 步骤 1: 开通 DIS 通道	3
2.3 步骤 2: 准备 DIS 应用开发环境	7
2.4 步骤 3: 发送数据到 DIS	11
2.5 步骤 4: 从 DIS 获取数据	12
2.6 获取认证信息	13
2.7 连接 OBS	13
2.8 IAM 中创建 DIS 委托	13
3 管理通道	15
3.1 通道列表简介	15
3.2 查看通道监控信息	15
3.3 变更源数据类型	17
3.4 管理源数据 Schema	17
3.5 管理通道标签	21
3.6 管理 App	23
3.7 授权管理	24
3.8 调试通道	24
3.9 弹性伸缩分区	26
3.10 删除通道	29
4 使用 DIS	30
4.1 检查与配置 DNS 信息	30
4.2 使用 Agent 上传数据	31
4.2.1 DIS Agent 概述	31
4.2.2 安装前准备	32
4.2.3 安装 DIS Agent	33
4.2.4 配置 DIS Agent	34
4.2.5 启动 DIS Agent	42
4.2.6 验证 DIS Agent	43
4.2.7 停止 DIS Agent	45

4.3 使用 DIS Flume Plugin 上传与下载数据.....	45
4.3.1 DIS Flume Plugin 概述.....	45
4.3.2 安装 DIS Flume Plugin 前准备.....	46
4.3.3 安装 Plugin.....	47
4.3.4 配置 Plugin.....	48
4.3.5 验证 Plugin.....	52
4.3.6 卸载 Plugin(可选).....	53
4.4 使用 DIS Logstash Plugin 上传与下载数据.....	54
4.4.1 DIS Logstash Plugin 概述.....	54
4.4.2 安装 DIS Logstash Plugin 前准备.....	55
4.4.3 在线安装 DIS Logstash Plugin.....	55
4.4.4 离线安装 DIS Logstash Plugin.....	56
4.4.5 配置 DIS Logstash Plugin.....	57
4.4.6 验证 DIS Logstash Plugin.....	60
4.4.7 卸载 DIS Logstash Plugin(可选).....	61
4.5 使用 Kafka Adapter 上传与下载数据.....	61
4.5.1 Kafka Adapter 概述.....	61
4.5.2 准备环境.....	61
4.5.3 上传数据.....	63
4.5.4 数据下载的消费模式.....	68
4.5.5 下载数据之消费位移.....	74
4.5.6 与原生 KafkaConsumer 接口适配说明.....	75
4.5.7 使用 AuthToken 认证.....	79
4.6 使用 DIS Spark Streaming 下载数据.....	79
4.6.1 DIS Spark Streaming 概述.....	79
4.6.2 准备 DIS Spark Streaming 的相关环境.....	80
4.6.3 自定义 SparkStreaming 作业.....	81
4.7 使用 DIS Flink Connector 上传与下载数据.....	86
4.7.1 DIS Flink Connector 概述.....	86
4.7.2 准备 DIS Flink Connector 的相关环境.....	86
4.7.3 自定义 Flink Streaming 作业.....	87
5 管理转储任务.....	94
5.1 新增转储任务.....	94
5.2 转储至 OBS.....	96
5.3 转储至 DLI.....	102
5.4 转储至 DWS.....	103
5.5 转储至 MRS.....	106
6 管理企业项目.....	109
7 事件通知.....	111
7.1 事件通知概述.....	111
7.2 订阅事件通知.....	112

7.3 查看事件.....	114
8 使用 CES 监控 DIS.....	115
8.1 支持的监控指标.....	115
8.2 设置告警规则.....	116
8.3 查看监控指标.....	117

1 IAM 权限管理

1.1 创建用户并授权使用 DIS

如果您需要对您所拥有的数据接入服务（DIS）进行精细的权限管理，您可以使用[统一身份认证服务](#)（Identity and Access Management，简称IAM），通过IAM，您可以：

- 根据企业的业务组织，在您的华为云账号中，给企业中不同职能部门的员工创建IAM用户，让员工拥有唯一安全凭证，并使用DIS资源。
- 根据企业用户的职能，设置不同的访问权限，以达到用户之间的权限隔离。
- 将DIS资源委托给更专业、高效的其他华为云账号或者云服务，这些账号或者云服务可以根据权限进行代运维。

如果华为云账号已经能满足您的要求，不需要创建独立的IAM用户，您可以跳过本章节，不影响您使用DIS服务的其它功能。

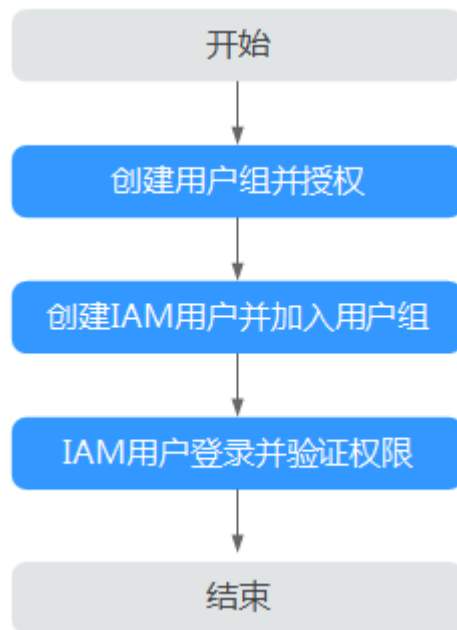
本章节为您介绍对用户授权的方法，操作流程如[图1-1](#)所示。

前提条件

给用户组授权之前，请您了解用户组可以添加的DIS权限，并结合实际需求进行选择，DIS支持的系统权限，请参见[DIS系统权限](#)。

示例流程

图 1-1 IAM 用户授权流程



1. **创建用户组并授权**

在IAM控制台创建用户组，并授予数据接入服务的通道管理权限“DIS Operator”。

2. **创建用户并加入用户组**

在IAM控制台创建用户，并将其加入1中创建的用户组。

3. **用户登录**并验证权限

新创建的用户登录控制台，切换至授权区域，验证权限：

- 在“服务列表”中选择“数据接入服务”，进入DIS主界面创建通道，若未提示权限不足，表示“DIS Operator”已生效。
- 在“服务列表”中选择除DIS服务外的任一服务，若提示权限不足，表示“DIS Operator”已生效。

2 入门

2.1 DIS 使用流程简介

DIS的使用流程如下：

步骤1：开通DIS通道

用户使用DIS前需要先开通DIS通道。

步骤2：准备DIS应用开发环境

用户开发DIS应用程序前，首先需要安装应用开发工具。然后获取SDK和样例工程，并导入到用户的开发环境中。

步骤3：发送数据到DIS

基于数据上传业务开发应用程序，并运行程序，实现数据上传功能。数据上传过程中可在Console控制台查看数据上传通道相关信息。

步骤4：从DIS获取数据


基于数据下载业务开发应用程序，并运行程序，实现数据下载功能。

2.2 步骤 1：开通 DIS 通道

用户可以基于云管理平台Web界面开通DIS通道。

操作步骤

步骤1 使用注册账户登录[DIS控制台](#)。





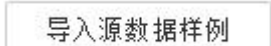




步骤2 单击管理控制台左上角的  ，选择区域和项目。

步骤3 单击“购买接入通道”配置相关参数。

表 2-1 接入通道参数说明

参数	参数解释	参数示例
计费模式	按需计费	按需计费
区域	指的是云服务所在的物理位置。您可以在下拉框中选择并切换区域。	-
基本信息		
通道名称	用户发送或者接收数据时，需要指定通道名称，通道名称不可重复。通道名称由英文字母、数字、中划线和下划线组成。长度为1~64个字符。	dis-Tido
通道类型	<ul style="list-style-type: none"> 普通通道单分区容量：最高发送速度可达 1MB/秒或1000条记录/秒（达到任意一种速度上限才会被限流），最高提取速度可达 2MB/秒，单次请求的记录总大小不能超过1MB（不包含partitionKey数据大小）。 高级通道单分区容量：最高发送速度可达 5MB/秒或2000条记录/秒（达到任意一种速度上限才会被限流），最高提取速度可达 10MB/秒，单次请求的记录总大小不能超过5MB（不包含partitionKey数据大小） 	-
分区数量	分区是DIS数据通道的基本吞吐量单位。	5

参数	参数解释	参数示例
分区计算	<p>用户可以根据实际需求通过系统计算得到一个建议的分区数量值。</p> <ol style="list-style-type: none"> 单击“分区计算”，弹出“计算所需分区数量”对话框。 根据实际需求填写“平均记录大小”、“最大写入记录数”和“消费程序数量”，“预估所需分区数量”选项框中将显示所需的分区数量，此值不可修改。 <p>说明 所需分区计算公式：</p> <ul style="list-style-type: none"> 按流量计算所需写分区数：（所得数值需向上取整后作为分区数） 普通通道：平均记录大小*（1+分区预留比例20%）*最大写入记录数/（1*1024KB） 高级通道：平均记录大小*（1+分区预留比例20%）*最大写入记录数/（5*1024KB） 按消费程序数量计算读分区数：（消费程序数量/2后的数值需要保留两位小数，然后乘以“按流量计算所需写分区数”，最终取值需向上取整） （消费程序数量/2）*按流量计算所需的写分区数 <p>获取“按流量计算所需写分区数”、“按消费程序数量计算读分区数”中的最大值作为预估所需分区数量。</p> <ol style="list-style-type: none"> 单击“使用计算值”将系统计算出的建议值应用于“分区数量”。 	-
生命周期（小时）	<p>存储在DIS中的数据保留的最长时间，超过此时长数据将被清除。</p> <p>取值范围：24~72的整数。</p>	24
源数据类型	<ul style="list-style-type: none"> BLOB：存储在数据库管理系统中的一组二进制数据。“源数据类型”选择“BLOB”，则支持的“转储服务类型”为“OBS”。 JSON：一种开放的文件格式，以易读的文字为基础，用来传输由属性值或者序列性的值组成的数据对象。“源数据类型”选择“JSON”，则支持的“转储服务类型”为“OBS”、“MRS”、“DLI”和“DWS”。 CSV：纯文本形式存储的表格数据，分隔符默认采用逗号。“源数据类型”选择“CSV”，则支持的“转储服务类型”为“OBS”、“DLI”、“DWS”。 	JSON

参数	参数解释	参数示例
自动扩缩容	<p>创建通道的同时是否开启自动扩缩容功能。</p> <p>通过单击  或  来关闭或开启自动扩缩容开关。</p>	<p>说明</p> <p>用户可在创建通道时定义是否自动扩缩容，也可对已创建的通道修改自动扩缩容属性。</p>
自动缩容最小分区数	设置自动缩容的分区下限，自动缩容的目标分区数不小于下限值。	-
自动扩容最大分区数	设置自动扩容的分区上限，自动扩容的目标分区数不超过上限值。	-
源数据分隔符	源数据为CSV格式时的数据分隔符。	-
Schema开关	<p>创建通道的同时是否为其创建数据Schema。源数据类型为JSON或CSV时可配置该参数。</p> <p>通过单击  或  来关闭或开启Schema配置开关。</p> <p>说明</p> <p>若创建通道时，没有同时创建数据Schema,可待通道创建成功后。到通道的管理页面创建数据Schema，详情请参见管理源数据Schema。</p>	<p>“源数据类型”为“JSON”和“CSV”时，可选择创建数据Schema。</p>
源数据Schema	<p>支持输入和导入源数据样例，源数据样例格式为JSON或者CSV，详细操作请参见管理源数据Schema。</p> <ol style="list-style-type: none"> 在左侧文本框中输入JSON或者CSV格式的源数据样例，也可单击  导入源数据样例。 在左侧文本框中单击  ，可删除左侧文本框中已输入或导入的源数据样例。 在左侧文本框中单击  ，可在右侧文本框中根据源数据样例生成Avro schema。 在右侧文本框中单击  ，可删除已生成的Avro schema。 在右侧文本框中单击  ，可修改已生成的Avro schema。 	<p>仅当“Schema配置开关”配置为“开启”：时需要配置此参数。</p>

参数	参数解释	参数示例
企业项目	配置通道所属的企业项目。已开通企业项目管理服务的用户才可以配置该参数。默认值为 default。 企业项目是一种云资源管理方式，企业项目管理服务提供统一的云资源按项目管理，以及项目内的资源管理、成员管理。 您可以选择默认的企业项目“default”或其他已有的企业项目。如果要创建新的企业项目，请登录企业管理控制台进行创建，详细操作请参考《企业管理用户指南》。	-
现在配置	单击“现在配置”，呈现添加标签。 添加标签具体请参考 管理通道标签 。	-
暂不配置	暂不配置任何信息。	-
标签	标签是通道的标识。为通道添加标签，可以方便用户识别和管理拥有的通道资源。	-

步骤4 单击“立即购买”，弹出“规格确认”页面。

步骤5 单击“提交”，完成通道接入。

----结束

2.3 步骤 2：准备 DIS 应用开发环境

用户开发DIS应用程序前，首先需要安装和配置应用开发环境。获取SDK和样例工程，并导入到用户的开发环境中。

前提条件

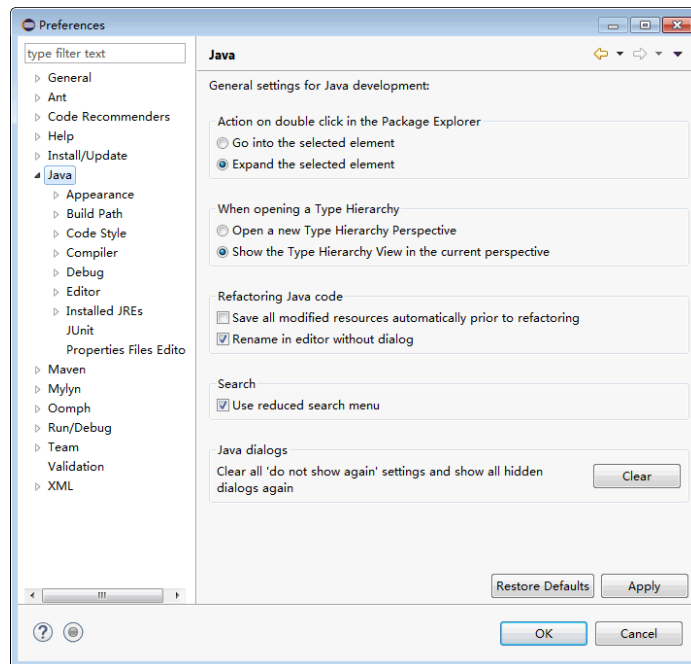
- JDK(1.8版本或以上版本)工具已安装成功。
- Eclipse工具已安装成功。

操作步骤

步骤1 Eclipse中配置JDK。

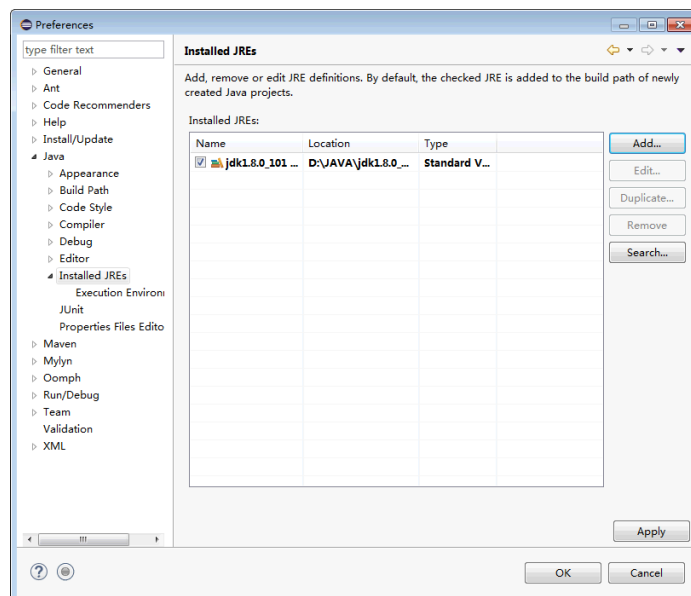
1. 打开Eclipse工具，选择“Window > Preferences”，弹出“Preferences”窗口。
2. 在左侧菜单栏单击“Java”，显示如[图2-1](#)所示内容，选择相关配置，单击“OK”。

图 2-1 Preferences



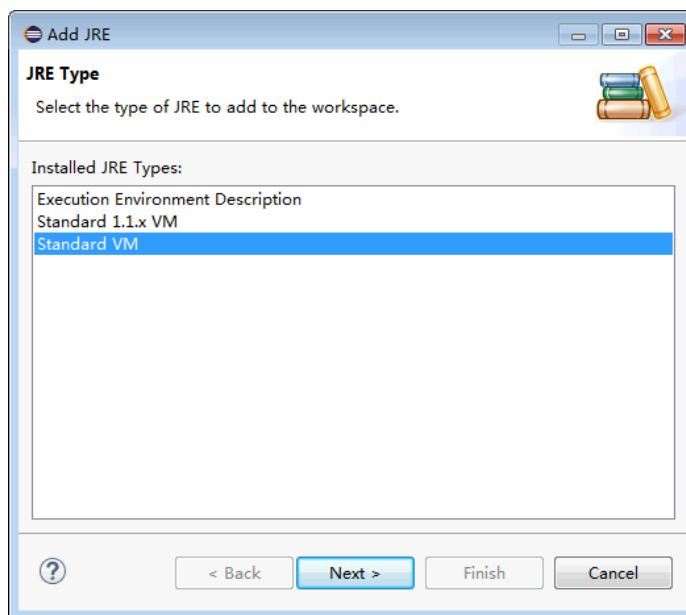
3. 在左侧菜单栏选择“Java > Installed JREs”配置JDK环境变量，显示如图2-2所示。
 - 右侧窗口中显示已配置好的JDK变量，执行步骤1.3.a完成JDK变量配置。
 - 如需配置多个不同的变量对应不同版本的JDK，请执行步骤1.3.b ~ 步骤1.3.d。

图 2-2 Installed JREs



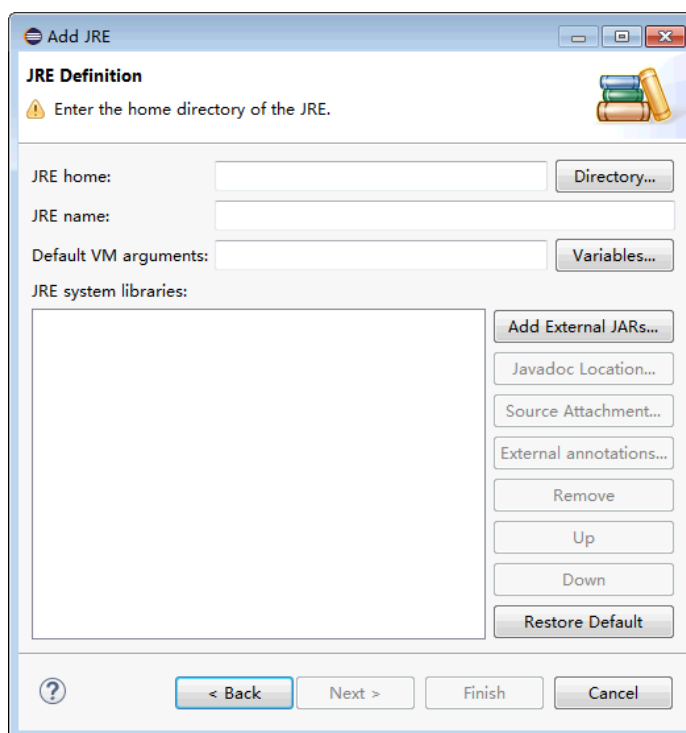
- a. 勾选已安装的JDK，单击“OK”。
- b. 单击“Add”按钮，弹出“Add JRE”窗口，如图2-3所示。

图 2-3 JRE Type



- c. 选择一个JRE类型，单击“Next”，弹出如图2-4所示窗口。

图 2-4 JRE Definition



- d. 配置JDK基本信息，单击“Finish”完成配置。
- JRE home: JDK安装路径。
 - Default VM arguments: JDK运行参数。

步骤2 下载资源包。

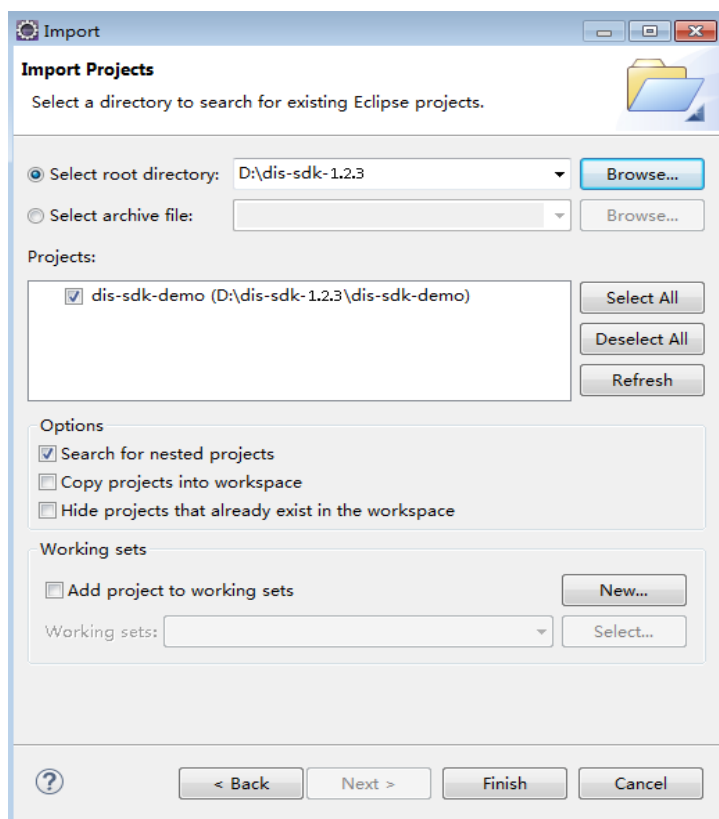
在<https://github.com/huaweicloud/huaweicloud-sdk-java-dis>中下载DIS的Java SDK压缩包。

在DIS SDK桶中获取“huaweicloud-sdk-dis-java-X.X.X.zip”压缩包，其中包含了示例工程demo包。

步骤3 导入Eclipse项目。

1. 打开Eclipse。选择“File > Import”弹出“Import”窗口。
2. 选择“Maven > Existing Maven Projects”，单击“Next”，进入“Import Maven Projects”页面。
3. 单击“Browse”按钮，根据实际情况选择“dis-sdk-demo”样例工程的存储位置，勾选样例工程，如图2-5所示。

图 2-5 Import Maven Projects



4. 单击“Finish”完成项目导入。

步骤4 配置Demo工程。

1. 配置项目编码为“UTF-8”。
 - a. 在左侧导航栏“Project Explorer”中右键单击所需工程，选择“Properties”，进入“Properties for dis-sdk-demo”页面。
 - b. 左侧页签栏选择“Resource”，右侧对话框显示“Resource”页面。
 - c. 在“Text file encoding”栏中选择“Other”，单击下拉框选择“UTF-8”。
 - d. 单击“Apply and Close”完成编码配置。
2. 添加JDK。
 - a. 在左侧导航栏“Project Explorer”中右键单击所需工程，选择“Properties”，进入“Properties for dis-sdk-demo”页面。

- b. 左侧页签栏选择“Java Build Path”，右侧对话框显示“Java Build Path”页面。
- c. 在“Java Build Path”页面选择“Libraries”页签，单击“Add Library”，弹出“Add Library”对话框。
- d. 选择“JRE System Library”，单击“Next”确认“Workspace default JRE”为jdk1.8及以上版本。
- e. 单击“Finish”退出“Add Library”对话框。
- f. 单击“Apply and Close”完成JDK添加。

步骤5 初始化DIS客户端实例。其中，“endpoint”，“ak”，“sk”，“region”，“projectId”信息请参见[获取认证信息](#)。

----结束

2.4 步骤 3：发送数据到 DIS

功能简介

将用户本地数据通过DIS通道不断上传至DIS服务。

说明

目前数据支持存储至DIS和对象存储服务（Object Storage Service，简称OBS）MapReduce服务（MapReduce Service，简称MRS）、数据湖探索（Data Lake Insight，简称DLI），具体存储位置在[新增转储任务](#)的“数据转储”中配置。

DIS为临时存储器，存储在DIS中的数据最长保留时间为[步骤3](#)中配置的“生命周期”的值。

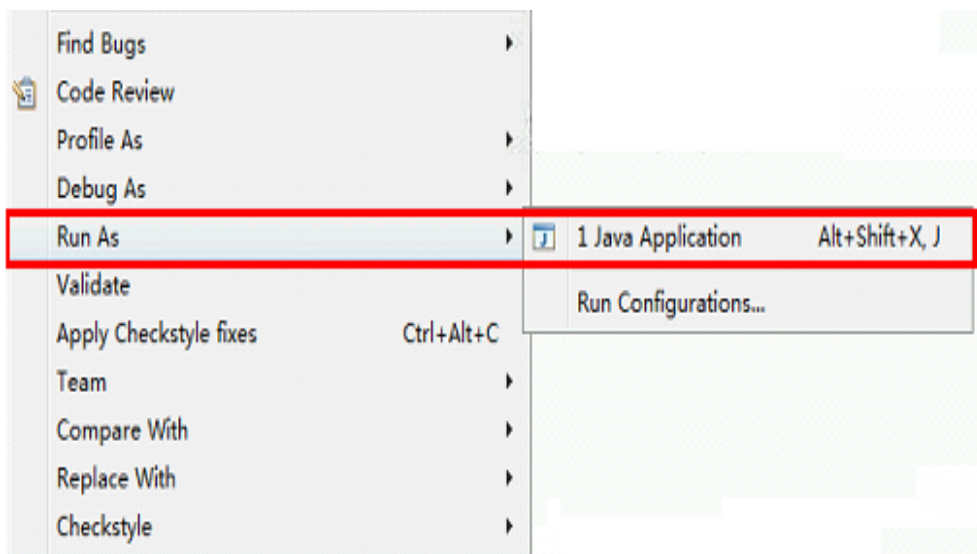
样例代码

样例工程为[DIS SDK桶](#)中下载的“huaweicloud-sdk-dis-java-X.X.X.zip”压缩包中“\dis-sdk-demo\src\main\java\com\bigdata\dis\sdk\demo”路径下“ProducerDemo.java”文件。

运行程序

程序开发完成后，右键选择“Run As > 1 Java Application”运行程序，如[图2-6](#)所示。

图 2-6 运行上传数据程序



数据上传过程中可在Console控制台查看数据上传通道量信息。出现类似信息表示数据上传成功。

```
14:40:20.090 [main] INFOcom.bigdata.dis.sdk.DISConfig - get from classLoader
14:40:20.093 [main] INFODEMOT - ===== BEGIN PUT =====
14:40:21.186 [main] INFOcom.bigdata.dis.sdk.util.config.ConfigurationUtils - get from classLoader
14:40:21.187 [main] INFOcom.bigdata.dis.sdk.util.config.ConfigurationUtils - propertyMapFromFile size : 2
14:40:22.092 [main] INFOcom.bigdata.dis.sdk.demo.ProducerDemo - Put 3 records[3 successful / 0 failed].
14:40:22.092 [main] INFOcom.bigdata.dis.sdk.demo.ProducerDemo - [hello world.] put success, partitionId
[shardId-0000000000], partitionKey [964885], sequenceNumber [0]
14:40:22.092 [main] INFOcom.bigdata.dis.sdk.demo.ProducerDemo - [hello world.] put success, partitionId
[shardId-0000000000], partitionKey [910960], sequenceNumber [1]
14:40:22.092 [main] INFOcom.bigdata.dis.sdk.demo.ProducerDemo - [hello world.] put success, partitionId
[shardId-0000000000], partitionKey [528377], sequenceNumber [2]
14:40:22.092 [main] INFOcom.bigdata.dis.sdk.demo.ProducerDemo - ===== PUT OVER =====
```

2.5 步骤 4: 从 DIS 获取数据

功能简介

从DIS服务中下载数据。

样例代码

样例工程为**DIS SDK桶**中下载的“huaweicloud-sdk-dis-java-X.X.X.zip”压缩包中“\dis-sdk-demo\src\main\java\com\bigdata\dis\sdk\demo”路径下“ConsumerDemo.java”文件。

运行程序

出现类似信息表示下载数据成功：

```
14:55:42.954 [main] INFOcom.bigdata.dis.sdk.DISConfig - get from classLoader
14:55:44.103 [main] INFOcom.bigdata.dis.sdk.util.config.ConfigurationUtils - get from classLoader
14:55:44.105 [main] INFOcom.bigdata.dis.sdk.util.config.ConfigurationUtils - propertyMapFromFile size : 2
14:55:45.235 [main] INFOcom.bigdata.dis.sdk.demo.ConsumerDemo - Get stream
streamName[partitionId=0] cursor success :
```

```
eyJnZXRJdGVyYXRvcjBhcmFtlj7InN0cmVhbS1uYW1lIjoizGlzLTEzbW9uZXkiLCJwYXJ0aXRpb24taWQiOiOilwliwiY
3Vyc29yLXR5cGUiOiBjVBF9TRVFVRU5DRV9OVU1CRVliLCJzdGFydGluZy1zZXF1ZW5jZS1udW1iZXliOiIxMDY4O
Tcylu0slmdlbnVYXRlVGlzXN0YW1wljoxNTEzNjY2NjMxMTYxfQ
14:55:45.305 [main] INFOcom.bigdata.dis.sdk.demo.ConsumerDemo - Get Record [hello world.],
partitionKey [964885], sequenceNumber [0].
14:55:45.305 [main] INFOcom.bigdata.dis.sdk.demo.ConsumerDemo - Get Record [hello world.],
partitionKey [910960], sequenceNumber [1].
14:55:46.359 [main] INFOcom.bigdata.dis.sdk.demo.ConsumerDemo - Get Record [hello world.],
partitionKey [528377], sequenceNumber [2].
```

2.6 获取认证信息

获取 AK/SK

AK/SK (Access Key ID/Secret Access Key) 是用户调用接口的访问密钥。由用户在 IAM 中创建，可在“我的凭证 > 访问密钥”页面下载生成。

获取项目 ID

项目 ID 表示租户的资源。用户可在“我的凭证 > > API 凭证”页面下查看项目列表中不同 Region 对应的项目 ID。

获取 region 和 endpoint

请参见[地区和终端节点](#)。

2.7 连接 OBS

介绍

DIS 可以向对象存储服务 (Object Storage Service, 简称 OBS) 上传数据。

前提条件

已参考[IAM 中创建 DIS 委托](#)创建 IAM 委托，授权 DIS 服务去访问用户的 OBS。

数据转储

用户在[新增转储任务](#)时可设置“数据转储地址”。当“数据转储”设置为“OBS”时，DIS 会将通道数据周期性导入 OBS。

2.8 IAM 中创建 DIS 委托

介绍

用户创建 DIS 通道，选择将数据转储到对象存储服务 (Object Storage Service, 简称 OBS)、MapReduce 服务 (MRS) 集群、数据湖探索 (Data Lake Insight, 简称 DLI) 中，需要通过创建 IAM 委托授权 DIS 服务去访问用户的 OBS、MRS、或 DLI 资源。

创建委托

- 步骤1** 使用注册账户登录管理控制台。
- 步骤2** 选择“管理与部署 > 统一身份认证服务”，进入统一身份认证服务页面。
- 步骤3** 在“委托”页面，单击“创建委托”。
- 步骤4** 填选相应的委托信息，单击“确定”。

表 2-2 创建委托

参数	说明
委托名称	长度不超过64位，且不可为空。
委托类型	必须选择“云服务”。
云服务	单击“选择”，在“选择云服务”的弹出框中选择“DIS”，单击“确定”。
持续时间	选择“永久”。 说明 目前仅支持配置为永久，配置其他值可能导致委托授权失败。
描述	委托信息，长度为0~255位。
权限选择	权限修改：单击“操作”列的“修改”，在弹出“策略”对话框。在“可选择策略”中勾选对应策略，单击“确定”。 说明 创建委托完成后无法修改策略。

----结束

3 管理通道

3.1 通道列表简介

通道列表中可查看当前用户的已创建的所有通道信息。通道信息包括：

- 名称/ID：用户发送或者接收数据时，需要指定通道名称，通道名称不可重复。通道名称由英文字母、数字、中划线和下划线组成。长度为1~64个字符。
- 状态：通道的运行状态。
- 通道类型：普通和高级。
 - 普通通道单分区容量：最高发送速度可达1MB/秒或1000条记录/秒（达到任意一种速度上限才会被限流），最高提取速度可达2MB/秒，单次请求的记录总大小不能超过1MB（不包含partitionKey数据大小）。
 - 高级通道单分区容量：最高发送速度可达5MB/秒或2000条记录/秒（达到任意一种速度上限才会被限流），最高提取速度可达10MB/秒，单次请求的记录总大小不能超过5MB（不包含partitionKey数据大小）。
- 分区数量：分区是DIS数据通道的基本吞吐量单位。通道的多个分区可以并发进行数据传输，以提升效率。
- 源数据类型：BLOB、JSON、CSV。
- 生命周期（小时）：存储在DIS中的数据保留的最长时间，超过此时长数据将被清除。取值范围：24~72的整数。单位：小时。
- 创建时间：显示通道创建的时间。格式为：yyyy/MM/dd HH:mm:ss GMT。其中，yyyy表示年份，MM表示月份，dd表示日期，HH表示小时，mm表示分钟，ss表示秒，GMT表示时区。例如：2017/05/09 08:00:00 GMT+08:00。
- 计费模式：目前仅支持按需付费方式。
- 操作：当前操作列表支持删除通道、查看转储任务和变更源数据类型的操作。

3.2 查看通道监控信息

用户可以通过控制台查看通道的监控信息，支持按照App维度监控App在通道中消费的数据信息。

步骤1 使用注册账户登录[DIS控制台](#)。




- 步骤2** 单击管理控制台左上角的 ，选择区域和项目。
- 步骤3** 在左侧列表栏中选择“通道管理”。
- 步骤4** 单击需要查看监控信息的通道名称，进入监控页面。
- 步骤5** 根据实际情况在“通道管理”页面选择“通道监控”或“分区监控”页签，查看各监控项情况。监控信息参数说明如表3-1所示。其中，通道基本信息的参数说明请参见表2-1。

表 3-1 DIS 监控信息参数说明

参数	说明
时间范围	<ul style="list-style-type: none"> 选择查看监控信息的时间段，可查看所选时间范围内的监控信息。 取值范围： <ul style="list-style-type: none"> - 1h - 3h - 12h 可自定义查看监控信息的时间段。 <ul style="list-style-type: none"> - 单击“自定义”页签后的 ，分别设置开始时间和结束时间。 - 其中，结束时间不能晚于当前的系统时间。 - 开始时间与结束时间的差值不超过72h。
分区监控	
分区编号	流分区编号，默认从0开始。取值方式：从下拉框选择。
该分区的总输入/输出流量（KB/秒）	用户指定时间范围内，指定分区的输入/输出流量。单位：KB/s。
该分区的总输入/输出记录数（个/秒）	用户指定时间范围内，指定分区的输入/输出记录数。单位：个/秒。
通道监控	
总输入/输出流量（KB/秒）	用户指定时间范围内，指定通道的输入/输出流量。单位：KB/s。
总输入/输出记录数（个/秒）	用户指定时间范围内，指定通道的输入/输出记录数。单位：个/秒。
上传/下载请求成功次数（个/秒）	用户指定时间范围内，指定通道的上传/下载请求成功次数。单位：个/秒。
因流控拒绝的上传/下载请求次数（个/秒）	用户指定时间范围内，指定通道因流控拒绝的上传/下载请求次数。单位：个/秒。
上传/下载请求平均处理时间（毫秒/个）	用户指定时间范围内，指定通道的上传/下载请求平均处理时间。单位：毫秒/个。


步骤6 在监控指标视图右上角，单击  可放大查看监控指标视图详情。

----结束


3.3 变更源数据类型

源数据Schema作为通道下特定转储任务进行数据转换的依据，如果没有正确配置将引起数据转换失败从而导致转储任务异常。您可以当前就为通道配置源数据Schema，也可后期创建转储任务时再配置。您还可以在通道详情页面对已配置的源数据Schema进行修改。

步骤1 使用注册账户登录DIS控制台。

步骤2 单击管理控制台左上角的 ，选择区域和项目。

步骤3 在左侧列表栏中选择“通道管理”。

1. 单击需要查看的通道名称。进入所选通道的管理页面。
2. 单击“源数据类型”后的 ，从下拉框中选择对应的源数据类型，可修改创建通道时已设置的源数据类型。或者选择待修改源数据类型通道对应的操作列，选择“更多 > 变更源数据类型”，弹出变更源数据类型对话框，修改创建通道时已设置的源数据类型。

说明

- “源数据类型”为“BLOB”、“JSON”、“CSV”的通道，当该通道无转储任务，才可修改源数据类型。
- 已配置源数据Schema的通道，更改了源数据类型后，已有源数据Schema将失效且无法恢复，需要重新配置。

----结束

3.4 管理源数据 Schema

源数据Schema，即用户的JSON或CSV数据样例，用于描述JSON或CSV数据格式。DIS可以根据此JSON或CSV数据样例生成Avro schema，将通道内上传的JSON或CSV数据转换为Parquet或CarbonData格式。

创建源数据Schema有如下三个入口：

- 创建通道同时开启“Schema开关”，创建源数据Schema，参见图3-1。
- 创建通道时，关闭“Schema开关”。待通道创建成功后，选择“通道管理”页签，单击已创建的通道名称，进入所选通道的管理页面。选择“源数据类型”后的“创建源数据Schema”进行创建，参见图3-2。
- 创建通道时，关闭“Schema开关”。待通道创建成功后，选择“通道管理”页签，单击已创建的通道名称，进入所选通道的管理页面。选择“转储任务”页签，单击“添加转储任务”按钮，在弹出的“添加转储任务”页面进行创建，参见图3-3。

图 3-1 创建 Schema1



图 3-2 创建 Schema2

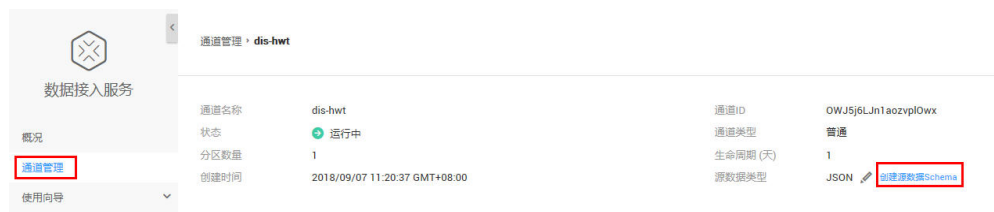
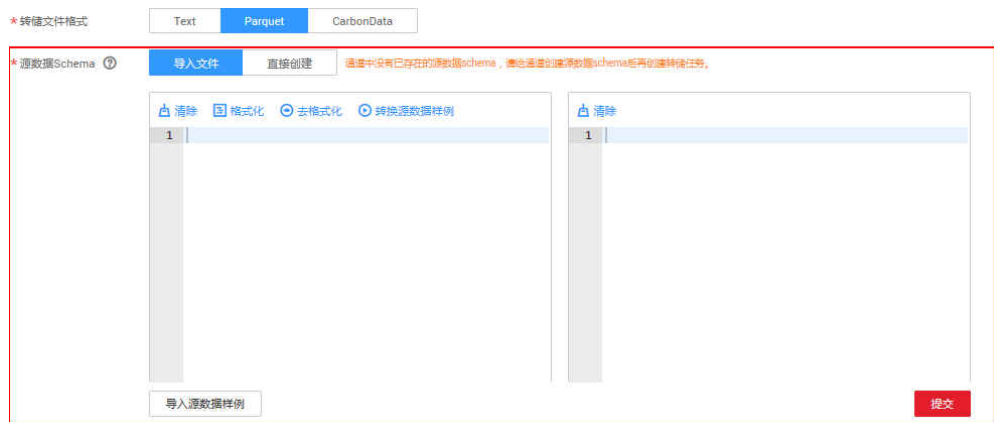


图 3-3 创建 Schema3

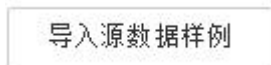


创建源数据 Schema（导入文件方式）

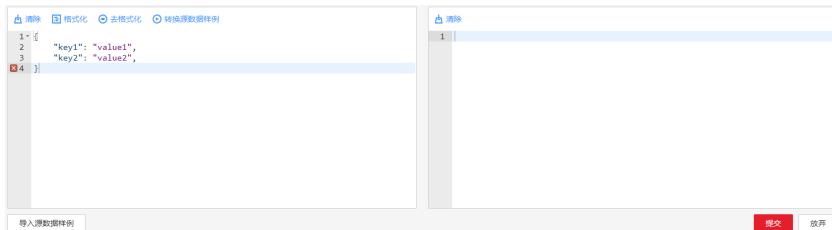
找到创建源数据Schema入口后，按照如下方法创建源数据Schema：

步骤1 单击“源数据Schema”后的“导入文件”。

步骤2 在左侧文本框中输入JSON或者CSV格式的源数据样例，也可单击



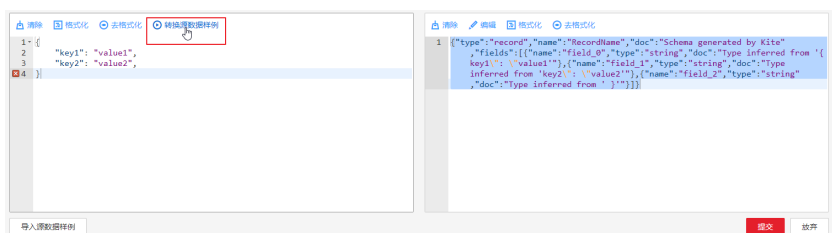
导入源数据样例。例如：



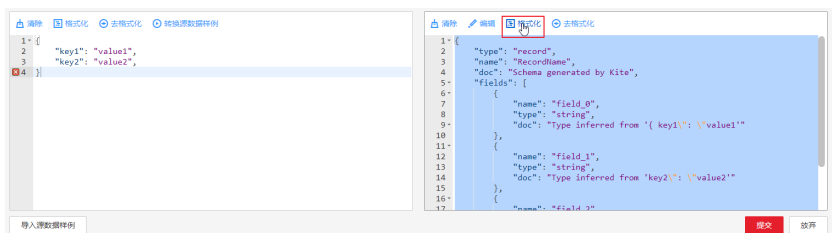
说明

导入源数据样例时，仅支持导入“.txt”，“.json”，“.csv”和“.java”的文件格式。

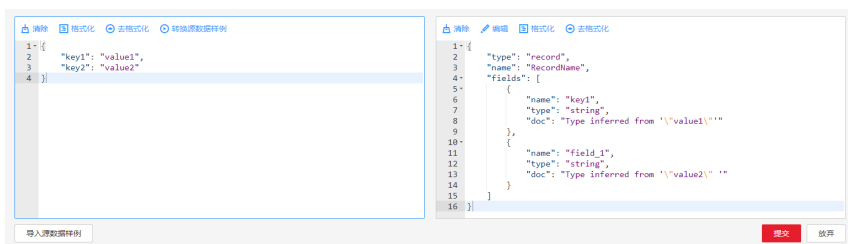
步骤3 左侧文本框中单击 ，可在右侧文本框中根据源数据样例生成Avro schema。例如：



步骤4 右侧文本框中单击 ，可修改已生成的Avro schema。例如：



步骤5 文本框中单击“格式化”，可格式化解析数据。例如：



步骤6 文本框中单击 ，可删除源数据样例。

----结束

创建源数据 Schema（直接创建方式）

找到创建源数据Schema入口后，按照如下方法创建源数据Schema：

步骤1 单击“源数据Schema”后的“直接创建”。

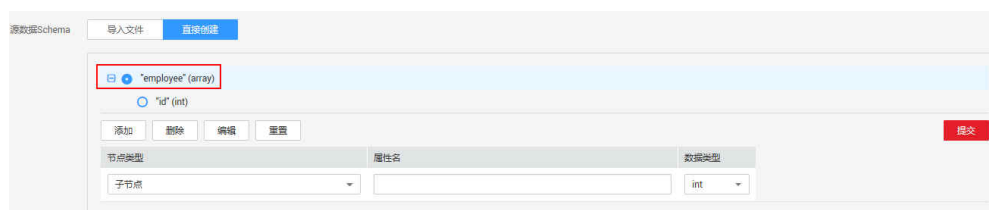
步骤2 配置“属性名”和“数据类型”后，单击“添加”，如图3-4所示，添加根节点。

图 3-4 直接创建源数据 Schema-1



步骤3 根节点添加完成后，选中已创建的根节点，按照同样的方法，配置“属性名”和“数据类型”，添加子节点。

图 3-5 直接创建源数据 Schema-2



📖 说明

- 选中根节点或者子节点前的复选框，单击“删除”，可将节点删除。
- 选中根节点或者子节点前的复选框，单击“编辑”，可对已创建的节点属性进行编辑。
- 单击“重置”，可删除所有节点。

步骤4 单击“提交”，源数据Schema创建成功。

----结束

修改源数据 Schema

📖 说明

已创建了源数据Schema的通道，若该通道下存在转储任务，则不允许修改已有的源数据Schema。

步骤1 使用注册账户登录[DIS控制台](#)。

步骤2 单击管理控制台左上角的📍，选择区域。

步骤3 在左侧列表栏中选择通道管理。

1. 单击通道名称。进入所选通道的管理页面。
2. 单击“源数据类型”后的“查看已有源数据Schema”。
3. 弹出源数据Schema文本框，单击✎，修改源数据Schema。

图 3-6 修改源数据 Schema



说明

当通道中存在转储任务，修改源数据Schema可能导致通道内未转储完成的数据无法被成功转储。

4. 修改完成后，单击“提交”，保存修改结果。单击“放弃”，不对源数据Schema进行修改。

---结束

3.5 管理通道标签

标签是通道的标识。为通道添加标签，可以方便用户识别和管理拥有的通道资源。

您可以在创建通道时添加标签，也可以在通道创建完成后，在通道的详情页添加标签，您最多可以给通道添加10个标签。

标签共由两部分组成：“标签键”和“标签值”，其中，“标签键”和“标签值”的命名规则如表3-2所示。

表 3-2 标签命名规则

参数	规则	样例
标签键	不能为空。 对于同一个通道，标签键唯一。 长度不超过36个字符，不能包含“=”，“*”，“<”，“>”，“\”，“，”，“ ”，“/”，且首尾字符不能为空格。	Organization
标签值	标签值可以为空 长度不超过43个字符，不能包含“=”，“*”，“<”，“>”，“\”，“，”，“ ”，“/”，且首尾字符不能为空格。	Apache

为通道增加标签

在购买接入通道页，为通道增加标签。

1. 登录管理控制台。
2. 在控制台页面中选择“服务列表 > 大数据 > 数据接入服务 DIS”。
3. 单击“购买接入通道”，进入“购买接入通道”页面。
4. “高级配置”页签，选择“现在配置”，展开标签页。

输入新添加标签的键和值。

系统支持添加多个标签，最多可添加10个标签，并取各个标签的交集，对目标通道进行搜索。

说明

您也可对现有通道增加标签，详见[管理标签](#)。

搜索目标通道

在现有通道列表页，按标签键或标签值搜索目标通道。

1. 登录管理控制台。
2. 选择“EI企业智能 > 数据接入服务”。
3. 单击现有通道列表右上角的“标签搜索”，展开查询页。
4. 输入待查询通道的标签。

标签键或标签值可以通过下拉列表中选择，当标签键或标签值全匹配时，系统可以自动查询到目标通道。当有多个标签条件时，会取各个标签的交集，进行通道查询。

5. 单击“搜索”。

系统根据标签键或标签值搜索目标通道。

管理标签

在现有通道的标签页，执行标签的增、删、改、查操作。

1. 登录管理控制台。
2. 选择“EI企业智能 > 数据接入服务”。
3. 在现有通道列表中，单击待管理标签的通道名称。
系统跳转至该通道详情页面。
4. 选择“标签”页签，对通道的标签执行增、删、改、查。

- 查看

在“标签”页，可以查看当前通道的标签详情，包括标签个数，以及每个标签的键和值。

- 添加

单击左上角的“添加标签”，在弹出的“添加标签”窗口，输入新添加标签的键和值，并单击“确认”。

- 修改

单击标签所在行“操作”列下的“编辑”，在弹出的“编辑标签”窗口，输入修改后标签的值，并单击“确认”。

- 删除

单击标签所在行“操作”列下的“删除”，如果确认删除，在弹出的“删除标签”窗口，单击“确认”。


3.6 管理 App

App表示应用程序标识符。当多个应用程序分别消费同一通道的数据时，为区分不同应用程序的消费检查点，使用App作为标识。

您可以创建App，也可进入通道管理页面，查看接入该通道的App详情。

创建 App

步骤1 使用注册账户登录[DIS控制台](#)。

步骤2 单击管理控制台左上角的 ，选择区域和项目。


步骤3 在左侧列表栏中选择“App管理”。

进入App管理页面，单击“创建App”，输入对应的名称，完成创建。

---结束

查看 App

步骤1 使用注册账户登录[DIS控制台](#)。

步骤2 单击管理控制台左上角的 ，选择区域和项目。

步骤3 在左侧列表栏中选择“通道管理”。

步骤4 单击需要查看的通道名称，进入所选通道的管理页面。

步骤5 单击“Apps”，可查看到接入该通道的所有App。

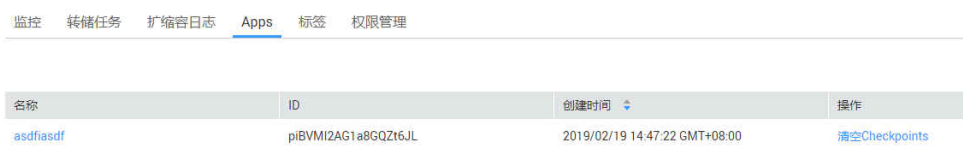
可查看接入该通道的App名称，ID和创建时间。

您也可通过单击“清空Checkpoints”，将App的所有Checkpoints清零。

说明

Checkpoint，消费检查点。应用程序消费数据时，记录已消费数据的最新序列号作为检查点。当重新消费数据时，可根据此检查点继续消费。

图 3-7 查看 Apps



名称	ID	创建时间	操作
asdfasdf	piBVMi2AG1a8GQZt6JL	2019/02/19 14:47:22 GMT+08:00	清空Checkpoints

步骤6 单击实际的App名称，可查看App对该通道数据的消费详情。

图 3-8 查看 App 详情

App: asdfiasdf

Checkpoint指消费检查点。应用程序消费数据时，记录已消费数据的最新序列号作为检查点。当重新消费数据时，可根据此检查点继续消费。


分区编号	分区状态	最早偏移量	最新偏移量	Checkpoint
0	运行中	6	6	1
1	运行中	1	1	-1
2	运行中	0	0	-1

----结束

3.7 授权管理

通过添加授权策略，可实现被授权的其他用户拥有DIS通道的上传和下载权限。

步骤1 使用注册账户登录DIS控制台。

步骤2 单击管理控制台左上角的 ，选择区域和项目。

步骤3 在左侧列表栏中选择“通道管理”。

步骤4 单击通道名称，进入所选通道的管理页面。

步骤5 选择“授权管理”，单击“添加授权策略”，选择授权模式，再在“被授权用户”文本框中，设置用户信息。

说明

- 支持通配符"*"，表示授权所有账号；
- 支持添加多账号，用","隔开
- 支持授权某账号下的特定用户，输入账号名，单击"查询用户"按钮，选择用户。

图 3-9 添加权限

×

添加授权策略

授权模式 上传权限 下载权限 通道详情

被授权用户 ?

确认
取消

----结束

3.8 调试通道

用户在创建通道成功后，可在界面进行简单的上传和下载操作，验证通道的可用性。

通道分区状态为ACTIVE（可用）时，同时支持上传和下载。

通道分区状态为DELETED（删除中）时，仅支持下载，不支持上传。


- 步骤1** 使用注册账户登录**DIS控制台**。
- 步骤2** 单击管理控制台左上角的  ，选择区域和项目。
- 步骤3** 在左侧列表栏中选择“通道管理”。
- 步骤4** 单击通道名称，进入所选通道的管理页面。
- 步骤5** 单击“通道调试”，选择对应分区后的上传或下载操作，进行数据的上传和下载。

图 3-10 上传下载数据



分区ID	分区状态	序列号	操作
shardId-0000000000	 运行中	[0 : 0]	上传 下载

- 步骤6** 单击“上传”，在上传文本框中输入上传内容后，确认上传。
系统提示上传数据成功，并在界面回显当前数据上传成功记录的序列号。

图 3-11 上传数据



上传

通道名称 dis-hwt

分区ID shardId-0000000000

消息内容

0/2048

[上传](#)

[确认](#) [取消](#)

- 步骤7** 单击“下载”，选择游标类型并输入对应的序列号，确认完成。下载成功后，您可以文本形式获取响应体。

说明

输入的序列号，需要在分区的数据有效范围内。分区的数据有效范围，可以通过调用 describeStream(查询通道详情)接口获取，其sequenceNumberRange代表数据有效范围，第一个值为最老数据的sequenceNumber，最后一个值为下一条上传数据的sequenceNumber(最新数据的sequenceNumber为此值-1)

图 3-12 下载数据

下载

通道名称 `dis-hwt3`

游标类型 ? AT_SEQUENCE_NUMBER TRIM_HORIZON

分区ID `shardId-0000000000`

序列号 ?

确认 取消

----结束

3.9 弹性伸缩分区

用户在创建通道成功后，随着业务的发展和变化，对通道容量有了新的需求。弹性伸缩分区可以对已经创建成功的通道进行分区扩容或者分区缩容以满足用户这一需求，支持自动和手动扩缩容两种方式。

约束限制

- 每个通道在一小时内仅可操作实现10次自动扩容、10次手动扩容、1次缩容（包含手动缩容和自动缩容）操作。
- 进行弹性伸缩分区后，有如下注意事项：
 - 上传数据时，不建议设置数据的PartitionKey，DIS会自动根据通道分区的数量将数据均匀散列到多个分片中。如果设置数据的PartitionKey，可能会导致数据倾斜，产生通道限流。
 - 下载数据时，需要定期的使用descriptStream接口检测通道分区数量的变化，以便DIS可以下载到所有分区的数据。

自动扩缩容

自动扩缩容原理

- 当上一分钟内通道触发流控（即超过通道内分区最大吞吐量开始限流）、且通道上传流量大于通道总带宽80%时，触发自动扩容操作，扩容目标分区数=分区数/0.6，向上取整。
例如，有5个普通分区时，上传总带宽为5MB/秒。当上一分钟通道触发流控后、且通道上传流量达到4MB/秒以上时，触发自动扩容操作，目标分区数为 $5/0.6=8.3$ ，向上取整后为9。
- 当通道上传流量和下载流量均小于30%时，触发自动缩容操作，缩容目标分区数=分区数/2，向下取整。


例如，有5个普通分区时，上传总带宽为5MB/秒，下载总带宽为10MB/秒。当通道上传流量低于1.5MB/秒、且下载流量低于3MB/秒时，触发自动缩容操作，目标分区数为 $5/2=2.5$ ，向下取整后为2。

自动扩缩容规则

- 自动扩缩容间隔时间大于1分钟，且发生扩容（含自动扩容和手工扩容）后2分钟内，不触发自动缩容。发生缩容（包含手动缩容和自动缩容）后的2分钟内，不触发自动扩容。
- 扩容分区时，首先将状态为“DELETED”的分区恢复为“ACTIVE”状态，成为可读写分区。其次将状态为“EXPIRED”的分区恢复为“ACTIVE”状态，成为可读写分区。若前两者恢复后仍不满足扩容需求，系统将新建分区。
- 对已有分区进行缩容操作后，缩容成功的分区不再进行计费也不参与配额控制。在**步骤1：开通DIS通道**中配置的“生命周期”时间内，缩容成功的分区可以读取数据不可写入数据，超过此时间则不可读取/写入数据。

执行自动扩缩容操作

步骤1 使用注册账户登录**DIS控制台**。

步骤2 单击管理控制台左上角的 ，选择区域和项目。

步骤3 按照如下方法进行自动扩缩容。

在左侧列表栏中选择“通道管理”。


1. 在“通道管理”页面中单击需要扩缩容的通道名称。
2. 单击通道详情页面的“自动扩缩容”菜单后的编辑按钮 。
3. 系统弹出“变更自动扩缩容参数”对话框，将自动扩缩容的开关开启。

图 3-13 变更自动扩缩容参数



4. 设置自动扩缩容的分区上限和下限，单击“确认”。

----结束


手动扩缩容

手动扩缩容规则

- 扩容分区时“目标分区数量”需大于当前分区数量，且小于等于租户剩余配额与当前分区数量的总和。
- 扩容分区时，首先将状态为“DELETED”的分区恢复为“ACTIVE”状态，成为可读写分区。其次将状态为“EXPIRED”的分区恢复为“ACTIVE”状态，成为可读写分区。若前两者恢复后仍不满足扩容需求，系统将新建分区。
- 缩容分区时“目标分区数量”需要大于等于“1”，且小于当前分区数量。
- 对已有分区进行缩容操作后，缩容成功的分区不再进行计费也不参与配额控制。在**步骤1：开通DIS通道**中配置的“生命周期”时间内，缩容成功的分区可以读取数据不可写入数据，超过此时间则不可读取/写入数据。

执行手动扩缩容操作

步骤1 使用注册账户登录**DIS控制台**。

步骤2 单击管理控制台左上角的 ，选择区域和项目。


步骤3 选择如下任意一种方法进行手动扩缩容。

- 在左侧列表栏中选择“通道管理”。
 - a. 在“通道管理”页面中单击需要扩缩容的通道名称。
 - b. 在通道详情页面的右上角单击“扩缩容”按钮，弹出“变更分区数目”对话框。
 - c. 修改“目标分区数”，单击“确认”。
- 在左侧列表栏中选择。
 - a. 针对待扩缩容的通道，选中“操作”列中的“更多”下拉列表中的“扩缩容”。
 - b. 弹出“变更分区数目”对话框。
 - c. 修改“目标分区数”，单击“确认”。

----结束

查看扩缩容日志

步骤1 使用注册账户登录**DIS控制台**。

步骤2 单击管理控制台左上角的 ，选择区域和项目。

步骤3 在左侧列表栏中选择“通道管理”。

步骤4 单击需要查看的通道名称。进入所选通道的管理页面。

步骤5 选择“扩缩容日志”页签。查看该通道的扩缩容详情。


----结束

3.10 删除通道

说明

通道删除后不再收取费用，也无法恢复，请谨慎操作。

步骤1 使用注册账户登录**DIS控制台**。

步骤2 单击管理控制台左上角的  ，选择区域和项目。

步骤3 在左侧列表栏中选择“通道管理”。

步骤4 在需要删除的通道中单击操作列的“删除”，弹出“删除DIS通道”对话框。

步骤5 单击“确认”，删除选中的通道。

----结束

4 使用 DIS

4.1 检查与配置 DNS 信息

默认情况下，弹性云服务器已经配置了两个外网DNS服务器。

```
# Generated by NetworkManager
search openstacklocal
nameserver 114.114.114.114
nameserver 114.114.115.115
```

如果弹性云服务器未绑定弹性IP，或者用户不同意使用该弹性IP来传输AEI_Register.sh工具和fisclient程序在使用过程中产生的流量，则需要编辑“/etc/resolv.conf”文件，新增DNS服务器。

示例如下，其中XXX.XXX.XXX.XXX为DNS服务器的IP地址。

```
# Generated by NetworkManager
search openstacklocal
nameserver XXX.XXX.XXX.XXX
nameserver 114.114.114.114
nameserver 114.114.115.115
```

说明

新增的DNS服务器地址必须位于所有原有的DNS服务器地址之前。

DNS配置操作在保存“/etc/resolv.conf”文件的修改操作后立即生效。

对“/etc/resolv.conf”文件的修改操作在弹性云服务器重启后会失效，需要重新进行配置。如果用户不希望每次重启弹性云服务器后都重新配置DNS，可以按如下步骤修改虚拟私有云的子网信息，将DNS服务器地址添加到弹性云服务器对应的子网中。

操作步骤

- 步骤1** 使用注册账户登录弹性云服务控制台，进入“云服务器控制台”页面。
- 步骤2** 在页面中单击用户将要使用的弹性云服务器，查看云服务器详情。
- 步骤3** 在云服务器详情页面中，单击“网卡”，打开相应的下拉菜单，查看弹性云服务器的子网名称。
- 步骤4** 在云服务器详情页面中，单击“虚拟私有云”的VPC名称/ID，进入“网络控制台”。

- 步骤5** 在“虚拟私有云”详情页面中单击“VPC名称/ID”对应的虚拟私有云名称，进入虚拟私有云详情页面。
- 步骤6** 在页面左侧选择“子网”，在子网列表中单击“子网名称/ID”，进入虚拟私有云子网页面。
- 步骤7** 单击“网关和DNS”中“DNS服务器地址”后的“修改”按钮，弹出“修改DNS服务器地址”窗口。
- 步骤8** 将DNS服务器地址修改为所需的DNS服务器地址，然后单击“确定”，保存DNS服务器地址的修改。
- 步骤9** 重启弹性云服务器，查看“/etc/resolv.conf”文件的内容，确认其中包含待配置的DNS服务器地址，并且新增DNS服务器地址位于其他DNS服务器地址之前。

```
# Generated by NetworkManager
search openstacklocal
nameserver XXX.XXX.XXX.XXX
nameserver 114.114.115.115
```

说明

对虚拟私有云的子网信息的修改会影响所有使用该子网创建的弹性云服务器。

----结束

4.2 使用 Agent 上传数据

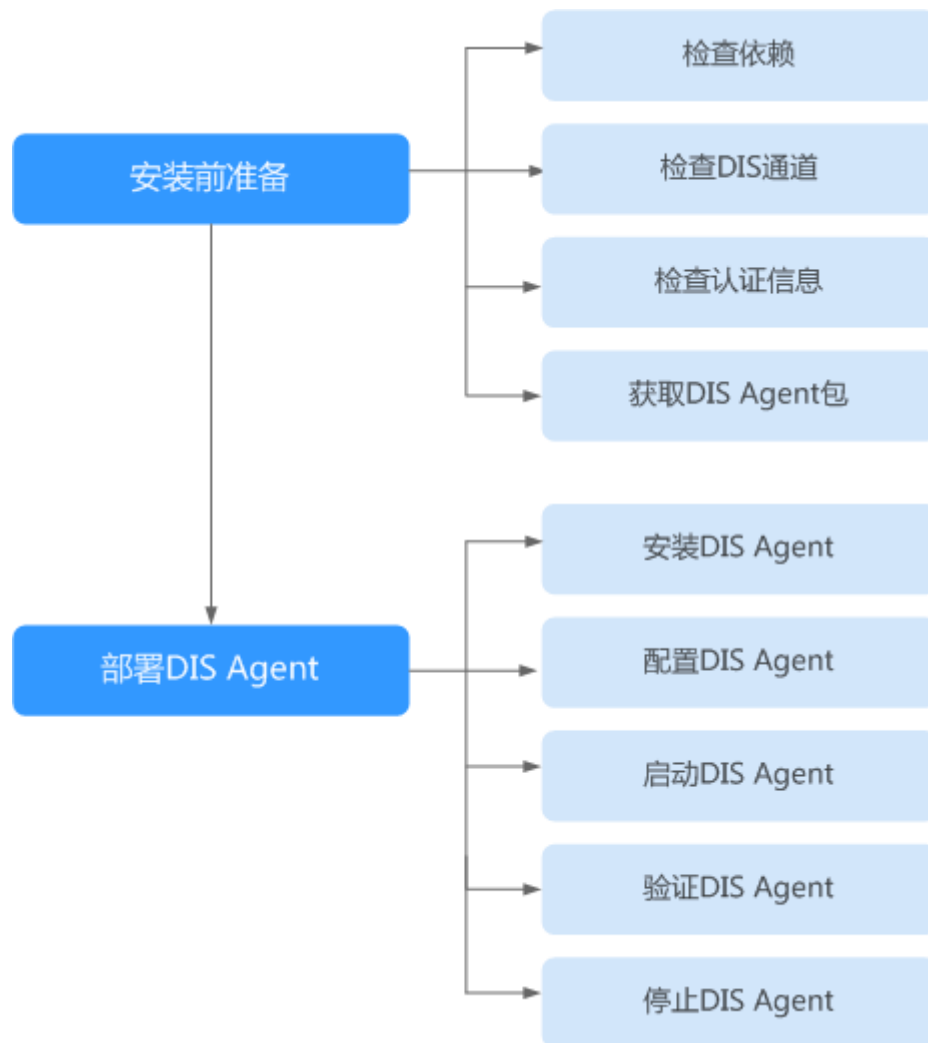
4.2.1 DIS Agent 概述

DIS Agent是数据接入服务（DIS）提供的一个客户端程序，具有如下功能：

持续查询文本文件，实时收集增量数据按分隔符解析并上传到DIS通道(通道源数据类型为BLOB/JSON/CSV)。

DIS Agent安装流程如[图4-1](#)所示。

图 4-1 安装流程



4.2.2 安装前准备

检查依赖

步骤1 服务器类型。

- Linux x86-64 (64位) 服务器，常见的有EulerOS、Ubuntu、Debian、CentOS、OpenSUSE等。
- Windows 7及以上版本。

步骤2 已安装1.8.0及以上版本的Java。

请参见[JRE地址](#)下载JRE。


Linux服务器安装请参考如下步骤：

1. 使用root用户，进入“/opt”目录。
cd /opt
2. 创建目录“jre”。
mkdir -p jre

3. 设置JDK安装目录的权限。
`chmod -R 640 jre/`
4. 将压缩包上传到“jre”目录下，执行如下命令解压JRE安装包。
`tar -zxvf JRE包名.tar.gz`
5. 修改“/etc/profile”配置文件。
 - a. 执行`vim /etc/profile`命令，进入“profile”文件。
 - b. 在“JAVA_HOME”配置项里添加JDK的安装目录，内容如下。
export JAVA_HOME=解压后的jre文件夹路径，请根据实际情况填写
export PATH=\$PATH:\$JAVA_HOME/bin
export CLASSPATH=.:\$JAVA_HOME/lib/rt.jar:\$JAVA_HOME/lib/ext
 - c. 退出编辑模式。
键入“:wq!”保存并退出“profile”文件。
6. 执行如下命令，使JDK配置生效。
`source /etc/profile`

----结束

检查 DIS 通道

- 步骤1 使用注册账户登录[DIS控制台](#)。
- 步骤2 单击管理控制台左上角的，选择区域和项目。
- 步骤3 在左侧列表栏中选择“通道管理”。
- 步骤4 确认有通道用于数据上传且通道状态为“运行中”。

----结束

检查认证信息

- 检查AK/SK
AK/SK (Access Key ID/Secret Access Key)是用户调用接口的访问密钥。由用户在IAM中创建，可在“我的凭证 > 管理访问密钥”页面下载生成。
- 检查项目ID
ProjectID表示租户的资源，每个Region都有一个唯一的项目ID。可在页面查看不同Region对应的项目ID值。

获取 DIS Agent 包

[这里](#)获取“dis-agent-X.X.X.zip”压缩包。

4.2.3 安装 DIS Agent

前提条件

已安装PuTTY工具。

Linux 服务器上安装 DIS Agent

- 步骤1** 使用PuTTY工具登录日志所在服务器，即[检查依赖](#)的服务器。
- 步骤2** 将[获取DIS Agent包](#)中获取的“dis-agent-X.X.X.zip”安装包上传到“/opt”文件夹中。
- 步骤3** 解压“dis-agent-X.X.X.zip”压缩包。
`unzip dis-agent-X.X.X.zip`
- 步骤4** 进入“dis-agent-X.X.X”文件夹。
`cd dis-agent-X.X.X`
- 结束

Windows 服务器上安装 DIS Agent

- 步骤1** 将[获取DIS Agent包](#)中获取的“dis-agent-X.X.X.zip”压缩包保存到本地。
- 步骤2** 解压“dis-agent-X.X.X.zip”压缩包至当前目录。
- 结束

4.2.4 配置 DIS Agent

DIS Agent配置文件格式为“YAML”，各配置项与值之间必须以英文格式的“冒号+空格”形式分隔。

agent.yml文件模板可从“dis-agent”压缩包中获取，内容示例如下。具体配置项说明请参见[表4-1](#)。

```
---
# cloud region id
region: myregion
## The plaintext storage of the AK and SK used for authentication has great security risks
## You are advised to use the /bin/dis-encrypt.sh script to encrypt the AK and SK before storing them
# you ak (get from 'My Credential')
ak: YOU_AK
# you sk (get from 'My Credential')
sk: YOU_SK
# you_key(ency you ak or sk)
encrypt.key: abc
# you project id (get from 'My Credential')
projectId: YOU_PROJECTID
# the dis endpoint
endpoint: https://dis.myregion.cloud.com
# config each flow to monitor file.
flows:
  ### DIS Stream
  - DISStream: YOU_DIS_STREAM_1
    ## only support specified directory, filename can use * to match some files. eg. * means match all file,
    test*.log means match test1.log or test-12.log and so on.
    filePattern: /tmp/*.log
    ## from where to start: 'START_OF_FILE' or 'END_OF_FILE'
    initialPosition: START_OF_FILE
    ## upload max interval(ms)
    maxBufferAgeMillis: 5000

    ### If there are other monitor files, continue to follow the above configuration
    ### another dis stream monitor config, uncomment # if you want to use this feature
  #- DISStream: YOU_DIS_STREAM_2
    # filePattern: /opt/*.log
```

```
# initialPosition: START_OF_FILE
# maxBufferAgeMillis: 5000

### OBS Stream: Upload the matching file to OBS and send the file name to DIS, uncomment # if you
want to use this feature
#- OBSStream: YOU_DIS_STREAM_3
# filePattern: /opt/*.log
# initialPosition: START_OF_FILE
# ## bucket name
# OBSBucket: YOU_OBS_BUCKET_NAME
# ## OBS endpoint
# OBSEndpoint: https://obs.myregion.cloud.com
# ## the directory(using / separated) where the files are stored under the bucket, automatically created
if it does not exist
# dumpDirectory: example/dis/
```

📖 说明

配置完成之后，请将agent.yml中flows下面无用的示例配置删除或者使用#注释（比如只配置了一个DISStream，则将下面的CustomFileStream与另外的DISStream模块删除或者注释）。

Linux 服务器上配置 DIS Agent

- 步骤1** 使用PuTTY 工具登录日志所在服务器。
- 步骤2** 执行cd /opt/dis-agent-X.X.X/命令，进入“dis-agent-X.X.X”文件夹。
- 步骤3** 执行vim conf/agent.yml命令，打开DIS Agent配置文件“agent.yml”，根据实际情况修改各配置项的值并保存，配置项说明请参见表4-1。

表 4-1 agent.yml 配置文件说明

配置项	是否必填	说明	默认值
region	是	DIS服务所在区域。 说明 获取DIS区域请参见 地区和终端节点 。	-
AK	是	用户的Access Key。 说明 支持用户自己加密AK以保证安全，也可以使用明文的AK，如若需要对AK加密，请查看表格下关于AK/SK加密的使用说明。 获取方式请参见 检查认证信息 。	请根据实际情况配置
SK	是	用户的Secret Key。 说明 支持用户自己加密SK以保证安全，也可以使用明文的SK，如若需要对SK加密，请查看表格下关于AK/SK加密的使用说明。 获取方式请参见 检查认证信息 。	请根据实际情况配置

配置项	是否必填	说明	默认值
encrypt.key	否	用户加密时使用key值。 说明 如果用户需要使用加密的AK或者SK，则必须配置该参数(自己在agent.yml文件中添加)，请务必保证加密时使用的key值和此处写的encrypt.key保持一致，否则将会解密失败。	请根据实际情况配置
projectId	是	用户所属区域的项目ID。 获取方式请参见 检查认证信息 。	请根据实际情况配置
endpoint	是	DIS数据网关地址。 格式：https://DIS终端节点。 说明 获取DIS终端节点请参见 地区和终端节点 。	-
body.serialize.type	否	DIS数据包上传格式。（非原始数据格式） <ul style="list-style-type: none"> json：DIS数据包封装为json格式，满足普通使用。 protobuf：DIS数据包封装为二进制格式，可以减少体积约1/3，在数据量较大的情况下推荐使用此格式。 	json
body.compress.enabled	否	是否开启传输数据压缩。	false
body.compress.type	否	开启压缩时选择的数据压缩格式，目前支持的压缩格式如下： lz4 ：综合来看效率最高的压缩算法,更加侧重压缩解压速度,压缩比并不是第一。 zstd ：一种新的无损压缩算法，旨在提供快速压缩，并实现高压缩比。	lz4
PROXY_HOST	否	配置代理IP，请求走代理服务器的需要配置。	请根据实际情况配置
PROXY_PORT	否	配置代理端口。	80
PROXY_PROTOCOL	否	配置代理协议。支持http和https。	http

配置项	是否必填	说明	默认值
PROXY_USERNAME	否	配置代理用户名。	请根据实际情况配置
PROXY_PASSWORD	否	配置代理密码。	请根据实际情况配置

[flows]

监控的文件信息，可同时配置多个监控文件信息。

当前支持如下模式上传：

DISStream:持续监控文本文件，实时收集增量数据按分隔符解析并上传到DIS通道(通道源数据类型为BLOB/JSON/CSV)，配置项说明请参见表4-2。

具体配置格式可以参见版本包中的“agent.yml”的样例。

关于AK/SK加密的使用说明：

参照[安装DIS Agent](#)下载安装dis-agent，利用dis-agent包中bin目录下的脚本对AK和SK进行加密，按照如下所示步骤进行操作（windows环境下）：

1. 进入到dis-agent的bin目录中，右键git bash here运行脚本，示例：`./dis-encrypt.sh {key} {ak}`，即可得到加密后的AK，将其配置在“agent.yml“配置文件中，SK同理。
2. 按上述操作将AK和SK加密之后，将加密后的AK/SK，key全部配置到“agent.yml”即可。

图 4-2 加密示例

```

C:\Users\mingw64\Desktop> cd dis-agent-1.2.4\bin
C:\Users\mingw64\Desktop\dis-agent-1.2.4\bin> ./dis-encrypt.sh abc Z6ATVN8R4T2GHIZKKHKL
Please wait...
Encrypt result: 333630643732326635326266366666643a3a3731316139363864336337613630
6133383363316337376566376134383133333a3a3939383132373639323438323964353736656335
6535333065666135623833656635656163636464313362653134653334656632366366623066396
30353137

C:\Users\mingw64\Desktop> cd dis-agent-1.2.4\bin
C:\Users\mingw64\Desktop\dis-agent-1.2.4\bin> ./dis-encrypt.sh abc 1YZINSp01sPKjkQYqrnpr3gved7mFsK7BFtXJIy8
Please wait...
Encrypt result: 373937353532336436343563396633333a3a6638383762343930343331323334
3262626135643838366338313162626165663a3a3862633737326561623533383861333762376537
37623133653664653061363837333164353934626638616231633736613435376664303065356465
336234363965613666626330363531633133613030343736396130336333353534336366
    
```

表 4-2 DISStream 配置项说明

配置项	是否必填	说明	默认值
DISStream	是	DIS 通道名称。 将“filePattern”所匹配到的文件内容按分隔符解析并上传到此通道。	请根据实际情况配置

配置项	是否必填	说明	默认值
filePattern	是	<p>文件监控路径，只能监控一个目录下的文件，无法递归目录监控。</p> <p>如果要监控多个目录，可以在flows下面配置多个“DISStream”，文件名可使用“*”进行匹配。</p> <ul style="list-style-type: none"> “/tmp/*.log”表示匹配“/tmp”目录下所有以“.log”结尾的文件。 “/tmp/access-*.log”表示匹配“/tmp”目录下所有以“access-”开头，以“.log”结尾的文件。 Windows上路径范例为“D:\logs*.log”。 	请根据实际情况配置
directoryRecursionEnabled	否	<p>是否查找子目录</p> <ul style="list-style-type: none"> false: 不递归查找子目录，只匹配根目录下的文件 true: 递归查找所有子目录。如filePattern配置为/tmp/*.log，此时可以匹配到/tmp/one.log, /tmp/child/two.log, /tmp/child/child/three.log 	false
initialPosition	否	<p>监控起始位置。</p> <ul style="list-style-type: none"> END_OF_FILE: 开始启动时不解析当前匹配的文件，而是从新增文件或新增的内容开始按分隔符解析并上传。 START_OF_FILE: 将“filePattern”配置的所有匹配文件按照修改时间，从旧到新按分隔符解析并上传到DIS服务。 	START_OF_FILE
maxBufferAgeMillis	否	<p>最长上传等待时间。</p> <p>单位：毫秒</p> <ul style="list-style-type: none"> 记录队列满则立即上传。 记录队列未满，等待此配置项配置的时间后上传到DIS服务。 	5000
maxBufferSizeRecords	否	<p>记录队列缓存的最大记录数，如果队列达到此值则立刻上传这批数据。</p>	500

配置项	是否必填	说明	默认值
partitionKeyOption	否	<p>每条记录会携带一个PartitionKey，相同PartitionKey的记录会分配到同一个分区。此配置项可设置每条记录的PartitionKey值，取值如下：</p> <ul style="list-style-type: none"> ● RANDOM_INT: PartitionKey的值为随机数字的字符串，记录均匀分布在每个分区。 ● FILE_NAME: PartitionKey的值为文件名称字符串，记录分布在特定的一个分区中。 ● FILE_NAME,RANDOM_INT: PartitionKey的值为文件名称字符串与随机数字字符串的组合体，以英文逗号分隔，记录携带所属的文件名并均匀分布在所有分区。 	RANDOM_INT
recordDelimiter	否	<p>每条记录之间的分隔符。</p> <p>取值范围：任意一个字符，且包含在双引号内。</p> <p>取值不可为空，即该配置项不可配置为“”。</p> <p>说明</p> <p>如果取值为特殊字符，使用反斜杠（\）转义，如分隔符为引号（"），可配置为"\\"，如果为反斜杠（\），可配置为"\\\"。</p> <p>如果为控制字符如STX（正文开始），可配置为"\u0002"。</p>	"\n"
isRemainRecordDelimiter	否	<p>上传记录时，是否携带分隔符。</p> <ul style="list-style-type: none"> ● true: 携带分隔符。 ● false: 不携带分隔符。 	false
isFileAppendable	否	<p>文件是否有追加内容的可能。</p> <ul style="list-style-type: none"> ● true: 文件可能会追加内容。Agent持续监控文件，若文件追加了内容则根据recordDelimiter解析后上传记录。此时要保证文件以recordDelimiter结尾，否则Agent会认为文件追加未完成，继续等待recordDelimiter写入。 ● false: 文件不会追加内容。文件最后一行不以recordDelimiter结尾，Agent仍会当做最后一条记录上传，上传完成后根据“deletePolicy”和“fileSuffix”的配置执行文件删除或重命名操作。 	true

配置项	是否必填	说明	默认值
maxFileCheckingMilliseconds	否	<p>最长文件变动检查时间，如果文件在此时间内“大小”、“修改时间”和“文件ID”都没有变化，则认为文件已经完成并开始上传。</p> <p>请根据实际文件变动的频率配置此值，避免文件未完成已开始上传的情况。</p> <p>若文件上传后有变动，则会重新全量上传。</p> <p>单位：毫秒</p> <p>说明 “isFileAppendable”配置为“false”时该配置项生效。</p>	5000
deletePolicy	否	<p>文件内容上传完成之后的删除策略。</p> <ul style="list-style-type: none"> • never: 文件内容上传完毕后不删除文件。 • immediate: 文件内容上传完毕后删除文件。 <p>说明 “isFileAppendable”配置为“false”时该配置项生效。</p>	never
fileSuffix	否	<p>文件内容上传完成之后添加的文件名后缀。</p> <p>例如：原文件名为“x.txt”，“fileSuffix”配置为“.COMPLETED”，则文件上传后的命名为“x.txt.COMPLETED”。</p> <p>说明 “isFileAppendable”配置为“false”，同时“deletePolicy”配置为“never”，该配置项生效。</p>	.COMPLETED
sendingThreadSize	否	<p>发送线程数。默认单线程发送。</p> <p>须知 使用多线程会导致如下问题：</p> <ul style="list-style-type: none"> • 数据发送不保证顺序。 • 程序异常停止并重新启动时会丢失部分数据。 	1
fileEncoding	否	<p>文件编码格式，支持UTF8, GBK, GB2312, ISO-8859-1等</p>	UTF8

配置项	是否必填	说明	默认值
resultLogLevel	否	每次调用DIS数据发送接口后的结果日志级别。 • OFF : 日志中不输出每次接口调用的结果。 • INFO : 每次接口调用的结果以INFO级别输出到日志。 • WARN : 每次接口调用的结果以WARN级别输出到日志。 • ERROR : 每次接口调用的结果以ERROR级别输出到日志。	INFO

----结束

Windows 服务器上配置 DIS Agent

- 步骤1** 使用文件管理器进入安装包解压后的目录，例如“C:\dis-agent-X.X.X”。
- 步骤2** 使用编辑器打开“agent.yml”文件，根据实际情况修改各配置项的值并保存。

说明

“agent.yml”文件为linux格式，建议使用通用文本编辑器工具编辑文件。

关于日志文件的补充说明：

在dis-agent程序的安装路径下，logs目录中存放程序运行产生的日志文件，其中dis-agent.log文件记录程序运行状况，dis-agent-2022-10-28.log等带日期的log文件记录文件上传记录，每天生成一个日志文件。

为此，用户也可以在dis-agent程序安装路径下的conf文件夹修改log4j2.xml文件，自定义log文件的存放位置（如下图红框所示位置修改）。

图 4-3 log4j2

```
log4j2.xml - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">
  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout charset="GBK" pattern="%d{yyyy-MM-dd HH:mm:ss.SSSZ} %X{hostname} [%-5p] (%t) %c{.1} %m%n"/>
    </Console>

    <RollingFile name="RollingFile" fileName="logs/${env:DIS_AGENT_NAME:-dis-agent}.log"
      filePattern="logs/${env:DIS_AGENT_NAME:-dis-agent}-%d{yyyy-MM-dd}-%i.log">
      <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss.SSSZ} %X{hostname} [%-5p] (%t) %c{.1} %m%n"/>
      <Policies>
        <TimeBasedTriggeringPolicy/>
        <SizeBasedTriggeringPolicy size="100MB"/>
      </Policies>
      <DefaultRolloverStrategy fileIndex="max" max="10">
        <Delete basePath="logs/" maxDepth="1">
          <IfFileName glob="*.log"/>
          <IfLastModified age="15d"/>
        </Delete>
      </DefaultRolloverStrategy>
    </RollingFile>

  </Appenders>
  <Loggers>
    <Root level="INFO">
      <AppenderRef ref="Console"/>
      <AppenderRef ref="RollingFile"/>
    </Root>
    <logger name="org.apache.http" level="INFO"/>
    <logger name="com.obs.services.internal.RestStorageService" level="WARN"/>
    <logger name="com.obs.services.ObsClient" level="WARN"/>
  </Loggers>
</Configuration>
```

----结束

4.2.5 启动 DIS Agent

Linux 服务器上启动 DIS Agent

- 步骤1** 使用PuTTY工具登录日志所在服务器。
- 步骤2** 进入DIS Agent安装目录。其中“x.x.x”表示版本号。

```
cd /opt/dis-agent-x.x.x/
```

- 步骤3** 启动DIS Agent。

```
bash bin/start-dis-agent.sh
```

如果需要启动多个DIS Agent进程，则新启的Agent进程需要通过-c指定配置文件以及-n参数指定名称。

```
bash bin/start-dis-agent.sh -c config/anotherAgent.yml -n anotherAgent
```

须知

请确保使用bash执行脚本，否则使用sh、./等方式启动脚本，可能由于系统默认shell的差异导致启动失败。

显示类似如下信息，表示Agent启动成功。

```
Success to start DIS Agent [xxxxx].
```

如果启动出现java变量找不到的情况，执行 `source /etc/profile`后重新启动Agent。

----结束

Windows 服务器上启动 DIS Agent

步骤1 进入DIS Agent程序的bin目录，例如“C:\dis-agent-X.X.X\bin”。

步骤2 双击“start-dis-agent.bat”，启动DIS Agent。

若控制台前几行中打印出如下日志表示启动成功。

```
[INFO ] (main) com.bigdata.dis.agent.Agent Agent: Startup completed in XXX ms.
```

----结束

4.2.6 验证 DIS Agent

Linux 服务器上验证 DIS Agent

步骤1 使用PuTTY工具登录日志所在服务器。

步骤2 进入DIS Agent的日志目录。

```
cd /opt/dis-agent-X.X.X/logs
```

步骤3 查看日志。

```
tail -100f dis-agent.log
```

- 显示如下信息，表示Agent正常运行。
Agent: Startup completed in xx ms
- Agent运行异常，常见问题原因和处理方法如下：
 - **HttpClientErrorException: 400 Bad Request**
可能原因：[配置DIS Agent](#)中“DISStream”或“projectId”的配置不正确。
处理方法：停止Agent进程后检查配置。
 - **HttpClientErrorException: 403 Forbidden**
可能原因：DIS网关将服务器IP加入黑名单，导致请求被拦截。列入黑名单通常由于多次使用错误的配置重复调用DIS接口导致。
处理方法：停止Agent进程，修改[配置DIS Agent](#)中“agent.yml”配置文件的配置。停止Agent30分钟后重启Agent。
 - **UnknownHttpStatusCodeException: Unknown status code [441]**
可能原因：AK/SK配置错误。
处理方法：停止Agent进程检查AK/SK配置。
 - **ConnectTimeoutException: Connect to DOMAIN[DOMAIN/IP] failed: connect timed out**
可能原因：服务器连接DIS网关超时。
处理方法：检查Agent所在日志服务器的网络配置是否可以连接公网。

步骤4 查看Agent是否上传日志。

- “agent.yml”中配置的监控目录下有匹配的文件，日志中会输出类似如下日志，表示解析了[N1行(B1字节数)/N2文件(B2字节数)]，成功上传了[N3行/N4文件]。


```
Agent: Progress: [N1 records (B1 bytes) / N2 files (B2 bytes)] parsed, and [N3 records / N4 files] sent successfully to destinations. Uptime: 30146ms
```

- 若监控目录下没有匹配文件，则执行如下命令生成日志文件。

```
echo "`date` Hello world." >> /tmp/test.log
```

步骤5 登录DIS控制台，查看[配置DIS Agent](#)中“DISStream”或“CustomFileStream”通道的监控。有数据上传，表示DIS服务接收正常，Agent安装成功。

----结束

Windows 服务器上验证 DIS Agent

步骤1 使用文件管理器进入“logs”目录。

步骤2 使用编辑器打开“dis-agent.log”文件查看日志。

- 显示如下信息，表示Agent正常运行。

```
Agent: Startup completed in xx ms
```

- Agent运行异常，常见问题原因和处理方法如下：

- **HttpClientErrorException: 400 Bad Request**

可能原因：[配置DIS Agent](#)中“DISStream”或“projectId”的配置不正确。

处理方法：停止Agent进程后检查配置。

- **HttpClientErrorException: 403 Forbidden**

可能原因：DIS网关将服务器IP加入黑名单，导致请求被拦截。列入黑名单通常由于多次使用错误的配置重复调用DIS接口导致。

处理方法：停止Agent进程，修改[配置DIS Agent](#)中“agent.yml”配置文件的配置。停止Agent30分钟后重启Agent。

- **UnknownHttpStatusCodeException: Unknown status code [441]**

可能原因：AK/SK配置错误。

处理方法：停止Agent进程检查AK/SK配置。

- **ConnectTimeoutException: Connect to DOMAIN[DOMAIN/IP] failed: connect timed out**

可能原因：服务器连接DIS网关超时。

处理方法：检查Agent所在日志服务器的网络配置是否可以连接公网。

步骤3 查看Agent是否上传日志。

- “agent.yml”中配置的监控目录下有匹配的文件，日志中会输出类似如下日志，表示解析了[N1行(B1字节数)/N2文件(B2字节数)]，成功上传了[N3行/N4文件]。

```
Agent: Progress: [N1 records (B1 bytes) / N2 files (B2 bytes)] parsed, and [N3 records / N4 files] sent successfully to destinations. Uptime: 30146ms
```

- 若监控目录下没有匹配文件，则执行如下命令生成日志文件。

```
echo %date%time%Hello world. >> C:\test.log
```

步骤4 登录DIS控制台，查看[配置DIS Agent](#)中“DISStream”或“CustomFileStream”通道的监控。有数据上传，表示DIS服务接收正常，Agent安装成功。

----结束

4.2.7 停止 DIS Agent

在 Linux 服务器上停止 DIS Agent

步骤1 使用PuTTY工具登录日志所在服务器。

步骤2 进入DIS Agent安装目录。

```
cd /opt/dis-agent-X.X.X/
```

步骤3 停止DIS Agent。

```
bash bin/stop-dis-agent.sh
```

须知

请确保使用bash执行脚本，否则使用sh、./等方式启动脚本，可能由于系统默认shell的差异导致启动失败。

显示类似如下内容，表示正在停止中。“xxxxx”表示进程ID。

```
Stopping Agent [xxxxx].....
```

显示类似如下内容，表示Agent进程已停止。

```
Stopping Agent [xxxxx]..... Successfully.
```

强制停止Agent进程操作如下。

1. 执行 `ps -ef | grep dis-agent | grep -v grep` 命令，获取Agent的进程标识（PID）。输出的第二个字段即为PID。
2. 执行 `kill -9 PID` 命令，强制停止Agent进程。

----结束

在 Windows 服务器上停止 DIS Agent

步骤1 在控制台窗口按“Ctrl+C”键，提示如下内容。

```
[INFO ] (Agent STOPPING) com.bigdata.dis.agent.Agent Agent: Shutting down...
```

步骤2 等待出现如下提示，则表示停止完成。输入“Y”，单击“回车”键即可正常关闭窗口。

```
Terminate batch job (Y/N)?
```

----结束

4.3 使用 DIS Flume Plugin 上传与下载数据

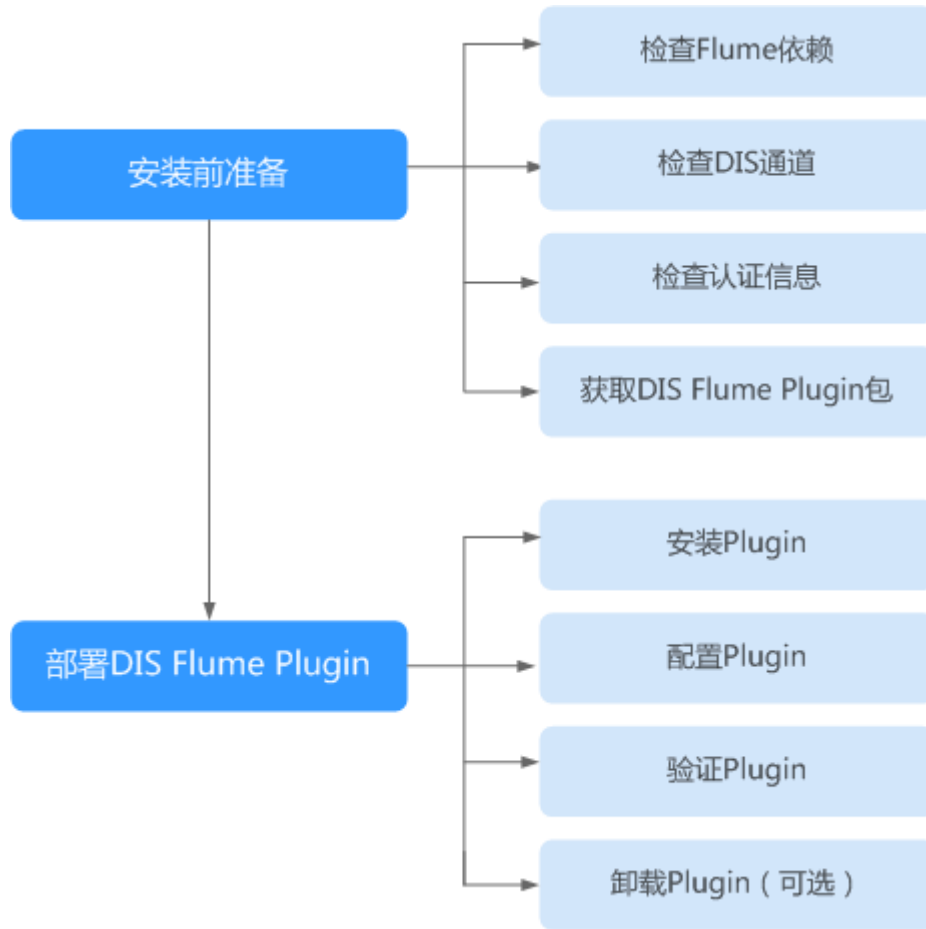
4.3.1 DIS Flume Plugin 概述

DIS Flume Plugin是数据接入服务（DIS）为Flume开发的插件，包含DIS Source与DIS Sink。

- DIS Source用于从DIS服务下载数据到Flume Channel。

- DIS Sink用于将Flume Channel中的数据上传到DIS服务。
DIS Flume Plugin安装流程如图4-4所示。

图 4-4 DIS Flume Plugin 安装流程



4.3.2 安装 DIS Flume Plugin 前准备

检查依赖

步骤1 确认Flume已经安装并能正常运行。

步骤2 确认Flume版本为1.4.0及以上版本。进入Flume安装目录，执行如下命令查看Flume版本。

```
$ bin/flume-ng version | grep Flume
```


步骤3 确认使用的Java版本为1.8.0及以上版本。执行如下命令查看java版本。

```
java -version
```

----结束

检查 DIS 通道

步骤1 使用注册账户登录DIS控制台。

步骤2 单击管理控制台左上角的 ，选择区域和项目。

步骤3 在左侧列表栏中选择“通道管理”。

步骤4 确认有通道用于数据上传且通道状态为“运行中”。

----结束

检查认证信息

- 检查AK/SK
AK/SK (Access Key ID/Secret Access Key)是用户调用接口的访问密钥。由用户在IAM中创建，可在页面下载生成。
- 检查项目ID
ProjectID表示租户的资源，每个Region都有一个唯一的项目ID。可在“我的凭证 > API凭证”页面查看不同Region对应的项目ID值。

获取 DIS Flume Plugin 包

[这里](#)获取“dis-flume-Plugin-X.X.X.zip”压缩包。

4.3.3 安装 Plugin

前提条件

已安装PuTTY工具。

操作步骤

步骤1 使用PuTTY工具(或其他终端工具)远程登录Flume服务器。

步骤2 进入到Flume的安装目录。

```
cd ${FLUME_HOME}
```

步骤3 上传“dis-flume-plugin-X.X.X.zip”安装包到此目录下。

步骤4 解压安装包。

```
unzip dis-flume-plugin-X.X.X.zip
```

步骤5 进入安装包解压后的目录。

```
cd dis-flume-plugin
```

步骤6 运行安装程序。

```
bash install.sh
```

DIS Flume Plugin安装在“\${FLUME_HOME}/plugin.d/dis-flume-plugin”目录下，安装完成后，显示类似如下内容，表示安装成功。

```
Install dis-flume-plugin successfully.
```

----结束

4.3.4 配置 Plugin

DIS Flume Plugin 分为Source与Sink插件，安装包中的dis-flume-plugin.conf.template文件列出了配置方法，本节介绍各种插件的配置项具体含义。

📖 说明

dis-flume-plugin.conf.template只是一个dis插件的配置样例，并不是实际运行Flume时会读取的配置文件。Flume自身提供了样例配置文件，路径为{FLUME_HOME}/conf/flume-conf.properties.template，其中{FLUME_HOME}是Flume的安装路径，用户可以基于此配置文件修改。

在配置项中，需要配置用户的SK，这属于敏感信息，如需加密，可以按如下步骤：

步骤1 进入dis-flume-plugin/目录

```
cd /dis-flume-plugin
```

步骤2 执行加密脚本，输入密码后回车

```
bash dis-encrypt.sh
```

步骤3 控制台打印的“Encrypt result:”后面的字符串即为加密后的结果。通过这种方式分别加密MySQL密码和用户SK，并将密文配置到配置文件中即可。

----结束

配置 DIS Source

表 4-3 DIS Source 配置项说明

配置项	是否必填	说明	默认值
channels	是	Flume channel的名称。	请根据实际情况配置
type	是	Source的类型。	com.cloud.dis.adapter.flume.source.DISSource
streams	是	指定在DIS服务上创建的通道名称。	与DIS控制台“购买接入通道”时配置的“通道名称”取值一致。
ak	是	用户的Access Key。 获取方式请参见 检查认证信息 。	请根据实际情况配置
sk	是	用户的Secret Key。 获取方式请参见 检查认证信息 。	请根据实际情况配置

配置项	是否必填	说明	默认值
region	是	将数据上传到指定Region的DIS服务。	请根据实际情况配置
projectId	是	用户所属区域的项目ID。 获取方式请参见 检查认证信息 。	请根据实际情况配置
endpoint	是	DIS对应Region的数据接口地址。	请根据实际情况配置
group.id	是	DIS App名称，用于标识一个消费组，由英文字符、数字、-、_组成。	请根据实际情况配置

配置 DIS Sink

表 4-4 DIS Sink 配置项说明

配置项	是否必填	说明	默认值
channel	是	Flume channel的名称。	请根据实际情况配置
type	是	Sink的类型。	com.cloud.dis.adapter.flume.sink.DISSink
streamName	是	指定在DIS服务上创建的通道名称。	与DIS控制台“购买接入通道”时配置的“通道名称”取值一致。
ak	是	用户的Access Key。 获取方式请参见 检查认证信息 。	请根据实际情况配置
sk	是	用户的Secret Key。 获取方式请参见 检查认证信息 。	请根据实际情况配置
region	是	将数据上传到指定Region的DIS服务。	-
projectId	是	用户所属区域的项目ID。 获取方式请参见 检查认证信息 。	请根据实际情况配置

配置项	是否必填	说明	默认值
endpoint	是	DIS对应Region的数据接口地址。	-
partitionNumber	否	通道的分区数量。 用于计算batchSize的大小。	1
batchSize	否	一个Flume事务内批量处理的数据集大小。	(“partitionNumber”配置的值)*250
sendingThreadSize	否	发送线程数。默认单线程发送。 说明 使用多线程会出现如下情况： <ul style="list-style-type: none"> • 发送不保证顺序。 • 当程序从异常停止恢复时重传部分数据。 	1
sendingRecordSize	否	单次调用DIS数据发送接口时的数据集大小。 说明 “batchSize”表示一个事务的批量值(如1000)，而“sendingRecordSize”表示一个Rest请求的批量值(如250表示会发起四次请求)。当“batchSize”的数据全部发送成功之后，才会完成Flume的事务，否则事务不提交，重启时导致数据重传。当“sendingThreadSize”配置为“1”时，表示“sendingRecordSize”与“batchSize”配置相同值，避免数据重新上传。	250
retrySize	否	Sink程序调用DIS接口异常时尝试重新调用的次数。 建议使用默认值2147483647，表示会无限重试。 重试时会使用指数退避算法，并等待一段时间，以减轻异常时服务器压力。	2147483647

配置项	是否必填	说明	默认值
resultLogLevel	否	<p>每次调用DIS接口后输出最新sequenceNumber到日志的级别。</p> <p>有效值与级别由低到高为OFF < DEBUG < INFO < WARN < ERROR。</p> <p>OFF：不输出日志。</p> <p>如果Flume log4j配置的日志级别高于resultLogLevel配置的值，则日志也不会输出。</p>	OFF
maxBufferAgeMillis	否	<p>最长上传等待时间。单位：毫秒</p> <ul style="list-style-type: none"> 记录队列满则立即上传。 记录队列未满，等待此配置项配置的时间后上传。 	5000
connectionTimeoutSeconds	否	<p>连接DIS接口超时时间。</p> <p>单位：秒。</p>	30
socketTimeoutSeconds	否	<p>接口响应超时时间。</p> <p>单位：秒。</p>	60
dataEncryptEnabled	否	<p>数据是否使用AES加密。</p> <ul style="list-style-type: none"> 是：true。 否：false。 	false
dataPassword	否	<p>数据加密或解密时使用的密码。</p> <p>当“dataEncryptEnabled”配置项配置为“false”时无需配置“dataPassword”。</p>	请根据实际情况配置
bodySerializeType	否	<p>DIS数据包上传格式(非原始数据格式)。</p> <ul style="list-style-type: none"> json：DIS数据包封装为json格式，满足普通使用。 protobuf：DIS数据包封装为二进制格式，可以减少体积约1/3，在数据量较大的情况下推荐使用此格式。 	json

4.3.5 验证 Plugin

验证 DIS Source

步骤1 使用PuTTY工具远程登录Flume所在服务器。

步骤2 确认已配置好包含dis source的配置文件

可基于Flume自带的flume-conf.properties.template修改，文件样例如下所示：

```
agent.sources = dissource
agent.channels = memoryChannel
agent.sinks = loggerSink
# 定义 Source (使用dis source, 从DIS读取数据)
agent.sources.dissource.channels = memoryChannel
agent.sources.dissource.type = com.cloud.dis.adapter.flume.source.DISSource
agent.sources.dissource.streams = YOU_DIS_STREAM_NAME
agent.sources.dissource.ak = YOU_ACCESS_KEY_ID
agent.sources.dissource.sk = YOU_SECRET_KEY_ID
agent.sources.dissource.region = YOU_Region
agent.sources.dissource.projectId = YOU_PROJECT_ID
agent.sources.dissource.endpoint = https://{region}.cloud.com
agent.sources.dissource.group.id = YOU_APP_NAME
# 定义 Channel
agent.channels.memoryChannel.type = memory
agent.channels.memoryChannel.capacity = 10000
# 定义 Sink (使用logger sink, 输出到控制台)
agent.sinks.loggerSink.type = logger
agent.sinks.loggerSink.channel = memoryChannel
```

步骤3 启动Flume程序，启动命令请参考Apache Flume官网指导。

如果从Flume安装目录启动，示例命令如下所示：

```
bin/flume-ng agent --conf-file conf/flume-conf.properties.template --name agent --conf conf/ -
Dflume.root.logger=INFO,console
```

其中bin/flume-ng agent表示启动Flume Agent；--conf-file 为用户编写的配置文件路径；--name 为配置文件中agent的名称，--conf 为Flume自带的conf/路径

启动之后查看日志，若日志中有类似“source disSource started.”内容，表示DIS Source正常启动，其中“disSource”是用户配置的source名称。

步骤4 检查DIS Source下载数据是否正常。

向source指向的通道上传数据，如果flume没有报错且sink端能正常获取到数据，表示下载正常。

📖 说明

如果使用**步骤2**中示例的配置，则从DIS获取的数据会输出到控制台上，其内容显示为字节数组格式。

步骤5 登录DIS控制台，等待2分钟后，查看**表4-3**中“streams”配置的通道的监控。如果显示有数据下载(蓝色线条)，表示DIS Source运行成功。

----结束

验证 DIS Sink

步骤1 使用PuTTY工具远程登录Flume所在服务器。

步骤2 确认已配置好包含dis sink的配置文件

可基于Flume自带的flume-conf.properties.template修改，文件样例如下所示：

```
agent.sources = exec
agent.channels = memoryChannel
agent.sinks = dissink
# 定义 Source (使用exec source, 监控/tmp/dis.txt文件)
agent.sources.exec.type = exec
agent.sources.exec.command = tail -F /tmp/dis.txt
agent.sources.exec.shell = /bin/bash -c
agent.sources.exec.channels = memoryChannel
# 定义 Channel
agent.channels.memoryChannel.type = memory
agent.channels.memoryChannel.capacity = 10000
# 定义 Sink (使用dis sink, 输出到dis通道)
agent.sinks.dissink.channel = memoryChannel
agent.sinks.dissink.type = com.cloud.dis.adapter.flume.sink.DISSink
agent.sinks.dissink.streamName = YOU_DIS_STREAM_NAME
agent.sinks.dissink.ak = YOU_ACCESS_KEY_ID
agent.sinks.dissink.sk = YOU_SECRET_KEY_ID
agent.sinks.dissink.region = YOU_Region
agent.sinks.dissink.projectId = YOU_PROJECT_ID
agent.sinks.dissink.endpoint = https://dis.${region}.myhuaweicloud.com
agent.sinks.dissink.resultLogLevel = INFO
```

步骤3 启动Flume程序，启动命令请参考Apache Flume官网指导。

如果从Flume安装目录启动，示例命令如下所示

```
bin/flume-ng agent --conf-file conf/flume-conf.properties.template --name agent --conf conf/ -
Dflume.root.logger=INFO,console
```

其中bin/flume-ng agent表示启动Flume Agent；--conf-file 为用户编写的配置文件路径；--name 为配置文件中agent的名称，--conf 为Flume自带的conf/路径。

查看日志，若日志中有类似“Dis flume sink [dissink] start.”内容，表示DIS Sink正常启动，其中“dissink”是用户配置的sink名称。

步骤4 检查DIS Sink上传数据是否正常。

向Flume的source端输入数据，在DIS Sink的resultLogLevel级别不为OFF且不低于log4j配置的值，查看日志输出类似如下结果，表示DIS Sink上传数据正常。

```
CurrentPut 5 events[success 5 / failed 0] spend 131 ms.
```

说明

如果使用**步骤2**中示例的配置，您可创建/tmp/dis.txt文件，并在此文件中追加内容。则启动Flume之后，追加的每行内容会被Flume读取并通过dis sink插件发到DIS通道中。

步骤5 登录DIS控制台，等待2分钟后，查看**表4-4**中“streamName”配置的通道的监控。如果显示有数据上传(绿色线条)，表示DIS Sink运行成功。

----结束

4.3.6 卸载 Plugin(可选)

操作步骤

步骤1 使用PuTTY工具远程登录Flume所在服务器。

步骤2 停止Flume程序。

步骤3 进入DIS Flume Plugin插件所在的目录。

```
cd ${FLUME_HOME}
```

```
cd dis-flume-plugin
```

步骤4 卸载DIS Flume Plugin。

```
dos2unix install.sh
```

```
bash install.sh uninstall
```

出现类似如下提示，表示卸载成功。

```
Uninstall dis-flume-plugin successfully.
```

----结束

4.4 使用 DIS Logstash Plugin 上传与下载数据

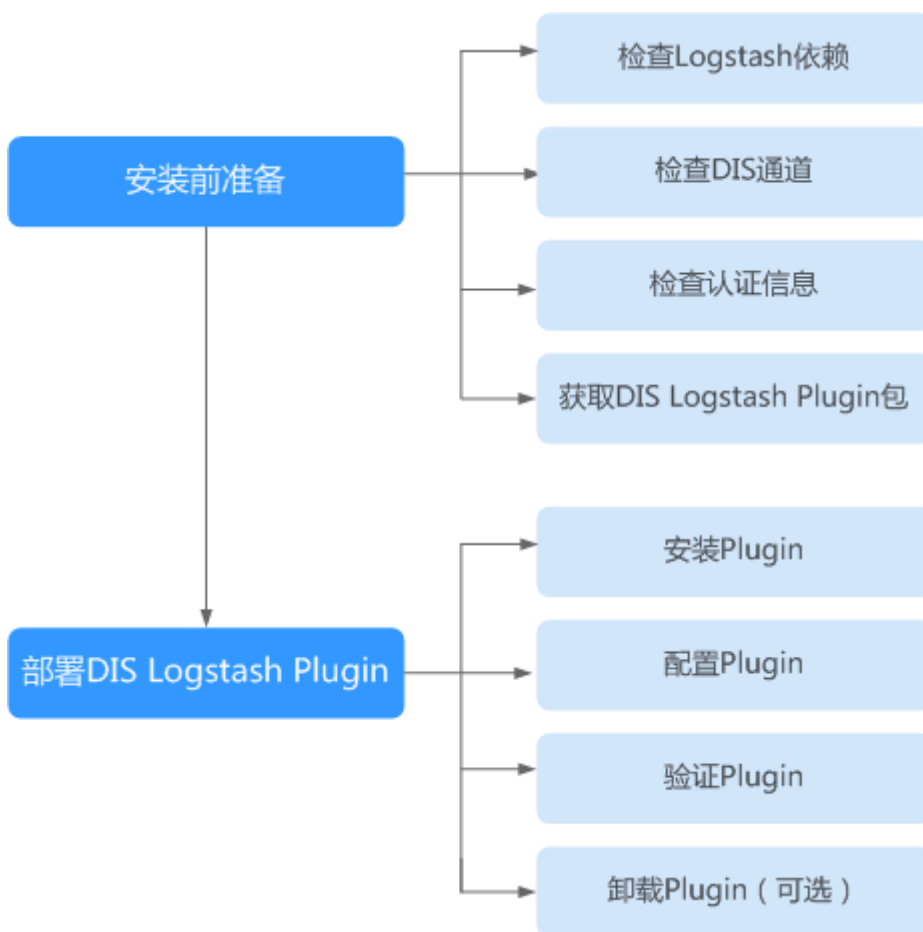
4.4.1 DIS Logstash Plugin 概述

DIS Logstash Plugin是数据接入服务（DIS）为Logstash开发的插件，包含DIS Input与DIS Output。

- DIS Input用于从DIS服务下载数据到Logstash。
- DIS Output用于将Logstash中的数据上传到DIS服务。

DIS Logstash Plugin安装流程如图4-5所示。

图 4-5 安装流程图



4.4.2 安装 DIS Logstash Plugin 前准备

检查依赖

步骤1 确认Logstash已安装并能正常运行。

步骤2 确认使用的Java版本为1.8.0及以上版本。执行如下命令查看Java版本。

```
java -version
```


步骤3 确认使用的JRuby版本为9.0.0.0及以上版本。执行如下命令查看JRuby版本。

```
$ bin/jruby -v
```

----结束

检查 DIS 通道

步骤1 使用注册账户登录**DIS控制台**。

步骤2 单击管理控制台左上角的 ，选择区域和项目。

步骤3 在左侧列表栏中选择“通道管理”。

步骤4 确认有通道用于数据上传且通道状态为“运行中”。

----结束

检查认证信息

- 检查AK/SK
AK/SK (Access Key ID/Secret Access Key)是用户调用接口的访问密钥。由用户在IAM中创建，可在“我的凭证 > 管理访问密钥”页面下载生成。
- 检查项目ID
ProjectID表示租户的资源，每个Region都有一个唯一的项目ID。可在页面查看不同Region对应的项目ID值。

获取 DIS Logstash Plugin 包

<https://dis-publish.obs-website.cn-north-1.myhuaweicloud.com/>获取“dis-logstash-Plugin-X.X.X.zip”压缩包。

4.4.3 在线安装 DIS Logstash Plugin

安装DIS Logstash Plugin有在线和离线安装两种方式：

在线安装无需下载插件包，直接连接公网即可安装。

前提条件

已安装PuTTY工具。

安装 logstash-input-dis

步骤1 使用PuTTY工具(或其他终端工具)远程登录Logstash服务器。

步骤2 进入到Logstash的安装目录。

```
cd ${LOGSTASH_HOME}
```

步骤3 执行安装命令。

```
bin/logstash-plugin install logstash-input-dis
```

安装完成后，显示类似如下内容，表示安装成功。

```
Validating logstash-input-dis  
Installing logstash-input-dis  
Installation successful
```

----结束

安装 logstash-output-dis

步骤1 使用PuTTY工具(或其他终端工具)远程登录Logstash服务器。

步骤2 进入到Logstash的安装目录。

```
cd ${LOGSTASH_HOME}
```

步骤3 执行安装命令。

```
bin/logstash-plugin install logstash-output-dis
```

安装完成后，显示类似如下内容，表示安装成功。

```
Validating logstash-output-dis  
Installing logstash-output-dis  
Installation successful
```

----结束

4.4.4 离线安装 DIS Logstash Plugin

安装DIS Logstash Plugin有在线和离线安装两种方式：

离线安装需要获取插件包并执行安装脚本。

前提条件

已安装PuTTY工具。

操作步骤

步骤1 使用PuTTY工具(或其他终端工具)远程登录Logstash服务器。

步骤2 进入到Logstash的安装目录。

```
cd ${LOGSTASH_HOME}
```

步骤3 上传“dis-logstash-plugins-X.X.X.zip”安装包到此目录下。

步骤4 解压安装包。

```
unzip dis-logstash-plugins-X.X.X.zip
```

步骤5 进入安装包解压后的目录。

cd logstash-plugins

步骤6 运行安装程序，需要指定Logstash的安装目录。

```
bash install.sh -p ${LOGSTASH_HOME}
```

安装完成后，显示类似如下内容，表示安装成功。

```
Install dis-logstash-plugins successfully.
```

----结束

4.4.5 配置 DIS Logstash Plugin

DIS Logstash Plugins 分为Input与Output插件，本节介绍插件的各个配置项具体含义。

配置 DIS Logstash Input

配置模板如下：（该模板为从DIS通道下载数据写入本地文件）

```
input
{
  dis {
    streams => ["YOUR_DIS_STREAM_NAME"]
    endpoint => "https://dis.${region}.myhuaweicloud.com"
    ak => "YOUR_ACCESS_KEY_ID"
    sk => "YOUR_SECRET_KEY_ID"
    region => "YOUR_Region"
    project_id => "YOUR_PROJECT_ID"
    group_id => "YOUR_APP_ID"
    client_id => "YOUR_CLIENT_ID"
    auto_offset_reset => "earliest"
  }
}
output
{
  file {
    path => ["/tmp/test.log"]
  }
}
```

表 4-5 DIS Logstash Input 配置项说明

配置项	是否必填	说明	默认值
stream	是	指定在DIS服务上创建的通道名称。	与DIS控制台“购买接入通道”时配置的“通道名称”取值一致。
ak	是	用户的Access Key。 获取方式请参见 检查认证信息 。	请根据实际情况配置
sk	是	用户的Secret Key。 获取方式请参见 检查认证信息 。	请根据实际情况配置

配置项	是否必填	说明	默认值
region	是	将数据上传到指定Region的DIS服务。	-
project_id	是	用户所属区域的项目ID。 获取方式请参见 检查认证信息 。	请根据实际情况配置
client_id	否	客户端ID，用于标识消费组内的消费者。 起多个pipeline或者多个Logstash实例消费时，需要配置不同的值。比如实例1的值为client1，实例2的值为client2。	logstash
endpoint	是	DIS对应Region的数据接口地址。	-
group_id	是	DIS App名称，用于标识一个消费组，值可以为任意字符串	请根据实际情况配置
auto_offset_reset	否	指定数据从通道中开始消费的位置，支持： earliest：从通道中最早的数据开始消费 latest：从通道中最新的数据开始消费	latest

配置 DIS Logstash Output

配置模板如下：（该模板为读取本地文件数据并上传到DIS通道）

```
input
{
  file {
    path => ["/tmp/test.log"]
    type => "log4j"
    start_position => "beginning"
  }
}
output
{
```

```
dis {
  stream => ["YOUR_DIS_STREAM_NAME"]
  endpoint => "https://dis.${region}.myhuaweicloud.com"
  ak => "YOUR_ACCESS_KEY_ID"
  sk => "YOUR_SECRET_KEY_ID"
  region => "YOUR_Region"
  project_id => "YOUR_PROJECT_ID"
}
```

表 4-6 DIS Logstash Output 配置项说明

配置项	是否必填	说明	默认值
stream	是	指定在DIS服务上创建的通道名称。	与DIS控制台“购买接入通道”时配置的“通道名称”取值一致。
ak	是	用户的Access Key。 获取方式请参见 检查认证信息 。	请根据实际情况配置
sk	是	用户的Secret Key。 获取方式请参见 检查认证信息 。	请根据实际情况配置
region	是	将数据上传到指定Region的DIS服务。	-
project_id	是	用户所属区域的项目ID。 获取方式请参见 检查认证信息 。	请根据实际情况配置
body_compress_enabled	否	是否开启传输数据压缩。	否

配置项	是否必填	说明	默认值
body_compress_type	否	<p>数据压缩类型，当前支持的压缩算法：</p> <p>lz4：综合来看效率最高的压缩算法，更加侧重压缩解压速度，压缩比并不是第一。</p> <p>snappy：其目标不是最大限度压缩或者兼容其他压缩格式，而是旨在提供高速压缩速度和合理的压缩率。</p> <p>zstd：一种新的无损压缩算法，旨在提供快速压缩，并实现高压缩比。</p>	lz4

4.4.6 验证 DIS Logstash Plugin

验证 DIS Logstash Input

步骤1 使用PuTTY工具远程登录Logstash所在服务器。

步骤2 启动Logstash程序。

```
bin/logstash -f dis_to_local.conf
```

其中 -f 为用户编写的配置文件路径。

步骤3 检查DIS Logstash Input下载数据是否正常。

向input指向的通道上传数据，如果Logstash没有报错且output端能正常获取到数据，表示下载正常。

步骤4 登录DIS控制台，等待2分钟后，查看表4-5中“streams”配置的通道的监控。如果显示有数据下载(蓝色线条)，表示DIS Logstash Input运行成功。

----结束

验证 DIS Logstash Output

步骤1 使用PuTTY工具远程登录Logstash所在服务器。

步骤2 启动Logstash程序。

```
bin/logstash -f local_to_dis.conf
```

其中 -f 为用户编写的配置文件路径。

步骤3 检查DIS Logstash Output上传数据是否正常。

向Logstash的input端输入数据，如果Logstash没有报错数据可以正常上传到指向的通道，表示上传正常。

步骤4 登录DIS控制台，等待2分钟后，查看表4-6中“streamName”配置的通道的监控。如果显示有数据上传(绿色线条)，表示DIS Logstash Output运行成功。

----结束

4.4.7 卸载 DIS Logstash Plugin(可选)

操作步骤

步骤1 使用PuTTY工具远程登录Logstash所在服务器。

步骤2 停止Logstash程序。

步骤3 进入DIS Logstash Plugins插件所在的目录。

```
cd ${LOGSTASH_HOME}
```

```
cd logstash-plugins
```

步骤4 卸载DIS Logstash Plugin。

```
bash uninstall.sh -p ${LOGSTASH_HOME}
```

出现类似如下提示，表示卸载成功。

```
Uninstall dis-logstash-plugins successfully.
```

----结束

4.5 使用 Kafka Adapter 上传与下载数据

4.5.1 Kafka Adapter 概述

dis-kafka-adapter是数据接入服务（DIS）提供的一个sdk，支持原本使用Kafka Client上传数据的用户，以类似原来的操作将数据上传到DIS。

目前只支持Java版本。

4.5.2 准备环境

配置 pom.xml 文件

如果已有maven工程，在pom.xml中使用如下依赖即可。

```
<dependency>
  <groupId>com.huaweicloud.dis</groupId>
  <artifactId>huaweicloud-dis-kafka-adapter</artifactId>
  <version>1.2.18</version>
</dependency>
```

使用 DIS 样例工程

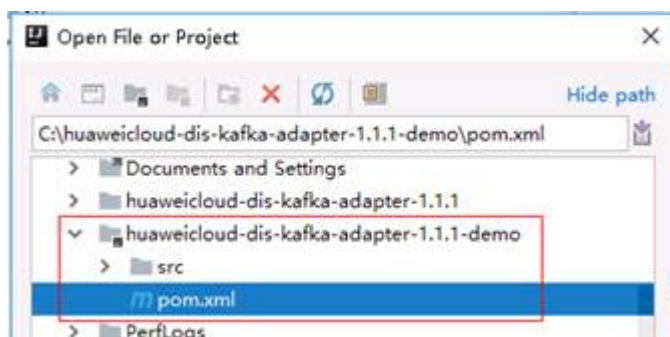
在<https://dis-publish.obs-website.cn-north-1.myhuaweicloud.com/>中下载DIS的kafka-adapter压缩包。

此zip包中有两个目录。

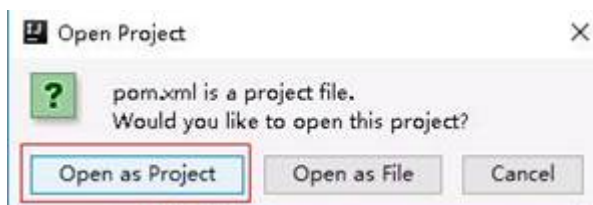
- huaweicloud-dis-kafka-adapter-X.X.X目录下是所有依赖的jar包，如果使用非Maven工程，则可导入此lib目录下的所有jar到环境依赖即可
- huaweicloud-dis-kafka-adapter-X.X.X-demo是一个样例工程，使用maven编写
例如使用IntelliJ IDEA，可按如下方式导入工程。

步骤1 打开IntelliJ IDEA，选择“File > Open”

在弹出的对话框中，选择huaweicloud-dis-kafka-adapter-X.X.X-demo目录，并双击pom.xml确定。



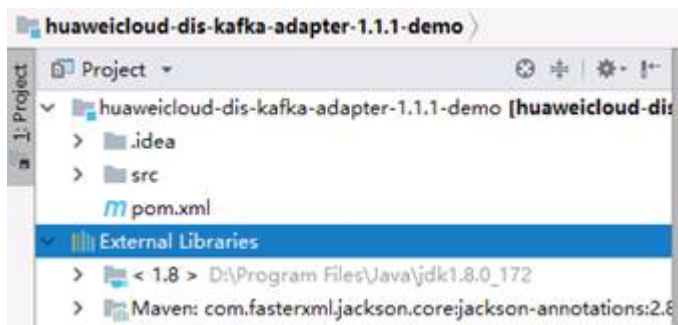
步骤2 当弹出如下对话框，请选择“Open as Project”，作为工程打开。



步骤3 单击“New Window”，在新窗口打开此工程。



步骤4 等待IntelliJ IDEA构建工程，完成之后会显示目录与文件



----结束

检查认证信息

- 检查AK/SK
 - AK/SK (Access Key ID/Secret Access Key)是用户调用接口的访问密钥。您可以通过如下方式获取访问密钥。
 - a. 登录控制台，在用户名下拉列表中选择“我的凭证”。
 - b. 进入“我的凭证”页面，选择“访问密钥 > 新增访问密钥”，如图4-6所示。

图 4-6 单击新增访问密钥



- c. 单击“确定”，根据浏览器提示，保存密钥文件。密钥文件会直接保存到浏览器默认的下载文件夹中。打开名称为“credentials.csv”的文件，即可查看访问密钥（Access Key Id和Secret Access Key）。

说明

- 每个用户仅允许新增两个访问密钥。
 - 为保证访问密钥的安全，访问密钥仅在初次生成时自动下载，后续不可再次通过管理控制台界面获取。请在生成后妥善保管。
- 检查项目ID
 - 项目ID表示租户的资源，账号ID对应当前账号，IAM用户ID对应当前用户。用户可在对应页面下查看不同Region对应的项目ID、账号ID和用户ID。
 - a. 注册并登录管理控制台。
 - b. 在用户名的下拉列表中单击“我的凭证”。
 - c. 在“API凭证”页面，查看账号名和账号ID、IAM用户名和IAM用户ID，在项目列表中查看项目和项目ID。

4.5.3 上传数据

代码样例

“ak”、“sk”和“projectId”信息的获取请参见[检查认证信息](#)。

```
package com.huaweicloud.dis.demo.adapter;
import com.huaweicloud.dis.DISConfig;
import com.huaweicloud.dis.adapter.kafka.clients.producer.*;
import com.huaweicloud.dis.adapter.kafka.common.serialization.StringSerializer;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.util.Properties;
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.Future;
import java.util.concurrent.ThreadLocalRandom;

public class DISKafkaProducerDemo
{
    private static final Logger LOGGER = LoggerFactory.getLogger(DISKafkaProducerDemo.class);

    public static void main(String[] args)
    {
        // 认证用的ak和sk直接写到代码中有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
        // 本示例以ak和sk保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量HUAWEICLOUD_SDK_AK和HUAWEICLOUD_SDK_SK。
        String ak = System.getenv("HUAWEICLOUD_SDK_AK");
        String sk = System.getenv("HUAWEICLOUD_SDK_SK");
        // YOU ProjectId
        String projectId = "YOU_PROJECT_ID";
        // YOU DIS Stream
        String streamName = "YOU_STREAM_NAME";
        // 消费组ID，用于记录offset
        String groupId = "YOU_GROUP_ID";
        // DIS region
        String region = "your region";

        Properties props = new Properties();
        props.setProperty(DISConfig.PROPERTY_AK, ak);
        props.setProperty(DISConfig.PROPERTY_SK, sk);
        props.setProperty(DISConfig.PROPERTY_PROJECT_ID, projectId);
        props.setProperty(DISConfig.PROPERTY_REGION_ID, region);
        props.setProperty(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class.getName());
        props.setProperty(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, StringSerializer.class.getName());

        // 默认情况下不需要设置endpoint，会自动使用域名访问；如需使用指定的endpoint，解除如下注释并设置endpoint即可
        // props.setProperty(DISConfig.PROPERTY_ENDPOINT, "https://dis-$(region).myhuaweicloud.com");

        // Create dis producer
        Producer<String, String> producer = new DISKafkaProducer<>(props);

        // 同步发送
        synchronousSendDemo(producer, streamName);

        // 异步发送
        asynchronousSendDemo(producer, streamName);

        // 关闭producer，防止资源泄露
        producer.close();
    }

    public static void synchronousSendDemo(Producer<String, String> producer, String streamName)
    {
        LOGGER.info("==== synchronous send =====");
        for (int i = 0; i < 5; i++)
        {
            // key设置为随机(或者设置为null)，数据会均匀分配到所有分区中
            String key = String.valueOf(ThreadLocalRandom.current().nextInt(1000000));
            String value = "Hello world[sync]. " + i;
        }
    }
}
```

```
Future<RecordMetadata> future = producer.send(new ProducerRecord<>(streamName, key, value));

try
{
    // 调用future.get会阻塞等待，直到发送完成
    RecordMetadata recordMetadata = future.get();
    // 发送成功
    LOGGER.info("Success to send [{}], Partition [{}], Offset [{}].",
        value, recordMetadata.partition(), recordMetadata.offset());
}
catch (Exception e)
{
    // 发送失败
    LOGGER.error("Failed to send [{}], Error [{}]", value, e.getMessage(), e);
}
}

public static void asynchronousSendDemo(Producer<String, String> producer, String streamName)
{
    LOGGER.info("==== asynchronous send =====");
    int totalSendCount = 5;
    CountdownLatch countDownLatch = new CountdownLatch(totalSendCount);
    for (int i = 0; i < totalSendCount; i++)
    {
        // key设置为随机(或者设置为null)，数据会均匀分配到所有分区中
        String key = String.valueOf(ThreadLocalRandom.current().nextInt(1000000));
        String value = "Hello world[async]. " + i;

        try
        {
            // 使用回调方式发送，不会阻塞
            producer.send(new ProducerRecord<>(streamName, key, value), new Callback()
            {
                @Override
                public void onCompletion(RecordMetadata recordMetadata, Exception e)
                {
                    countDownLatch.countDown();
                    if (e == null)
                    {
                        // 发送成功
                        LOGGER.info("Success to send [{}], Partition [{}], Offset [{}].",
                            value, recordMetadata.partition(), recordMetadata.offset());
                    }
                    else
                    {
                        // 发送失败
                        LOGGER.error("Failed to send [{}], Error [{}]", value, e.getMessage(), e);
                    }
                }
            });
        }
        catch (Exception e)
        {
            countDownLatch.countDown();
            LOGGER.error(e.getMessage(), e);
        }
    }

    try
    {
        // 等待所有发送完成
        countDownLatch.await();
    }
    catch (InterruptedException e)
    {
        LOGGER.error(e.getMessage(), e);
    }
}
```

```
}  
}
```

执行如上程序，发送数据成功会打印如下日志

```
09:32:52.001 INFO c.h.d.d.a.DISKafkaProducerDemo - ===== synchronous send =====  
09:32:53.523 INFO c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[sync]. 0], Partition [0],  
Offset [114].  
09:32:53.706 INFO c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[sync]. 1], Partition [0],  
Offset [115].  
09:32:53.956 INFO c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[sync]. 2], Partition [0],  
Offset [116].  
09:32:54.160 INFO c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[sync]. 3], Partition [0],  
Offset [117].  
09:32:54.450 INFO c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[sync]. 4], Partition [0],  
Offset [118].  
09:32:54.450 INFO c.h.d.d.a.DISKafkaProducerDemo - ===== asynchronous send =====  
09:32:54.673 INFO c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[async]. 0], Partition [0],  
Offset [119].  
09:32:54.674 INFO c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[async]. 1], Partition [0],  
Offset [120].  
09:32:54.674 INFO c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[async]. 2], Partition [0],  
Offset [121].  
09:32:54.674 INFO c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[async]. 3], Partition [0],  
Offset [122].  
09:32:54.674 INFO c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[async]. 4], Partition [0],  
Offset [123].
```

与原生 KafkaProducer 接口适配说明

DISKafkaProducer的实现与KafkaProducer的实现不同，DISKafkaProducer的客户端与服务端通过Rest API实现，而KafkaProducer是基于TCP协议实现，在接口兼容上有如下差异。

表 4-7 适配说明

原生 KafkaProducer	类型	DISKafkaProducer	说明
Future<RecordMetadata> send(ProducerRecord<K, V> record)	接口	支持	发送单条数据
Future<RecordMetadata> send(ProducerRecord<K, V> record, Callback callback)	接口	支持	发送单条数据并设置回调处理函数
void close()	接口	支持	关闭Producer
void close(long timeout, TimeUnit timeUnit)	接口	支持	关闭Producer并设置超时时间
List<PartitionInfo> partitionsFor(String topic)	接口	支持	获取通道的分区信息

原生 KafkaProducer	类型	DISKafkaProducer	说明
void flush(long timeout, TimeUnit unit)	接口	不支持	强制发送当前缓存数据，后续支持
Map<MetricName, ? extends Metric> metrics()	接口	不支持	获取统计信息
key.serializer	参数	支持	含义与kafka设置相同，但默认值为StringSerializer (kafka必须配置)
value.serializer	参数	支持	含义与kafka设置相同，但默认值为StringSerializer (kafka必须配置)
linger.ms	参数	支持	含义与kafka设置相同，但默认值为50(kafka是0)，目的是提高Rest接口的上传效率
batch.size	参数	支持	含义与kafka设置相同，但默认值为1MB(kafka是16KB)，目的是匹配流控的大小
buffer.memory	参数	支持	同kafka的默认设置(32MB)
max.in.flight.requests.per.connection	参数	支持	限制客户端在单个连接上能够发送的未响应请求的个数，默认值为100(Kafka默认为5)可提高发送性能，但可能出现数据顺序不一致的问题。如需严格保证顺序，建议此值设置为1
block.on.buffer.full	参数	支持	同Kafka默认设置(false)。 <ul style="list-style-type: none"> • true表示当发送缓冲区满，send一直阻塞不超时； • false表示发送缓冲区满后根据max.block.ms的时间阻塞，超过时间则抛出异常。
max.block.ms	参数	支持	同Kafka默认设置(60000)。当发送缓冲区满且block.on.buffer.full为false时，控制send()的阻塞时间(毫秒)。
retries	参数	支持，但是参数名改为exception.retries	Kafka默认设置为0，DIS默认设置为8。出现网络/服务端异常的重试次数，尽量保证数据上传成功
其他参数	参数	不支持	-

4.5.4 数据下载的消费模式

同Kafka类似，当前dis kafka adapter支持三种消费模式。

assign 模式

由用户手动指定consumer实例消费哪些具体分区，此时不会拥有group management机制，也就是当group内消费者数量变化或者通道扩缩容的时候不会有重新分配分区的行为发生。代码样例如下所示：

```
package com.huaweicloud.dis.demo.adapter;

import com.huaweicloud.dis.DISConfig;
import com.huaweicloud.dis.adapter.kafka.clients.consumer.*;
import com.huaweicloud.dis.adapter.kafka.common.PartitionInfo;
import com.huaweicloud.dis.adapter.kafka.common.TopicPartition;
import com.huaweicloud.dis.adapter.kafka.common.serialization.StringDeserializer;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.Properties;

public class DISKafkaConsumerAssignDemo
{
    private static final Logger LOGGER = LoggerFactory.getLogger(DISKafkaConsumerAssignDemo.class);

    public static void main(String[] args)
    {
        // 认证用的ak和sk直接写到代码中有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
        // 本示例以ak和sk保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量HUAWEICLOUD_SDK_AK和HUAWEICLOUD_SDK_SK。
        String ak = System.getenv("HUAWEICLOUD_SDK_AK");
        String sk = System.getenv("HUAWEICLOUD_SDK_SK");
        // YOU ProjectId
        String projectId = "YOU_PROJECT_ID";
        // YOU DIS Stream
        String streamName = "YOU_STREAM_NAME";
        // 消费组ID，用于记录offset
        String groupId = "YOU_GROUP_ID";
        // DIS region
        String region = "your region";

        Properties props = new Properties();
        props.setProperty(DISConfig.PROPERTY_AK, ak);
        props.setProperty(DISConfig.PROPERTY_SK, sk);
        props.setProperty(DISConfig.PROPERTY_PROJECT_ID, projectId);
        props.setProperty(DISConfig.PROPERTY_REGION_ID, region);
        props.setProperty(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
        props.setProperty(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
        props.setProperty(ConsumerConfig.GROUP_ID_CONFIG, groupId);
        props.setProperty(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "false");
        props.setProperty(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG,
OffsetResetStrategy.LATEST.name());

        // 默认情况下不需要设置endpoint，会自动使用域名访问；如需使用指定的endpoint，解除如下注释并设置endpoint即可
        // props.setProperty(DISConfig.PROPERTY_ENDPOINT, "https://dis-{$region}.myhuaweicloud.com");

        Consumer<String, String> consumer = new DISKafkaConsumer<>(props);
```

```
List<TopicPartition> topicPartitions = new ArrayList<>();
for (PartitionInfo partitionInfo : consumer.partitionsFor(streamName))
{
    topicPartitions.add(new TopicPartition(partitionInfo.topic(), partitionInfo.partition()));
}

// 使用assign模式, 指定需要消费的分区
consumer.assign(topicPartitions);

while (true)
{
    try
    {
        ConsumerRecords<String, String> records = consumer.poll(Long.MAX_VALUE);

        if (!records.isEmpty())
        {
            for (TopicPartition partition : records.partitions())
            {
                List<ConsumerRecord<String, String>> partitionRecords = records.records(partition);
                for (ConsumerRecord<String, String> record : partitionRecords)
                {
                    LOGGER.info("Value [{}], Partition [{}], Offset [{}], Key [{}]",
                        record.value(), record.partition(), record.offset(), record.key());
                }
            }
            // 数据处理完成之后异步提交当前offset(也可使用同步提交commitSync)
            consumer.commitAsync(new OffsetCommitCallback()
            {
                @Override
                public void onComplete(Map<TopicPartition, OffsetAndMetadata> map, Exception e)
                {
                    if (e == null)
                    {
                        LOGGER.debug("Success to commit offset [{}]", map);
                    }
                    else
                    {
                        LOGGER.error("Failed to commit offset [{}]", e.getMessage(), e);
                    }
                }
            });
        }
    }
    catch (Exception e)
    {
        LOGGER.info(e.getMessage(), e);
    }
}
```

执行如上程序之后, 如果有数据发送到通道中, 此时会打印如下日志。

```
09:36:45.071 INFO c.h.d.a.k.c.DISKafkaConsumer - create DISKafkaConsumer successfully
09:36:49.842 INFO c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[sync]. 0], Partition [0],
Offset [134], Key [769066]
09:36:49.963 INFO c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[sync]. 1], Partition [0],
Offset [135], Key [700005]
09:36:50.145 INFO c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[sync]. 2], Partition [0],
Offset [136], Key [338044]
09:36:51.093 INFO c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[sync]. 3], Partition [0],
Offset [137], Key [537495]
09:36:51.093 INFO c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[sync]. 4], Partition [0],
Offset [138], Key [980016]
09:36:51.093 INFO c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[async]. 0], Partition [0],
Offset [139], Key [182772]
09:36:51.093 INFO c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[async]. 1], Partition [0],
Offset [140], Key [830271]
09:36:51.093 INFO c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[async]. 2], Partition [0],
```

```
Offset [141], Key [927054]
09:36:51.093 INFO c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[async]. 3], Partition [0],
Offset [142], Key [268122]
09:36:51.093 INFO c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[async]. 4], Partition [0],
Offset [143], Key [992787]
```

subscribe 模式

用户指定通道名称即可，无需指定具体分区，服务端会根据消费者的数量变化或者通道扩缩容变化，触发group management机制，自动给每一个消费者分配分区，保证一个分区只会被一个消费者消费。

代码样例如下所示：

```
package com.huaweicloud.dis.demo.adapter;

import com.huaweicloud.dis.DISConfig;
import com.huaweicloud.dis.adapter.kafka.clients.consumer.*;
import com.huaweicloud.dis.adapter.kafka.common.TopicPartition;
import com.huaweicloud.dis.adapter.kafka.common.serialization.StringDeserializer;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.Collections;
import java.util.List;
import java.util.Map;
import java.util.Properties;

public class DISKafkaConsumerSubscribeDemo
{
    private static final Logger LOGGER = LoggerFactory.getLogger(DISKafkaConsumerSubscribeDemo.class);

    public static void main(String[] args)
    {
        // 认证用的ak和sk直接写到代码中有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
        // 本示例以ak和sk保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量HUAWEICLOUD_SDK_AK和HUAWEICLOUD_SDK_SK。
        String ak = System.getenv("HUAWEICLOUD_SDK_AK");
        String sk = System.getenv("HUAWEICLOUD_SDK_SK");
        // YOU ProjectId
        String projectId = "YOU_PROJECT_ID";
        // YOU DIS Stream
        String streamName = "YOU_STREAM_NAME";
        // 消费组ID，用于记录offset
        String groupId = "YOU_GROUP_ID";
        // DIS region
        String region = "your region";

        Properties props = new Properties();
        props.setProperty(DISConfig.PROPERTY_AK, ak);
        props.setProperty(DISConfig.PROPERTY_SK, sk);
        props.setProperty(DISConfig.PROPERTY_PROJECT_ID, projectId);
        props.setProperty(DISConfig.PROPERTY_REGION_ID, region);
        props.setProperty(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
        props.setProperty(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
        props.setProperty(ConsumerConfig.GROUP_ID_CONFIG, groupId);
        props.setProperty(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "false");
        props.setProperty(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG,
OffsetResetStrategy.LATEST.name());

        // 默认情况下不需要设置endpoint，会自动使用域名访问；如需使用指定的endpoint，解除如下注释并设置endpoint即可
```

```
// props.setProperty(DISConfig.PROPERTY_ENDPOINT, "https://dis-$(region).myhuaweicloud.com");

Consumer<String, String> consumer = new DISKafkaConsumer<>(props);
// 使用subscribe模式，指定需要消费的通道名即可
consumer.subscribe(Collections.singleton(streamName));

while (true)
{
    try
    {
        ConsumerRecords<String, String> records = consumer.poll(Long.MAX_VALUE);

        if (!records.isEmpty())
        {
            for (TopicPartition partition : records.partitions())
            {
                List<ConsumerRecord<String, String>> partitionRecords = records.records(partition);
                for (ConsumerRecord<String, String> record : partitionRecords)
                {
                    LOGGER.info("Value [{}], Partition [{}], Offset [{}], Key [{}]",
                        record.value(), record.partition(), record.offset(), record.key());
                }
            }
            // 数据处理完成之后异步提交当前offset(也可使用同步提交commitSync)
            consumer.commitAsync(new OffsetCommitCallback()
            {
                @Override
                public void onComplete(Map<TopicPartition, OffsetAndMetadata> map, Exception e)
                {
                    if (e == null)
                    {
                        LOGGER.debug("Success to commit offset [{}]", map);
                    }
                    else
                    {
                        LOGGER.error("Failed to commit offset [{}]", e.getMessage(), e);
                    }
                }
            });
        }
    }
    catch (Exception e)
    {
        LOGGER.info(e.getMessage(), e);
    }
}
```

程序启动后，会每10s发起心跳请求(Heartbeat)，然后发起加入消费组的请求(JoinGroup)，服务端此时会开始给此消费组中的消费者分配分区，此过程大约需要等待20s，完成之后消费者会发起同步请求(SyncGroup)获取分配结果，等日志中输出Heartbeat {"state":"STABLE"}的信息，表示整个消费组都完成分配，可以正常消费数据了。

此过程的关键日志说明如下

- Heartbeat {"state":"JOINING"}

Heartbeat表示心跳请求，每10s发起一次，用于和服务端保持连接。如果超过1分钟服务端没有收到心跳，会认为消费端已离线，消费组会重新分配。若心跳结果为JOINING表示消费者需要重新加入消费组，若为STABLE表示消费组稳定。

- JoinGroup

如果Heartbeat的结果不为STABLE，则消费者会发起joinGroup的请求，通知服务端自己要加入消费组，服务端收到客户端的join请求之后，会将消费组重新分配，此时返回

一个syncDelayedTimeMs，告诉客户端分配需要多久完成，客户端可以等待syncDelayedTimeMs之后，再发起同步请求(SyncGroup)获取分配结果

- SyncGroup

此请求用于获取分配结果，返回的assignment中即为消费者需要消费的通道名和分区

执行样例程序，等待消费组分配完成之后，发送数据到通道，完整的日志如下

```
09:42:37.296 INFO c.h.d.a.k.c.DISKafkaConsumer - create DISKafkaConsumer successfully
09:42:37.354 INFO c.h.d.a.k.c.Coordinator - Heartbeat {"state":"JOINING"}
09:42:37.363 INFO c.h.d.a.k.c.Coordinator - joinGroupRequest {"groupId":"ding","clientId":"consumer-c2d43144-0823-4eea-aaa8-7af95c536144","interestedStream":["liuhao12"]}
09:42:37.406 INFO c.h.d.a.k.c.Coordinator - joinGroupResponse {"state":"OK","syncDelayedTimeMs":21000}
09:42:58.408 INFO c.h.d.a.k.c.Coordinator - syncGroup {"groupId":"ding","clientId":"consumer-c2d43144-0823-4eea-aaa8-7af95c536144","generation":-1}
09:42:58.451 INFO c.h.d.a.k.c.Coordinator - syncGroup {"state":"OK","generation":33,"assignment":{"distinct":[0]}}
09:42:58.488 INFO c.h.d.a.k.c.Coordinator - Heartbeat {"state":"STABLE"}
09:43:08.960 INFO c.h.d.a.k.c.Coordinator - Heartbeat {"state":"STABLE"}
09:46:56.227 INFO c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[sync]. 0], Partition [0], Offset [144], Key [799704]
09:46:56.327 INFO c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[sync]. 1], Partition [0], Offset [145], Key [683757]
09:46:56.449 INFO c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[sync]. 2], Partition [0], Offset [146], Key [439062]
09:46:56.535 INFO c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[sync]. 3], Partition [0], Offset [147], Key [374939]
09:46:56.654 INFO c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[sync]. 4], Partition [0], Offset [148], Key [321528]
09:46:56.749 INFO c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[async]. 0], Partition [0], Offset [149], Key [964841]
09:46:56.749 INFO c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[async]. 1], Partition [0], Offset [150], Key [520262]
09:46:56.749 INFO c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[async]. 2], Partition [0], Offset [151], Key [619119]
09:46:56.749 INFO c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[async]. 3], Partition [0], Offset [152], Key [257094]
09:46:56.749 INFO c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[async]. 4], Partition [0], Offset [153], Key [310331]
```

subscribePattern 模式

subscribePattern是在subscribe的基础上，用户不用指定具体的通道名称而是使用通配符，例如stream.*表示会消费stream1, stream2, stream_123等等。已有/新增/删除的通道，只要匹配通配符，就可被消费组消费。

代码样例如下所示：

```
package com.huaweicloud.dis.demo.adapter;

import com.huaweicloud.dis.DISConfig;
import com.huaweicloud.dis.adapter.kafka.clients.consumer.*;
import com.huaweicloud.dis.adapter.kafka.common.TopicPartition;
import com.huaweicloud.dis.adapter.kafka.common.serialization.StringDeserializer;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.Collection;
import java.util.List;
import java.util.Map;
import java.util.Properties;
import java.util.regex.Pattern;

public class DISKafkaConsumerSubscribePatternDemo
{
```

```
private static final Logger LOGGER =
LoggerFactory.getLogger(DISKafkaConsumerSubscribePatternDemo.class);

public static void main(String[] args)
{
    // 认证用的ak和sk直接写到代码中有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全；
    // 本示例以ak和sk保存在环境变量中来实现身份验证为例，运行本示例前请先在本地环境中设置环境变量HUAWEICLOUD_SDK_AK和HUAWEICLOUD_SDK_SK。
    String ak = System.getenv("HUAWEICLOUD_SDK_AK");
    String sk = System.getenv("HUAWEICLOUD_SDK_SK");
    // YOU ProjectId
    String projectId = "YOU_PROJECT_ID";
    // YOU DIS Stream
    String streamName = "YOU_STREAM_NAME";
    // 消费组ID，用于记录offset
    String groupId = "YOU_GROUP_ID";
    // DIS region
    String region = "your region";

    Properties props = new Properties();
    props.setProperty(DISConfig.PROPERTY_AK, ak);
    props.setProperty(DISConfig.PROPERTY_SK, sk);
    props.setProperty(DISConfig.PROPERTY_PROJECT_ID, projectId);
    props.setProperty(DISConfig.PROPERTY_REGION_ID, region);
    props.setProperty(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
    props.setProperty(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
    props.setProperty(ConsumerConfig.GROUP_ID_CONFIG, groupId);
    props.setProperty(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "false");
    props.setProperty(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG,
OffsetResetStrategy.LATEST.name());

    // 默认情况下不需要设置endpoint，会自动使用域名访问；如需使用指定的endpoint，解除如下注释并设置endpoint即可
    // props.setProperty(DISConfig.PROPERTY_ENDPOINT, "https://dis-{$region}.myhuaweicloud.com");

    Consumer<String, String> consumer = new DISKafkaConsumer<>(props);
    // 使用subscribePattern模式，指定通配符即可
    consumer.subscribe(Pattern.compile(streamNamePattern), new ConsumerRebalanceListener()
    {
        @Override
        public void onPartitionsRevoked(Collection<TopicPartition> collection)
        {
            LOGGER.info("onPartitionsRevoked [{}]", collection);
        }

        @Override
        public void onPartitionsAssigned(Collection<TopicPartition> collection)
        {
            LOGGER.info("onPartitionsAssigned [{}]", collection);
        }
    });

    while (true)
    {
        try
        {
            ConsumerRecords<String, String> records = consumer.poll(Long.MAX_VALUE);

            if (!records.isEmpty())
            {
                for (TopicPartition partition : records.partitions())
                {
                    List<ConsumerRecord<String, String>> partitionRecords = records.records(partition);
                    for (ConsumerRecord<String, String> record : partitionRecords)
                    {
                        LOGGER.info("Value [{}], Partition [{}], Offset [{}], Key [{}]",
```

```
        record.value(), record.partition(), record.offset(), record.key());
    }
}
// 数据处理完成之后异步提交当前offset(也可使用同步提交commitSync)
consumer.commitAsync(new OffsetCommitCallback()
{
    @Override
    public void onComplete(Map<TopicPartition, OffsetAndMetadata> map, Exception e)
    {
        if (e == null)
        {
            LOGGER.debug("Success to commit offset [{}]", map);
        }
        else
        {
            LOGGER.error("Failed to commit offset [{}]", e.getMessage(), e);
        }
    }
});
}
}
catch (Exception e)
{
    LOGGER.info(e.getMessage(), e);
}
}
}
```

程序启动之后，仍然需要等待约20s，服务端分配完成之后，才可以开始消费数据，样例代码执行的日志如下所示

```
10:10:36.420 INFO c.h.d.a.k.c.DISKafkaConsumer - create DISKafkaConsumer successfully
10:10:36.481 INFO c.h.d.a.k.c.Coordinator - Heartbeat {"state":"JOINING"}
10:10:36.486 INFO c.h.d.a.k.c.Coordinator - joinGroupRequest {"groupId":"ding","clientId":"consumer-cad967ba-70ab-4e02-b184-f60b95fe3256","streamPattern":"stream.*"}
10:10:36.697 INFO c.h.d.a.k.c.Coordinator - joinGroupResponse {"state":"OK","subscription":["stream_hello","stream_world"],"syncDelayedTimeMs":21000}
10:10:57.699 INFO c.h.d.a.k.c.Coordinator - syncGroup {"groupId":"ding","clientId":"consumer-cad967ba-70ab-4e02-b184-f60b95fe3256","generation":-1}
10:10:57.746 INFO c.h.d.a.k.c.Coordinator - syncGroup {"state":"OK","generation":34,"assignment":{"stream_hello":[0],"stream_world":[0]}}
10:10:57.770 INFO c.h.d.a.DISKafkaConsumerSubscribePatternDemo - onPartitionsAssigned [[stream_hello-0, stream_world-0]]
10:10:57.770 INFO c.h.d.a.k.c.Coordinator - Heartbeat {"state":"STABLE"}
10:11:08.466 INFO c.h.d.a.k.c.Coordinator - Heartbeat {"state":"STABLE"}
10:11:09.992 INFO c.h.d.d.a.DISKafkaConsumerSubscribePatternDemo - Value [Hello world[sync]. 0], Partition [0], Offset [154], Key [181881]
10:11:09.993 INFO c.h.d.d.a.DISKafkaConsumerSubscribePatternDemo - Value [Hello world[sync]. 1], Partition [0], Offset [155], Key [483023]
10:11:09.993 INFO c.h.d.d.a.DISKafkaConsumerSubscribePatternDemo - Value [Hello world[sync]. 2], Partition [0], Offset [156], Key [32453]
10:11:10.093 INFO c.h.d.d.a.DISKafkaConsumerSubscribePatternDemo - Value [Hello world[sync]. 3], Partition [0], Offset [157], Key [111948]
10:11:10.180 INFO c.h.d.d.a.DISKafkaConsumerSubscribePatternDemo - Value [Hello world[sync]. 4], Partition [0], Offset [158], Key [822860]
```

4.5.5 下载数据之消费位移

消费位移确认有自动提交与手动提交两种策略，在创建DISKafkaConsumer对象时，通过参数enable.auto.commit设定，true表示自动提交（默认）。

自动提交策略由消费者协调器（Coordinator）每隔\${auto.commit.interval.ms}毫秒执行一次偏移量的提交；手动提交需要由客户端自己控制偏移量的提交。

- 自动提交

在创建一个消费者时，默认是自动提交偏移量，默认的提交间隔是5000ms。使用自动提交相关参数设置如下：

```
props.setProperty("enable.auto.commit", "true");// 显示设置偏移量自动提交
props.setProperty("auto.commit.interval.ms", "5000");// 设置偏移量提交时间间隔
```

- 手动提交

在有些场景可能对消费偏移量有更精确的管理，以保证消息不被重复消费以及消息不被丢失。假设对拉取到的消息需要进行写入数据库处理，或者用于其他网络访问请求等等复杂的业务处理，在这种场景下，所有的业务处理完成后才认为消息被成功消费，此时必须手动控制偏移量的提交。使用手动提交相关参数设置如下：

```
props.put("enable.auto.commit", "false");// 显示设置偏移量手动提交
```

然后在业务处理成功后调用commitAsync()或commitSync()方法提交偏移量：commitAsync()是异步提交，消费者线程不会被阻塞，可能在提交偏移量操作的结果还未返回时就开始进行下一次的拉取操作，如需获取提交的结果，可以添加回调方法OffsetCommitCallback，当提交偏移量完成后会自动调用其onComplete()方法，以便根据回调结果执行不同的逻辑处理；

commitSync()是同步提交，会阻塞线程直到提交消费偏移量执行结果返回。

另外还可以精细的控制对具体分区具体offset数据的确认，确认的offset为已接受数据最大offset+1。例如消费一批数据，最后一条的offset为100，则此时需要commit 101，这样下次消费就会从101开始，不会重复。代码样例如下：

```
ConsumerRecords<String, String> records = consumer.poll(Long.MAX_VALUE);

if (!records.isEmpty())
{
    for (TopicPartition partition : records.partitions())
    {
        List<ConsumerRecord<String, String>> partitionRecords = records.records(partition);
        for (ConsumerRecord<String, String> record : partitionRecords)
        {
            LOGGER.info("Value [{}], Partition [{}], Offset [{}], Key [{}]",
                record.value(), record.partition(), record.offset(), record.key());
        }
        if (!partitionRecords.isEmpty())
        {
            // 同步确认某个分区的特定offset
            long lastOffset = partitionRecords.get(partitionRecords.size() - 1).offset();
            consumer.commitSync(Collections.singletonMap(partition, new OffsetAndMetadata(lastOffset + 1)));
        }
    }
}
```

4.5.6 与原生 KafkaConsumer 接口适配说明

表 4-8 接口适配说明

原生 KafkaConsumer	类型	DISKafkaConsumer	说明
Set<TopicPartition> assignment()	接口	支持	获取consumer消费的通道与分区信息
Set<String> subscription()	接口	支持	获取consumer已订阅的通道名称

原生 KafkaConsumer	类型	DISKafkaConsumer	说明
void assign(Collection<TopicPartition> var1)	接口	支持	分配指定的分区
void subscribe(Collection<String> var1)	接口	支持	订阅指定的通道
void subscribe(Collection<String> var1, ConsumerRebalanceListener var2)	接口	支持	订阅指定的通道并支持 ConsumerRebalanceListener 回调
void subscribe(Pattern var1, ConsumerRebalanceListener var2)	接口	支持	订阅所有匹配通配符的通道并支持 ConsumerRebalanceListener 回调
void unsubscribe()	接口	支持	取消所有订阅
ConsumerRecords<K, V> poll(long var1)	接口	支持	获取消息，但消息当中未实现 checksum(消息的CRC32校验值)、serializedKeySize(key序列化后的字节长度)、serializedValueSize(key序列化后的字节长度)。
void commitSync()	接口	支持	同步提交当前消费的offset
void commitSync(final Map<TopicPartition, OffsetAndMetadata> offsets)	接口	支持	同步提交指定的offset
void commitAsync()	接口	支持	异步提交当前消费的offset
public void commitAsync(OffsetCommitCallback callback)	接口	支持	异步提交当前消费的offset并支持 OffsetCommitCallback 回调

原生 KafkaConsumer	类型	DISKafkaConsumer	说明
void commitAsync(Map<TopicPartition, OffsetAndMetadata> offsets, OffsetCommitCallback callback)	接口	支持	异步提交指定的offset并支持OffsetCommitCallback回调
void seek(TopicPartition partition, long offset)	接口	支持	给分区设置指定的offset
void seekToBeginning(Collection<TopicPartition> partitions)	接口	支持	分区的offset设置为最旧的值
void seekToEnd(Collection<TopicPartition> partitions)	接口	支持	分区的offset设置为最新的值
long position(TopicPartition partition)	接口	支持	获取分区当前已消费数据的offset
OffsetAndMetadata committed(TopicPartition partition)	接口	支持	获取分区已提交的offset
List<PartitionInfo> partitionsFor(String topic)	接口	支持	获取通道的分区信息，但PartitionInfo里面的leader, replicas, inSyncReplicas未实现。
Map<String, List<PartitionInfo>> listTopics()	接口	支持	获取所有的通道信息，但PartitionInfo里面的leader, replicas, inSyncReplicas未实现。
void pause(Collection<TopicPartition> partitions)	接口	支持	暂停消费分区
void resume(Collection<TopicPartition> partitions)	接口	支持	恢复消费分区

原生 KafkaConsumer	类型	DISKafkaConsumer	说明
Set<TopicPartition> paused()	接口	支持	获取所有已暂停消费的分区
close()	接口	支持	关闭consumer
Map<MetricName, ? extends Metric> metrics()	接口	不支持	获取统计信息
wakeup()	接口	不支持	内部实现原理不一样，不需要。
group.id	参数	支持	消费组ID
client.id	参数	支持	每个consumer的client.id必须唯一，如果不配置client.id, diskafka consumer会生成一个uuid作为client.id。
key.deserializer	参数	支持	含义与kafka设置相同，但默认值为StringDeserializer (kafka必须配置)。
value.deserializer	参数	支持	含义与kafka设置相同，但默认值为StringDeserializer (kafka必须配置)。
enable.auto.commit	参数	支持	同kafka的默认设置，默认为true <ul style="list-style-type: none"> • true表示启用自动提交，每隔\${auto.commit.interval.ms}的时间提交一次offset; • false表示不自动提交offset
auto.commit.interval.ms	参数	支持	自动提交offset的周期(毫秒)，默认值5000。
auto.offset.reset	参数	支持	同Kafka的默认配置，默认为latest。 此值用于没有初始偏移量或者偏移量不正确的情况下，自动设置offset位置： <ul style="list-style-type: none"> • earliest 将偏移量自动重置为最旧的值; • latest将偏移量自动重置为最新的值; • none 如果在消费者组中没有发现前一个偏移量，就向消费者抛出一个异常;
其他参数	参数	不支持	-

4.5.7 使用 AuthToken 认证

代码样例

本代码只展示AuthToken配置方法，具体消费代码请参考5.5.3、5.5.4章节

```
package com.huaweicloud.dis.demo.adapter;
import com.huaweicloud.dis.DISConfig;
import com.huaweicloud.dis.adapter.kafka.clients.producer.*;
import com.huaweicloud.dis.adapter.kafka.common.serialization.StringSerializer;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.util.Properties;
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.Future;
import java.util.concurrent.ThreadLocalRandom;

public class DISKafkaProducerDemo
{
    private static final Logger LOGGER = LoggerFactory.getLogger(DISKafkaProducerDemo.class);

    public static void main(String[] args)
    {
        // YOU AuthType
        String authType="authToken";
        // YOU AuthToken
        String authToken="XXXXXXXX";
        // YOU ProjectId
        String projectId = "YOU_PROJECT_ID";
        // YOU DIS Stream
        String streamName = "YOU_STREAM_NAME";
        // DIS region
        String region = "YOU_Region";

        Properties props = new Properties();
        props.setProperty(DISConfig.PROPERTY_AUTH_TYPE,authType);
        props.setProperty(DISConfig.PROPERTY_AUTH_TOKEN,authToken);
        props.setProperty(DISConfig.PROPERTY_PROJECT_ID, projectId);
        props.setProperty(DISConfig.PROPERTY_REGION_ID, region);
        props.setProperty(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class.getName());
        props.setProperty(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
StringSerializer.class.getName());

        //doing next...
    }
}
```

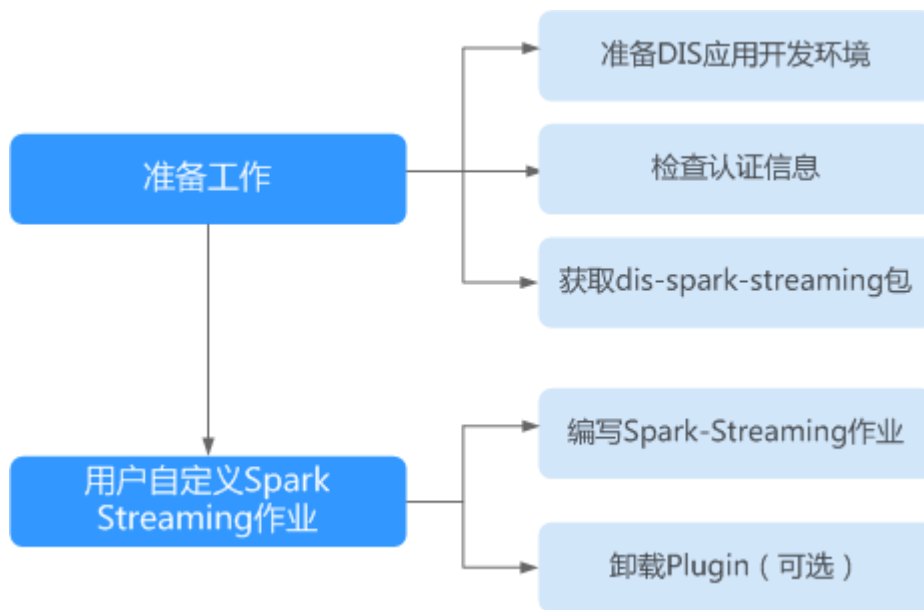
4.6 使用 DIS Spark Streaming 下载数据

4.6.1 DIS Spark Streaming 概述

DIS Spark Streaming是数据接入服务（DIS）提供的一个sdk，支持将DIS作为数据源创建DStream对接SparkStreaming。

DIS Spark Streaming使用流程如[图4-7](#)所示。

图 4-7 DIS Spark Streaming 使用流程



4.6.2 准备 DIS Spark Streaming 的相关环境

准备 DIS 应用开发环境

步骤1 参考[步骤1：开通DIS通道](#)准备相应DIS环境。

步骤2 安装Maven并配置本地仓库地址。

步骤3 安装scala-sdk。

----结束

配置 DIS Spark Streaming 依赖

项目中可通过以下配置引入DIS Spark Streaming依赖：

```
<dependency>
  <groupId>com.cloud.dis</groupId>
  <artifactId>cloud-dis-spark-streaming_2.11</artifactId>
  <version>1.2.1</version>
  <scope>compile</scope>
</dependency>
```

检查认证信息

- 检查AK/SK
AK/SK (Access Key ID/Secret Access Key)是用户调用接口的访问密钥。由用户在IAM中创建，可在“我的凭证 > 管理访问密钥”页面下载生成。
- 检查项目ID
ProjectID表示租户的资源，每个Region都有一个唯一的项目ID。可在页面查看不同Region对应的项目ID值。

4.6.3 自定义 SparkStreaming 作业

获取 DIS Spark Streaming Demo

步骤1 [这里](#)获取 “dis-spark-streaming-X.X.X.zip” 压缩包。解压 “dis-spark-streaming-X.X.X.zip” 压缩包，解压之后获得以下目录：

“dis-spark-streaming-demo” 目录包含一个Maven工程样例。

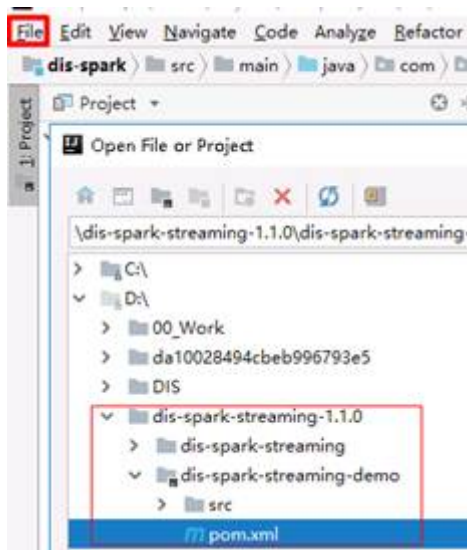
----结束

编写 SparkStreaming 作业

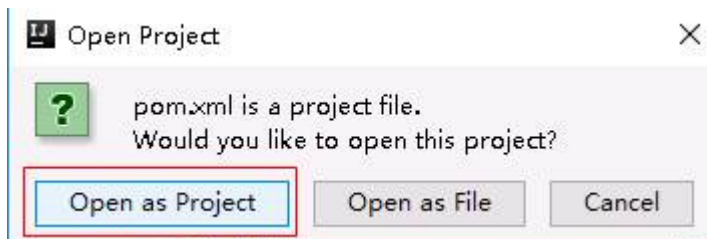
以IntelliJ IDEA社区版为例，说明如何编写SparkStreaming作业。请先确保在IDEA上已经正确配置好

- JDK 1.8+
- Scala-sdk-2.11
- Maven 3.3.*

步骤1 打开IntelliJIDEA，选择“File > Open”。选择解压至本地的dis-spark-streaming-demo目录，双击pom.xml。



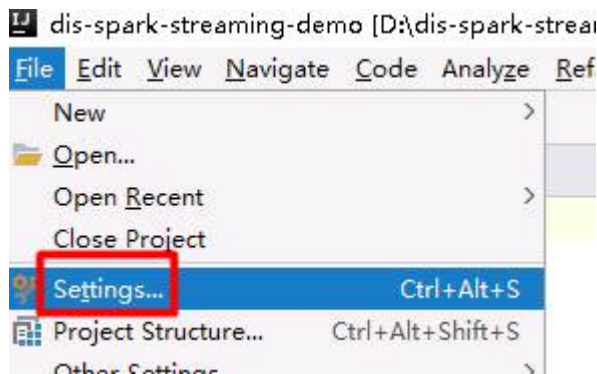
步骤2 当弹出如下对话框，请选择“Open as Project”，作为工程打开。



步骤3 单击“New Window”，在新窗口打开此工程。



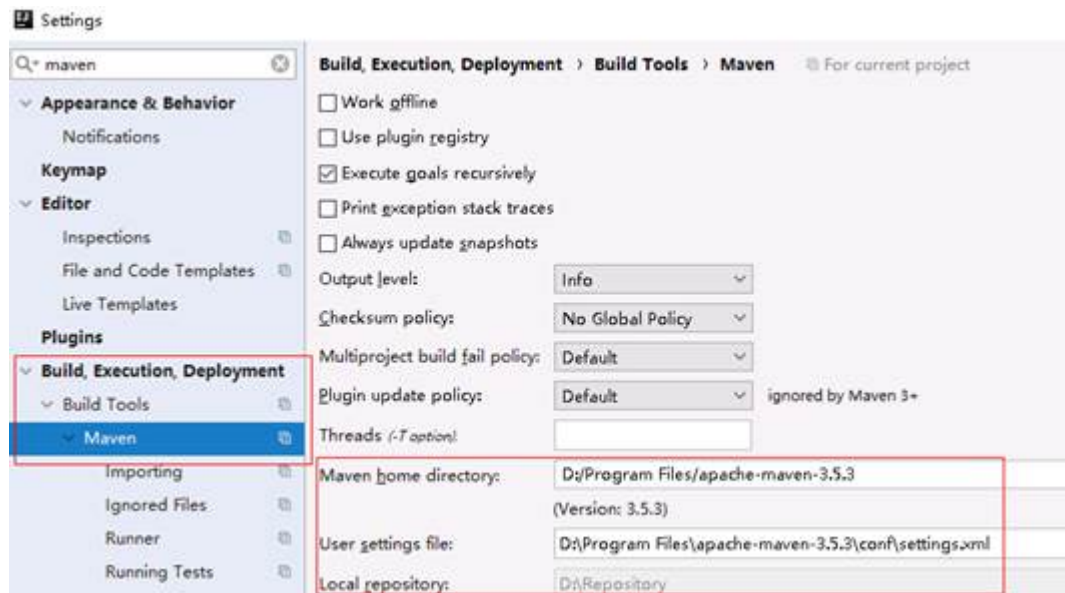
步骤4 在新打开的IDEA窗口中，单击“File > Settings”。



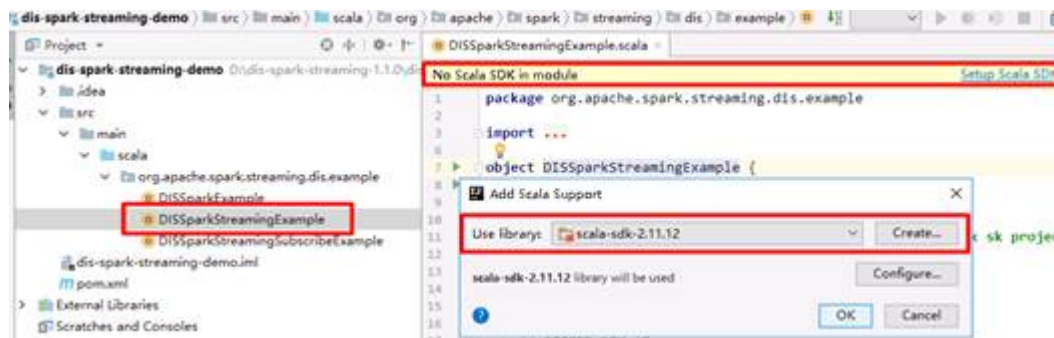
步骤5 在搜索框搜索maven，找到maven的配置，请确保Maven home directory(Maven安装路径)，User settings file (settings.xml文件位置)和Local repository(本地仓库地址)配置正确。

说明

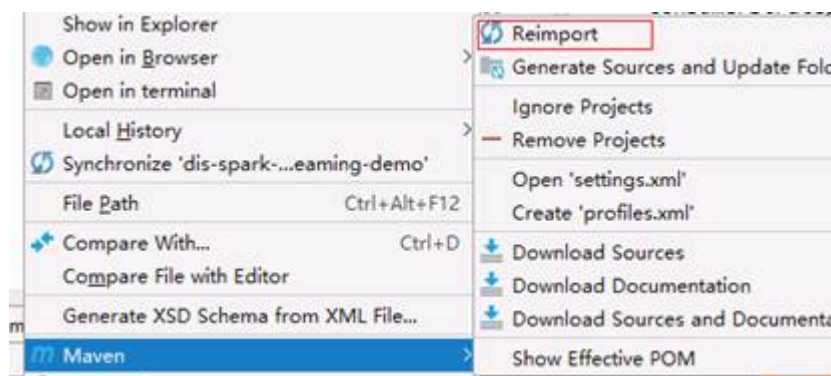
若不正确，请修改，否则**步骤2**中安装的sdk无法找到。



步骤6 打开DISparkStreamingExample文件，如果IDEA提示“ No Scala SDK in module”，请单击旁边的“Setup Scala SDK”，会显示Scala SDK列表(如果没有可以创建一个并关联scala路径)，选择2.11版本的即可。



步骤7 在pom.xml上单击右键，选择“Maven > Reimport”，重新引入maven依赖库。



步骤8 此时IDEA打开的DISSparkStreamingExample文件内没有错误即表示开发环境配置成功，此文件的逻辑是读取DIS通道中的数据并统计每个单词出现次数。

1. DISSparkStreamingExample是一个使用Assign模式的样例，不具备停止再启动时从上一次停止位置开始的能力。使用到的SDK构造方法如下：

```
ConsumerStrategies.Assign[String, String](streamName, params, startingOffsets)
```

- streamName为DIS通道名称。
- params为参数Map集合，至少包括endpoint（DIS网关地址），region（区域），ak（用户ak），sk（用户sk），projectId（用户项目ID）。
- startingOffsets为读取DIS数据的起始位置，LATEST表示从最新的数据开始读取；EARLIEST表示从最旧的数据开始读取；如果要指定每个分区的精确起始位置，则可以写为json字符串，例如{"0":23,"1":-1,"2":-2}表示第0分区起始位置是23，第1分区从最新数据的位置开始，第2分区从最老数据的位置开始，如果有分区没有指定位置，则默认从最新数据位置开始。

2. DISSparkStreamingSubscribeExample是一个使用Subscribe模式的样例，具备停止再启动时从上一次停止位置开始的能力。使用到的SDK构造方法如下：

```
ConsumerStrategies.Subscribe[String, String](Array(streamName), params)
```

- streamName为DIS通道名称
- params为参数Map集合，至少包括endpoint（DIS网关地址），region（区域），ak（用户ak），sk（用户sk），projectId（用户项目ID），group.id（app名称，表示某一个消费组）；还可以包含auto.offset.reset，参数含义同Assign模式下的startingOffsets；另外一个参数enable.auto.commit，设置为true会自动每隔5000ms(可通过设置auto.commit.interval.ms修改)提交一次offset，设置为false则不自动提交，用户可以手动调用commitAsync提交，参见示例代码中如下部分

```
stream.foreachRDD { rdd =>
    val offsetRanges = rdd.asInstanceOf[HasOffsetRanges].offsetRanges
```




```
// commit offset to DIS async.
stream.asInstanceOf[CanCommitOffsets].commitAsync(offsetRanges)
}
```

----结束

验证 sparkStreaming 作业

实际场景中，SparkStreaming作业需要提交在Spark集群上运行，但本次验证只介绍在本地IDE上测试，目的是了解sdk基本使用方法。测试完成后用户可自行创建集群（如MRS集群）并提交作业验证。

步骤1 使用注册账户登录[DIS控制台](#)。

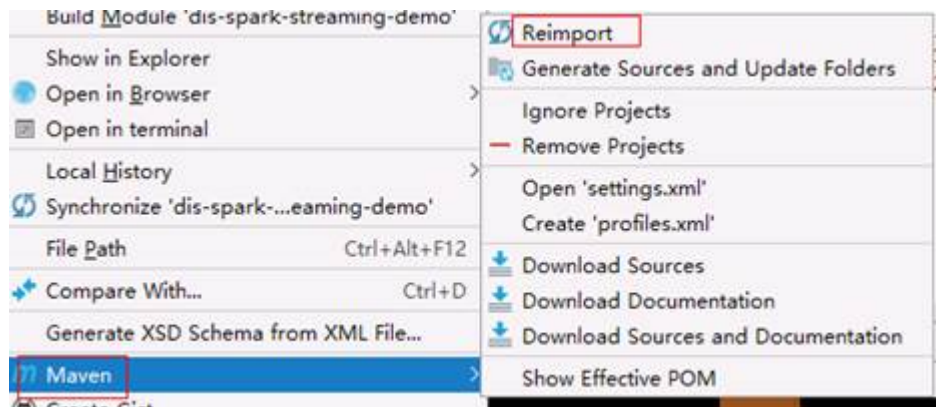
步骤2 单击管理控制台左上角的 ，选择区域和项目。

步骤3 参考**步骤1: 开通DIS通道**申请开通DIS通道，并持续上传数据到新创建的DIS通道。本次范例上传的内容为hello world。

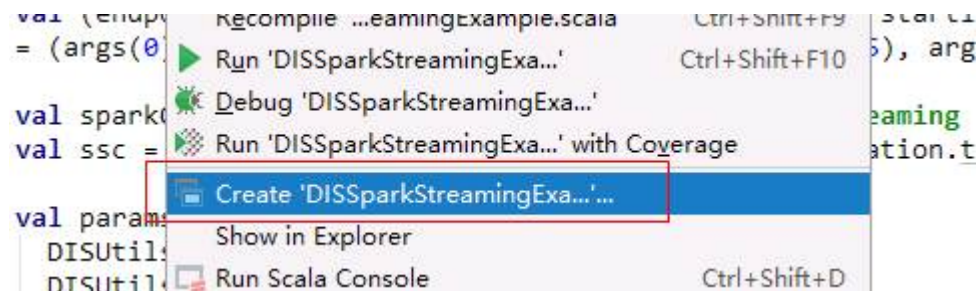
步骤4 打开pom.xml文件，选择<scope>provided</scope>这一行，并按**Ctrl+/**注释掉此行并保存。

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-streaming_${scala.binary.version}</artifactId>
  <version>${spark 2.11.version}</version>
  <!--<scope>provided</scope-->
</dependency>
```

步骤5 右键单击pom.xml，选择“Maven > Reimport”，重新引入依赖包。



步骤6 在DISparkStreamingExample文件内任意地方，右键选择“Create 'DISparkStreamingExample'”。



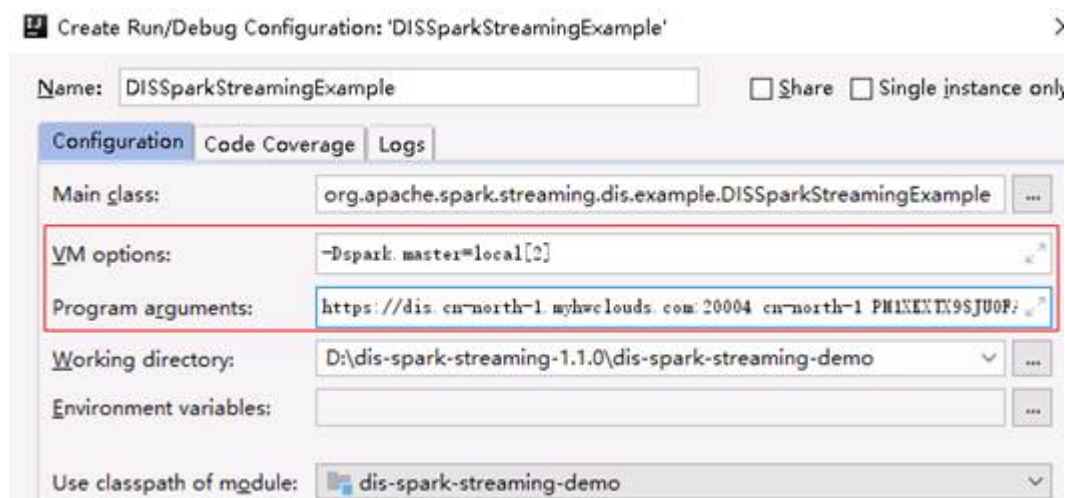
步骤7 在打开的配置页面中，“VM options”中输入-Dspark.master=local[*]，表示用local模式运行spark作业；“Program arguments”中输入运行参数，格式为：

```
DIS网关地址Region名称AK SK ProjectID 通道名称起始位置Streaming批次时间
https://dis.${region}.cloud.com ${region} YOU_AK YOU_SK YOU_PROJECTID YOU_STREAM_NAME latest 10
```

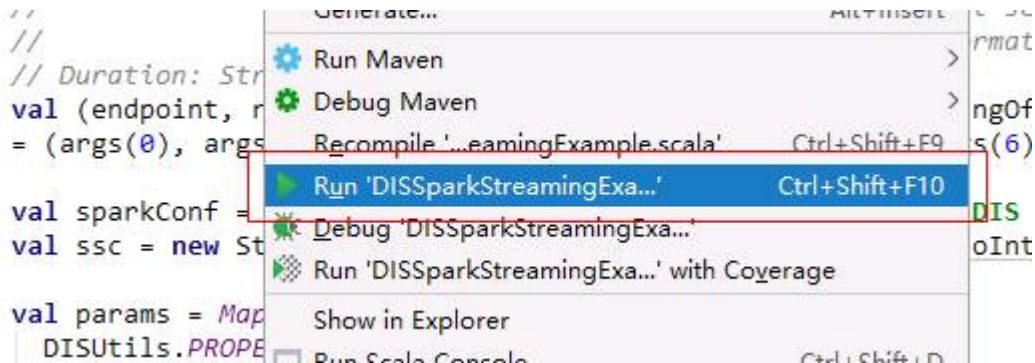
参数顺序与含义在示例代码中有，可以参考。

```
// DIS GW url
// Region ID
// Access Key Id
// Secret Access Key
// User ProjectId
// DIS stream name
// Starting offsets: 'LATEST' (Starting with the latest sequenceNumber)
// 'EARLIEST' (Starting with the earliest sequenceNumber)
// '{"0":23,"1":-1,"2":-2}' (Use json format to specify the st
// Duration: StreamingContext duration(second)
val (endpoint, region, ak, sk, projectId, streamName, startingOffsets, duration)
= (args(0), args(1), args(2), args(3), args(4), args(5), args(6), args(7))
```

最终IDEA的配置如下图所示，确认无误后单击“OK”关闭此窗口。



步骤8 在DISSparkStreamingExample文件内任意地方，右键选择“Run 'DISSparkStreamingExample'”，即可启动作业。



步骤9 启动过程中会报一个hadoop binary path的错误，这个可以忽略。

```
18/08/28 10:26:10 ERROR Shell: Failed to locate the winutils binary in the hadoop binary path
java.io.IOException: Could not locate executable null\bin\winutils.exe in the Hadoop binaries.
```

步骤10 如果没有其他错误，则作业会每隔duration运行一个批次，从DIS读取此批次内的数据并输出结果，示例如下：

```
-----  
Time: 1535423650000 ms  
-----
```

```
(hello,30)  
(world.,30)
```

步骤11 在本地运行作业验证无误之后，请把pom.xml中的<scope>provided</scope>解除注释（防止以后打包会把spark依赖也打进来），然后停止数据上传程序。

----结束

4.7 使用 DIS Flink Connector 上传与下载数据

4.7.1 DIS Flink Connector 概述

DIS Flink Connector是数据接入服务（DIS）提供的一个sdk。

DIS Flink Connector支持将DIS作为数据源创建Stream对接Flink。

4.7.2 准备 DIS Flink Connector 的相关环境

准备 DIS 应用开发环境

步骤1 参考[步骤1：开通DIS通道](#)准备相应DIS环境。

步骤2 安装Maven并配置本地仓库地址。

步骤3 安装scala-sdk。

----结束

配置 DIS Flink Connector 依赖

项目中可通过以下配置引入DIS Flink Connector依赖：

```
<dependency>  
  <groupId>com.cloud.dis</groupId>  
  <artifactId>cloud-dis-flink-connector_2.11</artifactId>  
  <version>1.0.5</version>  
  <scope>compile</scope>  
</dependency>
```

检查认证信息

- 检查AK/SK
AK/SK (Access Key ID/Secret Access Key)是用户调用接口的访问密钥。由用户在IAM中创建，可在“我的凭证 > 管理访问密钥”页面下载生成。
- 检查项目ID
ProjectID表示租户的资源，每个Region都有一个唯一的项目ID。可在页面查看不同Region对应的项目ID值。

4.7.3 自定义 Flink Streaming 作业

获取 DIS Flink Connector Demo

步骤1 [这里](#)获取 “dis-flink-connector-X.X.X.zip” 压缩包。解压 “dis-flink-connector-X.X.X.zip” 压缩包，解压之后获得以下目录：

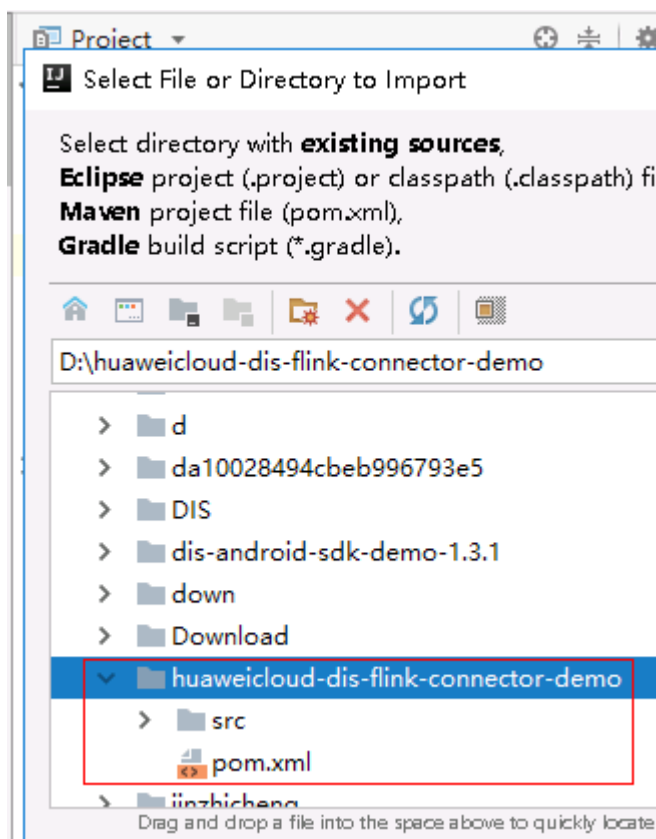
- “huaweicloud-dis-flink-connector-demo” 目录包含一个Maven工程样例。
- 结束

IntelliJ IDEA 中导入 Demo 工程

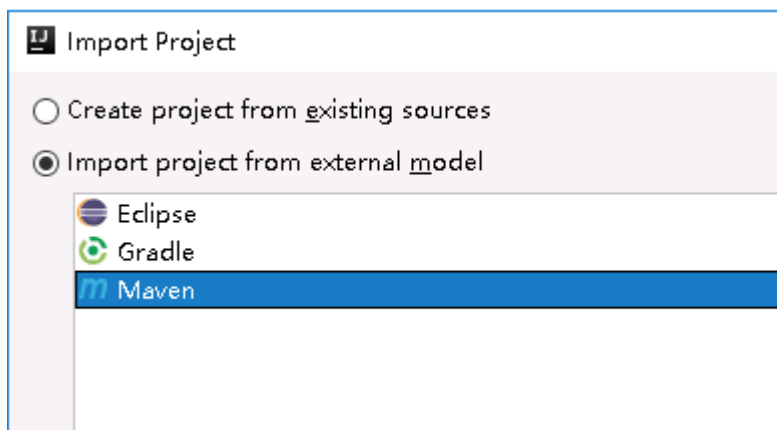
以IntelliJ IDEA社区版为例，说明如何编写Flink作业。请先确保在IDEA上已经正确配置好。

- JDK 1.8+
- Scala-sdk-2.11
- Maven 3.3.*

步骤1 打开IntelliJIDEA，选择 “File > New >> Project from Existing Sources...”。选择解压至本地的huaweicloud-dis-flink-connector-demo目录，单击确认。



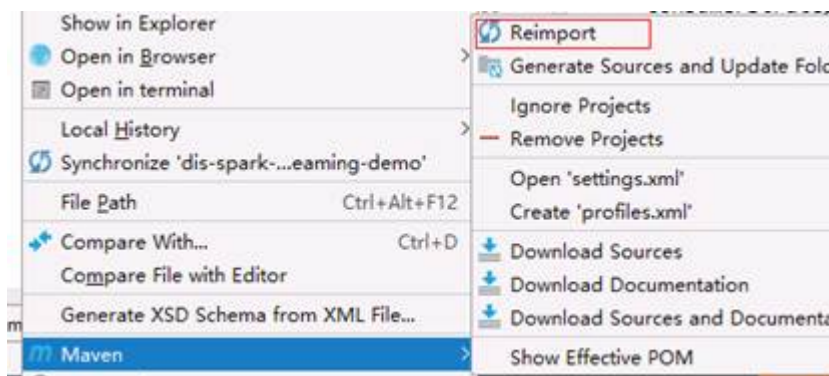
步骤2 选择导入Maven工程，保持默认配置，一直单击下一步即可。



步骤3 单击“New Window”，在新窗口打开此工程。



步骤4 在pom.xml上单击右键，选择“Maven > Reimport”，重新引入maven依赖库。




----结束

验证 Flink Streaming Source 作业

实际场景中，Flink Streaming作业需要提交在Flink集群上运行，但本次验证只介绍在本地IDE上测试，目的是了解sdk基本使用方法。测试完成后用户可自行创建集（如MRS集群）并提交作业验证。

步骤1 使用注册账户登录[DIS控制台](#)。

步骤2 单击管理控制台左上角的 ，选择区域和项目。

步骤3 参考**步骤1：开通DIS通道**申请开通DIS通道，并持续上传数据到新创建的DIS通道。本次范例上传的内容为hello world。

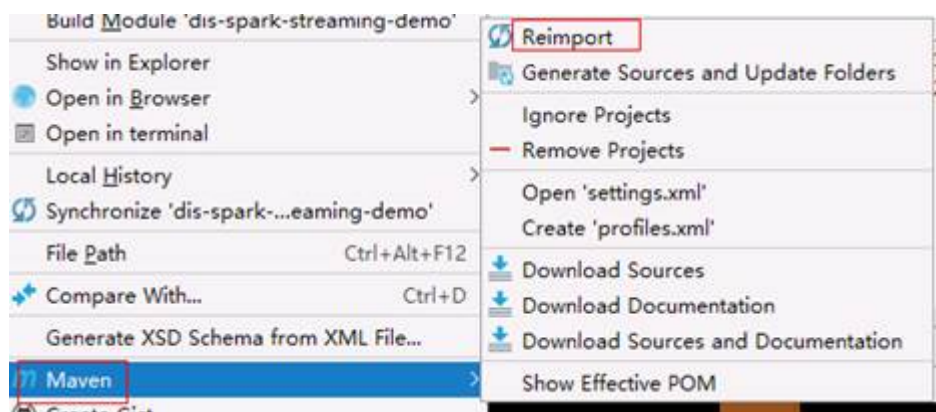
步骤4 打开pom.xml文件，选择<scope>provided</scope>这一行，并按**Ctrl+/**注释掉此行并保存。

```

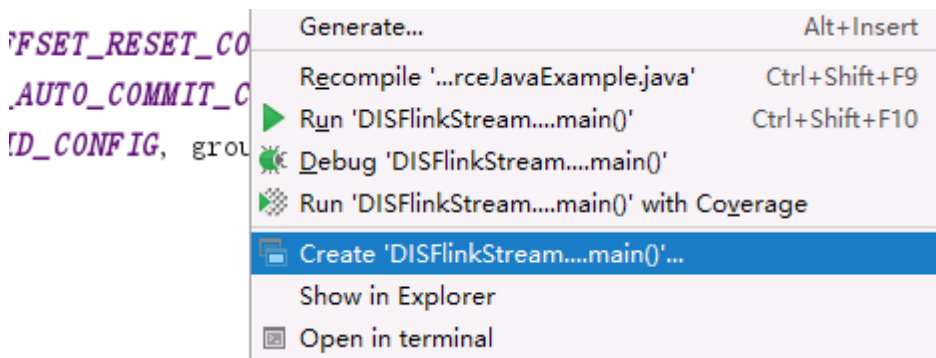
<!-- streaming-java dependencies -->
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-streaming-java_2.11</artifactId>
  <version>1.5.3</version>
  <!-- please comment this if run demo locally -->
  <!--<scope>provided</scope>-->
</dependency>

```

步骤5 右键单击pom.xml，选择“Maven > Reimport”，重新引入依赖包。



步骤6 在DISFlinkStreamingSourceJavaExample文件内任意地方，右键选择“Create 'DISFlinkStreamingSourceJavaExample'”。



步骤7 在打开的配置页面中，“Program arguments”中输入运行参数，格式为：

DIS网关地址 Region名称 AK SK ProjectID 通道名称 起始位置 消费者标识

https://dis.{\$region}.myhuaweicloud.com {\$region} YOU_AK YOU_SK YOU_PROJECTID YOU_STREAM_NAME latest GROUP_ID

参数顺序与含义在示例代码中有，可以参考。

```

// DIS终端节点
String endpoint;
// DIS服务所在区域ID
String region;
//
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，或者使用交互方式传参，确保安全；

```

```
// 本示例以交互式方式。
System.out.print("Enter your Access Key: ");
String ak = scanner.nextLine();
System.out.print("Enter your Secret Key: ");
String sk = scanner.nextLine();
// 用户的项目ID
String projectId;
// DIS通道名称
String streamName;
// 消费策略，只有当分区没有Checkpoint或者Checkpoint过期时，才会使用此配置的策略；如果存在有效的Checkpoint，则会从此Checkpoint开始继续消费
// 取值有： LATEST 从最新的数据开始消费，此策略会忽略通道中已有数据
// EARLIEST 从最老的数据开始消费，此策略会获取通道中所有的有效数据
String startingOffsets;
// 消费组标识，同一个消费组下的不同客户端可以同时消费同一个通道
String groupId;
```

说明

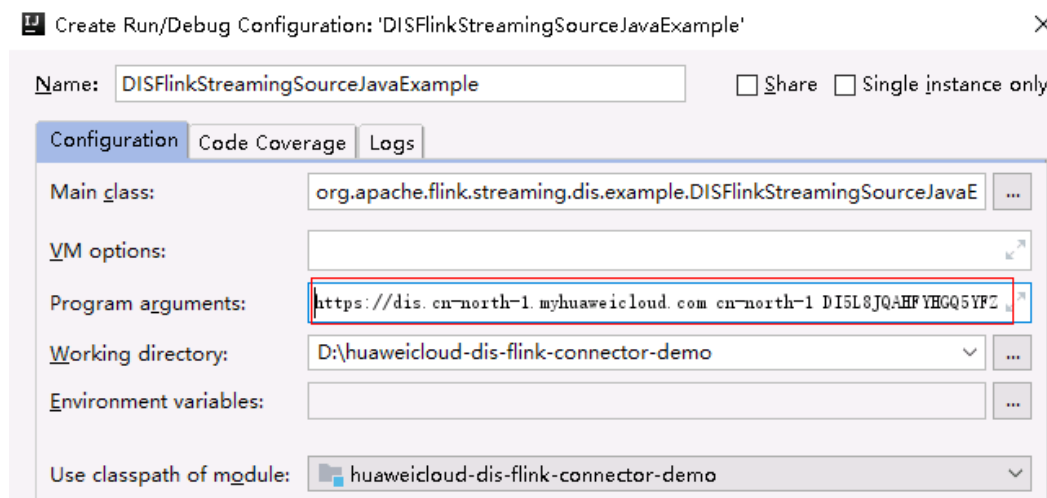
- 断点消费必须指定checkpoint或者按照如下设置自动打上消费点。

```
disConfig.put(DisConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "true");
disConfig.put(DisConsumerConfig.AUTO_COMMIT_INTERVAL_MS_CONFIG, "1000");
```

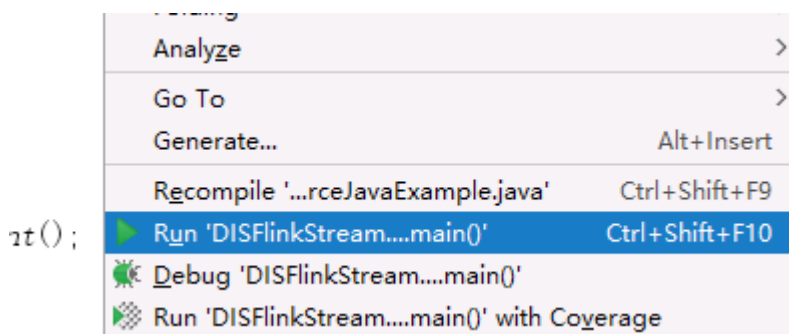
```
disConfig.put(DisConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "LATEST"); //LATEST 表示从最新的数据开始消费。
```

- 如果都不设置 默认的是从lastest开始消费。

最终IDEA的配置如下图所示，确认无误后单击“OK”关闭此窗口。



步骤8 在DISFlinkStreamingSourceJavaExample文件内任意地方，右键选择“Run 'DISFlinkStreamingSourceJavaExample'”，即可启动作业。



步骤9 如果没有其他错误，将从DIS读取数据并输出到控制台，示例如下：

```
2> hello world
2> hello world
2> hello world
```


步骤10 在本地运行作业验证无误之后，请把pom.xml中的<scope>provided</scope>解除注释(防止以后打包会把flink依赖也打进来)，然后停止数据上传程序。

---结束

验证 Flink Streaming Sink 作业

实际场景中，Flink作业需要提交在Flink集群上运行，但本次验证只介绍在本地IDE上测试，目的是了解sdk基本使用方法。测试完成后用户可自行创建集群（如MRS集群）并提交作业验证。

步骤1 使用注册账户登录[DIS控制台](#)。

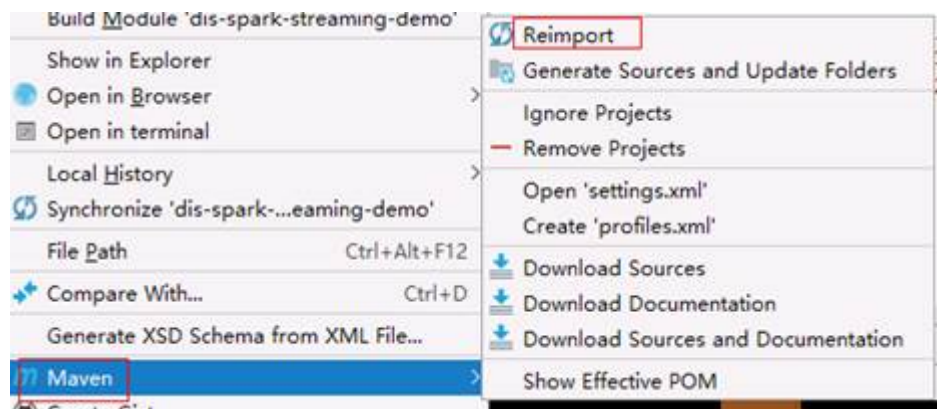
步骤2 单击管理控制台左上角的 ，选择区域和项目。

步骤3 参考**步骤1：开通DIS通道**申请开通DIS通道。

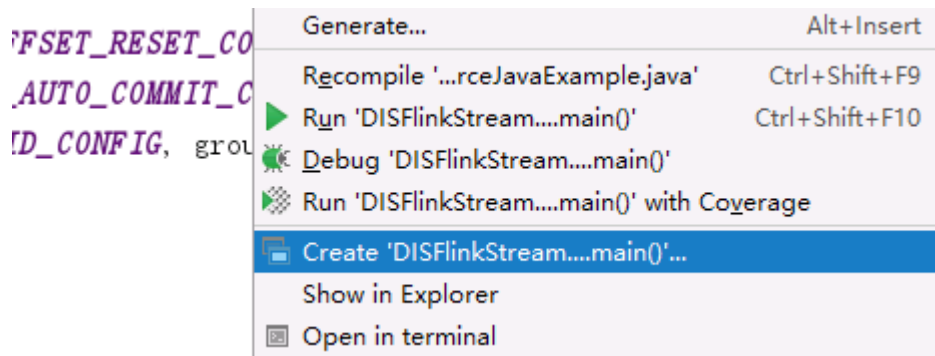
步骤4 打开pom.xml文件，选择<scope>provided</scope>这一行，并按Ctrl+/注释掉此行并保存。

```
<!-- streaming-java dependencies -->
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-streaming-java_2.11</artifactId>
  <version>1.5.3</version>
  <!-- please comment this if run demo locally -->
  <!--<scope>provided</scope>-->
</dependency>
```

步骤5 右键单击pom.xml，选择“Maven > Reimport”，重新引入依赖包。



步骤6 在DISFlinkStreamingSinkJavaExample文件内任意地方，右键选择“Create 'DISFlinkStreamingSinkJavaExample'”。



步骤7 在打开的配置页面中，“Program arguments”中输入运行参数，格式为：

```
DIS网关地址 Region名称 AK SK ProjectID 通道名称
https://dis.${region}.myhuaweicloud.com ${region} YOU_AK YOU_SK YOU_PROJECTID YOU_STREAM_NAME
```

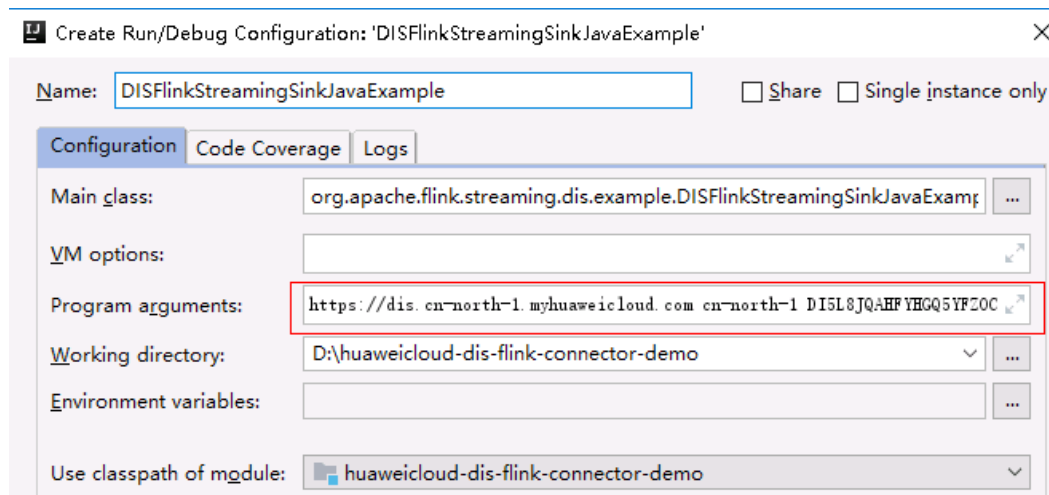
参数顺序与含义在示例代码中有，可以参考。

```
// DIS终端节点
String endpoint;
// DIS服务所在区域ID
String region;

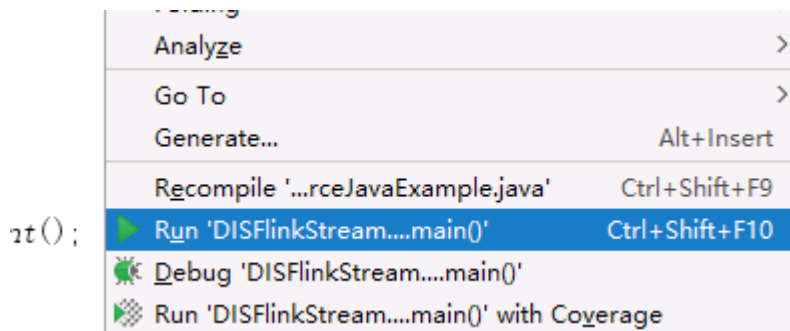
// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密
文存放，或者使用交互方式传参，确保安全；
// 本示例以交互式方式。
System.out.print("Enter your Access Key: ");
String ak = scanner.nextLine();

System.out.print("Enter your Secret Key: ");
String sk = scanner.nextLine();
// 用户的项目ID
String projectId;
// DIS通道名称
String streamName;
```

最终IDEA的配置如下图所示，确认无误后单击“OK”关闭此窗口。



步骤8 在DISFlinkStreamingSinkJavaExample文件内任意地方，右键选择“Run 'DISFlinkStreamingSinkJavaExample'”，即可启动作业。



步骤9 如果没有其他错误，可以到DIS控制台通道监控页面查看数据是否上传成功。

步骤10 在本地运行作业验证无误之后，请把pom.xml中的<scope>provided</scope>解除注释(防止以后打包会把flink依赖也打进来)，然后停止数据上传程序。


----结束

5 管理转储任务

5.1 新增转储任务

将数据发送到DIS通道后，通过为通道添加转储任务，数据将自动传输到您选择的目标。

步骤1 使用注册账户登录[DIS控制台](#)。

步骤2 单击管理控制台左上角的 ，选择区域和项目。

步骤3 在左侧列表栏中选择。

步骤4 单击需要查看的通道名称，进入所选通道的管理页面，选择“转储管理”页签。也可选中待查看通道名称对应操作列，选择“更多”下拉列表中的“查看转储任务”。

步骤5 单击“添加转储任务”按钮，在弹出的“添加转储任务”页面配置转储相关配置项。具体参数解释请参见。

说明

每个通道最多可创建5个转储任务。

步骤6 单击“立即创建”。

步骤7 在对应“任务名称”的操作列单击“更多 > 查看转储日志”，查看该通道的转储详情。转储参数说明如[表5-1](#)所示。

表 5-1 DIS 转储日志参数说明


参数	说明
开始时间	<p>用户转储日志开始构建的时间。</p> <p>格式：YYYY/MM/dd HH:mm:ss GMT</p> <ul style="list-style-type: none"> • YYYY：表示年份 • MM：表示月份 • dd：表示日期 • HH：表示小时 • mm：表示分钟 • ss：表示秒 • GMT：表示时区
结束时间	<p>用户转储日志构建完成时间。</p> <p>格式：YYYY/MM/dd HH:mm:ss GMT</p> <ul style="list-style-type: none"> • YYYY：表示年份 • MM：表示月份 • dd：表示日期 • HH：表示小时 • mm：表示分钟 • ss：表示秒 • GMT：表示时区
状态	<p>日志转储的状态。</p> <ul style="list-style-type: none"> • 已完成 • 失败 • 异常
转储文件名	<p>转储到目标服务的文件名称。从通道内读取的用户记录会写入文件后，再通过文件的形式转储到目标服务（如OBS）</p>
记录数	<p>用户转储日志开始构建到构建完成的时间间隔内，上传的记录条数。</p>
数据量（字节）	<p>用户转储日志开始构建到构建完成的时间间隔内，上传数据的大小。</p> <p>单位：字节</p>
操作	<p>转储失败的详情。</p> <ul style="list-style-type: none"> • “状态”为“已完成”，该列不可操作。 • “状态”为“失败”，单击“查看详情”查看转储失败详情。 • “状态”为“异常”，单击“查看详情”查看转储失败详情。

----结束

修改和启用转储任务

用户在创建通道并增加转储任务成功后，支持对已创建的通道属性进行修改。

步骤1 使用注册账户登录DIS控制台。

步骤2 单击管理控制台左上角的 ，选择区域和项目。

步骤3 在左侧列表栏中选择“通道管理”。

步骤4 单击需要查看的通道名称。进入所选通道的管理页面，选择“转储管理”页签。也可选择待查看通道名称对应操作列，选择“更多”下拉列表中的“查看转储任务”。

步骤5 对于已添加转储任务的通道，在对应“任务名称”的操作列：

1. 单击“更多 >修改”，可修改转储任务。
2. 单击“更多 >启用”，可启用转储任务。
3. 单击“更多 >暂停”，可暂停转储任务。

----结束

5.2 转储至 OBS

源数据类型 JSON/BLOB/CSV==>转储文件格式 Text

表 5-2 转储 Text 格式文件的配置参数

参数	说明	取值
任务名称	用户创建转储任务时，需要指定转储任务名称，同一通道的转储任务名称不可重复。任务名称由英文字母、数字、中划线和下划线组成。长度为1~64个字符。	-
数据转储地址	存储该通道数据的OBS桶名称。桶名称在“对象存储服务”中“创建桶”时创建。	-
转储文件目录	在OBS中存储通道文件的自定义目录，多级目录可用“/”进行分隔，不能以“/”开头。 取值范围：0~50个字符。 默认配置为空。	-

参数	说明	取值
时间目录格式	<p>数据将存储在OBS桶中转储文件目录下，按时间格式作为层级的目录中。</p> <p>当选择的时间目录格式精确到日时，存储目录为“桶名称/转储文件目录/年/月/日”。</p> <p>取值范围：</p> <ul style="list-style-type: none"> • N/A：置空，不使用日期时间目录。 • yyyy：年 • yyyy/MM：年/月 • yyyy/MM/dd：年/月/日 • yyyy/MM/dd/HH：年/月/日/时 • yyyy/MM/dd/HH/mm：年/月/日/时/分 <p>此配置项仅支持选择，不可手动输入。</p>	-
记录分隔符	<p>进行OBS转储时，分隔不同转储记录的分隔符。</p> <p>取值范围：</p> <ul style="list-style-type: none"> • 逗号 "," • 分号 ";" • 竖线 " " • 换行符 "\n" • NULL <p>此配置项仅支持选择，不可手动输入。</p>	-
偏移量	<ul style="list-style-type: none"> • 最新：最大偏移量，即获取最新的有效数据。 • 最早：最小偏移量，即读取最早的有效数据。 	最新
数据转储周期	<p>根据用户配置的时间，周期性的将数据导入目的地（OBS，MRS，DLI，DWS），若某个时间段内无数据，则此时间段不会生成打包文件。</p> <p>取值范围：30~900。</p> <p>单位：秒。</p> <p>默认配置为300秒。</p>	-

源数据类型 JSON==>转储文件格式 CSV

表 5-3 转储 CSV 格式文件的配置参数

参数	说明	取值
任务名称	用户创建转储任务时，需要指定转储任务名称，同一通道的转储任务名称不可重复。任务名称由英文字母、数字、中划线和下划线组成。长度为1~64个字符。	-
数据转储地址	存储该通道数据的OBS桶名称。桶名称在“对象存储服务”中“创建桶”时创建。	-
转储文件目录	在OBS中存储通道文件的自定义目录，多级目录可用“/”进行分隔，不能以“/”开头。 取值范围：0~50个字符。 默认配置为空。	-
时间目录格式	数据将存储在OBS桶中转储文件目录下，按时间格式作为层级的目录中。 当选择的时间目录格式精确到日时，存储目录为“桶名称/转储文件目录/年/月/日”。 取值范围： <ul style="list-style-type: none"> • N/A：置空，不使用日期时间目录。 • yyyy：年 • yyyy/MM：年/月 • yyyy/MM/dd：年/月/日 • yyyy/MM/dd/HH：年/月/日/时 • yyyy/MM/dd/HH/mm：年/月/日/时/分 此配置项仅支持选择，不可手动输入。	-
偏移量	<ul style="list-style-type: none"> • 最新：最大偏移量，即获取最新的有效数据。 • 最早：最小偏移量，即读取最早的有效数据。 	最新
数据转储周期	根据用户配置的时间，周期性的将数据导入目的地，若某个时间段内无数据，则此时间段不会生成打包文件。 取值范围：30~900。 单位：秒。 默认配置为300秒。	-

源数据类型 JSON/CSV==>转储文件格式 Parquet

📖 说明

表5-4罗列了源数据类型为JSON，CSV，转储至OBS（对应转储文件格式Parquet）时需要配置的差异化参数，共性参数的配置请参见**表5-2**。

表 5-4 转储 Parquet 格式文件的配置参数

参数	说明	取值
源数据Schema	用户的JSON或CSV数据样例，用于描述JSON或CSV数据格式。DIS可以根据此JSON或CSV数据样例生成Avro schema，将通道内上传的JSON或CSV数据转换为Parquet格式。	-
转储文件目录	<p>在OBS中存储通道文件的自定义目录，多级目录可用“/”进行分隔，不能以“/”开头。</p> <p>取值范围：0~100个字符。</p> <p>默认配置为空。</p> <p>说明 源数据类型为JSON时，支持EL表达式和内置函数。</p>	<ul style="list-style-type: none"> EL表达式示例： 源数据： <pre>{"name":"Andy","city":"","time":1556323141582}</pre> 转储文件目录配置如下： dis/basePath/app_key_p=\${name} 则数据的最终存储目录结构为： {\$桶名称}/dis/basePath/app_key_p=Andy 内置函数示例： 源数据： <pre>{"name":"Andy","city":"","time":1556323141582}</pre> 转储文件目录配置如下： dis/basePath/date_p=toDate(\${time}, "yyMMdd") 则数据的最终存储目录结构为： {\$桶名称}/dis-basePath/date_p=20190427 <p>支持的内置函数列表：</p> <p>toDate(timestamp,format)：将时间戳转化为指定的时间格式，例如： toDate(1556323141582,'yyymmdd')</p>

参数	说明	取值
自定义时间目录	<p>通过单击或来关闭或开启自定义时间戳开关。</p> <ul style="list-style-type: none"> 关闭自定义时间戳开关，则写到OBS的Object文件所在的目录层次结构，将使用转储文件的生成时间。 例如系统在2018年10月16日生成转储文件，选择“时间目录格式”精确到日，则数据转储成功后，在OBS的存储目录为“桶名称/转储文件目录/2018/10/16”。 开启自定义时间戳开关，则写到OBS的Object文件所在的目录层次结构，将使用源数据中定义的时间。 例如您在2018年10月16日创建某转储任务，选择“时间目录格式”精确到日，待上传的源数据中已定义时间字段“2017/09/08 11:01:01”，则数据转储成功后，在OBS的存储目录为“桶名称/转储文件目录/2017/09/08”。存储目录按照源数据中定义的时间字段进行定义，而不是转储文件的生成时间。 	<ul style="list-style-type: none"> 示例1：转储简单的Json数据。 源数据： <pre>{ "id": "1", "date": "2018/10/16 11:00:05" }</pre> 配置如下： 根据待转储的源数据类型，分别设置“时间戳属性名”为“date”，数据类型为“String”，时间戳格式为“yyyy/MM/dd HH:mm:ss”。 因数据转储成功后，存储的目录结构取决于源数据定义的时间戳和时间目录格式定义的年月日层级。本示例中，时间目录格式精确到日，所以数据最终存储目录结构为“桶名称/转储文件目录/2018/10/16”。 示例2：转储多层嵌套的Json数据。 源数据： <pre>{ "id": "1", "detail": { "detID": "0579011000000000103#567fd3cb13a4493eaa43076953253eed", "endTime": "2018/10/07 13:26:35" } }</pre> 配置如下： 根据待转储的源数据类型，分别设置“时间戳属性名”为“detail.endTime”，数据类型为“String”，时间戳格式为“yyyy/MM/dd HH:mm:ss”。 因数据转储成功后，存储的目录结构取决于源数据定义的时间戳和时间目录格式定义的年月日层级。本示例中，时间目录格式精确到日，所以数据最终存储目录结构为“桶名称/转储
源数据时间戳	<ul style="list-style-type: none"> 时间戳的属性名。 说明 请输入您待上传的源数据中定义的时间戳对应的字段名称。 时间戳的格式，从下拉框中选择。 yyyy/MM/dd HH:mm:ss MM/dd/yyyy HH:mm:ss dd/MM/yyyy HH:mm:ss yyyy-MM-dd HH:mm:ss MM-dd-yyyy HH:mm:ss dd-MM-yyyy HH:mm:ss 数据类型，从下拉框中选择。 <ul style="list-style-type: none"> - String - Timestamp 说明 当您待上传的源数据类型为Timestamp，请精确到毫秒级。 	

参数	说明	取值
		<p>文件目录/2018/10/07”。</p> <ul style="list-style-type: none"> • 示例3: 转储CSV格式的数据。 <p>源数据: a,2010-10-12 11:00:00,b,2011-10-12 11:00:10</p> <p>配置如下: 根据待转储的源数据, 选定时间戳“2010-10-12 11:00:00”, 经DIS转换为Parquet格式后, 对应的属性字段名称为field_1。则创建转储任务时分别设置“时间戳属性名”为“field_1”, 数据类型为“String”, 时间戳格式为“yyyy/MM/dd HH:mm:ss”。</p> <p>因数据转储成功后, 存储的目录结构取决于源数据的时间戳和时间目录格式定义的年月日层级。本示例中, 时间目录格式精确到日, 所以数据最终存储目录结构为“桶名称/转储文件目录/2010/10/12”。</p>

源数据类型 JSON/CSV==>转储文件格式 CarbonData

📖 说明

表5-5罗列了源数据类型为JSON, CSV, 转储至OBS (对应转储文件格式CarbonData) 时需要配置的差异化参数, 共性参数的配置请参见表5-2。

表 5-5 转储 CarbonData 格式文件的配置参数

参数	说明	取值
源数据Schema	用户的JSON或CSV数据样例, 用于描述JSON或CSV数据格式。DIS可以根据此JSON或CSV数据样例生成Avro schema, 将通道内上传的JSON或CSV数据转换为CarbonData格式。	-

参数	说明	取值
CarbonData检索属性	<p>carbon表属性，用于创建carbon writer。</p> <p>支持的Key如下：</p> <ul style="list-style-type: none"> • table_blocksize: 表的block大小，取值范围是1~2048MB，默认值是1024MB。 • table_blocklet_size: 文件内的Blocklet大小，默认值是64MB。 • local_dictionary_enable: 配置为true或者false，默认值是false。 • sort_columns: 指定索引列，多级索引列用“，”分隔。 • sort_scope: 加载时，数据排序的范围。目前支持如下几种： <ul style="list-style-type: none"> - local_sort: 默认值，表示在一个node下做数据排序； - no_sort: 即不排序，在需要快速入库时使用，可以在入库后系统闲时通过Compaction命令再建立索引； - batch_sort: 表示在一个node下，内存排序后直接生成CarbonData文件，不再进行node下的全排序；使用该配置，可以提升加载速度，但查询性能不如LOCAL_SORT； 	-

5.3 转储至 DLI

源数据类型 JSON/CSV

表 5-6 转储相关配置参数

参数	说明	取值
任务名称	用户创建转储任务时，需要指定转储任务名称，同一通道的转储任务名称不可重复。任务名称由英文字母、数字、中划线和下划线组成。长度为1~64个字符。	-

参数	说明	取值
DLI数据库	单击“选择”，在“选择DLI数据库”窗口选择一个数据库。 此配置项仅支持选择，不可手动输入。	-
DLI数据表	单击“选择”，在“选择DLI数据表”窗口选择一个数据表。仅支持数据位置为DLI类型的数据表，且用户需具有该表的插入权限。 此配置项仅支持选择，不可手动输入。	配置此项必须已配置“DLI数据库”。
偏移量	<ul style="list-style-type: none"> 最新：最大偏移量，即获取最新的有效数据。 最早：最小偏移量，即读取最早的有效数据。 	最新
数据转储周期	根据用户配置的时间，周期性的将数据导入目的地（OBS，MRS，DLI，DWS），若某个时间段内无数据，则此时间段不会生成打包文件。 取值范围：30~900。 单位：秒。 默认配置为300秒。	-
数据临时桶	用户数据先临时存储在OBS桶中，再转储到指定的转储服务，转储完成后临时桶中的数据会被清除。	-
数据临时目录	需要转储的数据临时存储在OBS桶下此配置项配置的目录中，转储完成后临时目录中的数据会被清除。 配置为空时，数据直接存储在OBS桶内。	-

5.4 转储至 DWS

源数据类型 JSON/CSV

表 5-7 转储相关配置参数

参数	说明	取值
任务名称	用户创建转储任务时，需要指定转储任务名称，同一通道的转储任务名称不可重复。任务名称由英文字母、数字、中划线和下划线组成。长度为1~64个字符。	-

参数	说明	取值
DWS集群	存储该通道数据的DWS集群名称。 单击“选择”，在“选择DWS集群”窗口选择一个集群。 此配置项仅支持选择，不可手动输入。	-
DWS数据库	存储该通道数据的DWS数据库名称。 手动输入，不可配置为空。	-
数据库模式	一个数据库包含一个或多个命名的模式，模式又包含表。模式还包含其他命名的对象，包括数据类型、函数，以及操作符。同一个对象名可以在不同的模式里使用而不会导致冲突。	-
DWS数据表	存储该通道数据的DWS数据库模式下的数据表。	-
数据分隔符	用户数据的字段分隔符，根据此分隔符分隔用户数据插入DWS数据表的相应列。 取值范围：不可为空	-
偏移量	<ul style="list-style-type: none"> 最新：最大偏移量，即获取最新的有效数据。 最早：最小偏移量，即读取最早的有效数据。 	最新
数据转储周期	根据用户配置的时间，周期性的将数据导入目的地（OBS，MRS，DLI，DWS），若某个时间段内无数据，则此时间段不会生成打包文件。 取值范围：30~900。 单位：秒。 默认配置为300秒。	-
用户名	DWS集群的用户名。	-
密码	DWS集群的密码。	-
KMS密钥	集群的数据库加密密钥。	-
数据临时桶	用户数据先临时存储在OBS桶中，再转储到指定的转储服务，转储完成后临时桶中的数据会被清除。	-
数据临时目录	需要转储的数据临时存储在OBS桶下此配置项配置的目录中，转储完成后临时目录中的数据会被清除。 配置为空时，数据直接存储在OBS桶内。	-

参数	说明	取值
容错选项	<p>通过单击或来关闭或开启容错选项开关。</p> <ul style="list-style-type: none"> fill_missing_fields 当数据导入时，若数据源文件中一行的最后一个字段缺失的处理方式。 取值范围：true/on，false/off。缺省值为false/off。 <ul style="list-style-type: none"> - 参数为true/on，当数据导入时，若数据源文件中一行数据的最后一个字段缺失，则把最后一个字段的值设置为NULL，不报错。 - 参数为false/off，如果最后一个字段缺失会显示如下错误信息。 ignore_extra_data 数据源文件中的字段比外表定义列数多时，是否忽略多出的列。该参数只在数据导入过程中使用。 取值范围：true/on，false/off。缺省值为false/off。 <ul style="list-style-type: none"> - 参数为true/on，若数据源文件比外表定义列数多，则忽略行尾多出来的列。 - 参数为false/off，若数据源文件比外表定义列数多，会显示如下错误信息。 <p>说明 如果行尾换行符丢失，使两行变成一行时，设置此参数为true将导致后一行数据被忽略掉。</p> compatible_illegal_chars 导入非法字符容错参数。此语法仅对READ ONLY的外表有效。 取值范围：true/on，false/off。缺省值为false/off。 <ul style="list-style-type: none"> - 参数为true/on，则导入时遇到非法字符进行容错处理，非法字符转换后入库，不报错，不中断导入。 - 参数为false/off，导入时遇到非法字符进行报错，中断导入。 <p>须知 Windows平台下OBS若按照文本格式读取数据文件，遇到0x1A会作为EOF符号结束数据读入造成解析错误，这是Windows平台的实现约束。由于OBS不支持BINARY形式读取，可将相应数据文件交由Linux平台下的OBS读取。</p> 	-

参数	说明	取值
	<p>说明</p> <ul style="list-style-type: none"> 导入非法字符容错规则如下： <ol style="list-style-type: none"> 对于'\0'，容错后转换为空格； 对于其他非法字符，容错后转换为问号； 若compatible_illegal_chars为true/on标识导入时对于非法字符进行容错处理，则若NULL、DELIMITER、QUOTE、ESCAPE设置为空格或问号则会通过如"illegal chars conversion may confuse COPY escape 0x20"等报错信息提示用户修改可能引起混淆的参数以避免导入错误。 PER NODE REJECT LIMIT 'value' 指定本次数据导入过程中每个DN实例上允许出现的数据格式错误的数量，如果有一个DN实例上的错误数量大于设定值，本次导入失败，报错退出。 取值范围：整型值，unlimited（无限），缺省值为0，有错误信息立即返回。 <p>说明</p> <p>此语法指定的是单个节点的错误容忍度。</p> <p>数据格式错误是指缺少或者多出字段值，数据类型错误或者编码错误。对于非数据格式错误，一旦发生就将导致整个数据扫描失败。</p>	

5.5 转储至 MRS

操作前提

转储至MRS时，不支持MRS集群3.x及以上版本，且MRS集群不能开启Kerberos认证。

源数据类型 JSON/BLOB/CSV==>转储文件格式 Text

表 5-8 转储 Text 格式文件的配置参数

参数	说明	取值
任务名称	用户创建转储任务时，需要指定转储任务名称，同一通道的转储任务名称不可重复。任务名称由英文字母、数字、中划线和下划线组成。长度为1~64个字符。	-
MRS集群	单击“选择”，在“选择集群”窗口选择一个MRS集群。仅支持转储至非Kerberos认证的MRS集群。 此配置项仅支持选择，不可手动输入。	-

参数	说明	取值
HDFS路径	单击“选择”，在“选择HDFS文件路径”窗口按层级选择所需HDFS文件所在路径。 此处路径仅支持选择，不可手动输入。	配置此项必须已配置“MRS集群”。
转储文件目录	在MRS中存储通道文件的自定义目录，多级目录可用“/”进行分隔，不能以“/”开头。 取值范围：0~50个字符。 默认配置为空。	-
偏移量	<ul style="list-style-type: none"> 最新：最大偏移量，即获取最新的有效数据。 最早：最小偏移量，即读取最早的有效数据。 	最新
数据转储周期	根据用户配置的时间，周期性的将数据导入目的地（OBS，MRS，DLI，DWS），若某个时间段内无数据，则此时间段不会生成打包文件。 取值范围：30~900。 单位：秒。 默认配置为300秒。	-
数据临时桶	用户数据先临时存储在OBS桶中，再转储到指定的转储服务，转储完成后临时桶中的数据会被清除。	-
数据临时目录	需要转储的数据临时存储在OBS桶下此配置项配置的目录中，转储完成后临时目录中的数据会被清除。 配置为空时，数据直接存储在OBS桶内。	-

源数据类型 JSON/CSV==>转储文件格式 Parquet

说明

表5-9列出了源数据类型为JSON，CSV，转储至MRS（对应转储文件格式Parquet）时需要配置的差异化参数，共性参数的配置请参见**表5-8**。

表 5-9 转储 Parquet 格式文件的配置参数

参数	说明	取值
源数据Schema	用户的JSON或CSV数据样例，用于描述JSON或CSV数据格式。DIS可以根据此JSON或CSV数据样例生成Avro schema，将通道内上传的JSON或CSV数据转换为Parquet格式。	-

源数据类型 JSON/CSV==>转储文件格式 CarbonData

说明

表5-10罗列了源数据类型为JSON，CSV，转储至OBS（对应转储文件格式CarbonData）时需要配置的差异化参数，共性参数的配置请参见**表5-8**。

表 5-10 转储 CarbonData 格式文件的配置参数

参数	说明	取值
源数据Schema	用户的JSON或CSV数据样例，用于描述JSON或CSV数据格式。DIS可以根据此JSON或CSV数据样例生成Avro schema，将通道内上传的JSON或CSV数据转换为CarbonData格式。	-
CarbonData检索属性	<p>carbon表属性，用于创建carbon writer。</p> <p>支持的Key如下：</p> <ul style="list-style-type: none"> • table_blocksize：表的block大小，取值范围是1~2048MB，默认值是1024MB。 • table_blocklet_size：文件内的Blocklet大小，默认值是64MB。 • local_dictionary_enable：配置为true或者false，默认值是false。 • sort_columns：指定索引列，多级索引列用“，”分隔。 • sort_scope：加载时，数据排序的范围。目前支持如下几种： <ul style="list-style-type: none"> - local_sort：默认值，表示在一个node下做数据排序； - no_sort：即不排序，在需要快速入库时使用，可以在入库后系统闲时通过Compaction命令再建立索引； - batch_sort：表示在一个node下，内存排序后直接生成CarbonData文件，不再进行node下的全排序；使用该配置，可以提升加载速度，但查询性能不如LOCAL_SORT； 	-

6 管理企业项目

企业项目是一种云资源管理方式。企业管理（Enterprise Management）提供面向企业客户的云上资源管理、人员管理、权限管理、财务管理等综合管理服务。区别于管理控制台上独立操控、配置云产品的方式，企业管理控制台以面向企业资源管理为出发点，帮助企业以公司、部门、项目等分级管理方式实现企业云上的人员、资源、权限、财务的管理。

已开通企业项目服务的用户，可以使用企业项目管理华为云上的云服务资源。

绑定企业项目

用户可以在创建通道时为通道选择所属的企业项目，从而将DIS通道与企业项目进行关联，详情请参见[步骤1：开通DIS通道](#)。在选择“企业项目”的下拉列表中，将显示用户在企业项目服务中已创建的项目。系统还内置了一个缺省的企业项目“default”，如果用户没有为通道选择企业项目，将使用缺省项目“default”。

在通道创建过程中，如果通道与企业项目绑定成功，则通道创建成功，如果绑定失败，系统会发送告警，通道创建失败。

当删除DIS通道时，DIS通道与企业项目的关联关系就会被自动删除。

查看企业项目

通道创建成功后，您可以在通道列表和通道基本信息页面查看通道关联的企业项目。用户只能查询到有访问权限的项目下的通道资源。

在通道管理页面的列表中，查看通道所属的企业项目。

图 6-1 查看企业项目

名称/ID	状态	通道	分区数	源数	生命周期(天)	创建时间	计费模式	企业项目	操作
hmgphwv1T K71acYgo5yykjD9TdmG	运行中	高级	1	BLOB	1	2019/07/17 18:10:18 GMT+08:00	按需计费	default	删除 更多

在通道列表中，单击通道名称，进入通道“基本信息”页面，可以查看与通道关联的企业项目。单击企业项目的名称，可以跳转到企业管理的控制台页面对该企业项目进行查看或编辑。

图 6-2 查看通道的企业项目

通道名称	hadoop11	通道ID	K71acYgo5yykjD9TdmG
状态	运行中	通道类型	高级
分区数量	1	自动扩缩容	关闭
生命周期(天)	1	创建时间	2019/07/17 18:10:18 GMT+08:00
源数据类型	BLOB	企业项目	default

同时，在企业管理的控制台中，查询指定项目中的资源列表时，也可以查询到DIS服务的资源。

按企业项目搜索通道

登录DIS管理控制台，单击“通道管理”，在通道列表上方单击“所有项目”，然后在下拉列表中选择所需搜索的项目名称，即可查看与该项目关联的所有通道。

将通道迁入或迁出企业项目

一个DIS通道只能关联一个企业项目。当通道创建成功后，可以在企业管理的控制台中，执行迁出操作，将DIS通道从当前所属的企业项目中迁出到另一个企业项目中；或者执行迁入操作，在指定的企业项目中迁入另一个企业项目中的DIS通道。迁入迁出后，DIS通道与新的企业项目进行关联，DIS通道与原企业项目的关联关系将被自动解除。详细操作，请参考《企业管理用户指南》的“企业项目管理 > 如何管理资源”章节。

7 事件通知

7.1 事件通知概述

概述

DIS使用SMN（Simple Message Notification，消息通知服务）发送DIS事件的通知，订阅DIS事件即可启用通知。在订阅中，用户需要指定一个或多个事件筛选条件。每当发生与所有筛选条件匹配的事件时，DIS就会通过该订阅发送通知。筛选条件包含事件类别（例如：管理、监控或安全）、事件级别（例如：正常或警告）和事件源类型（例如：通道或转储任务）。

支持的事件类别和事件

事件是租户通道状态发生变化的记录。它可以是由用户操作触发的（比如审计事件），也有可能是通道状态变化引起的（比如转储任务异常或转储任务恢复）。以下为当前DIS支持的事件和事件类别列表。

- 下表显示了事件源类型为通道的事件。

表 7-1 事件源类型为通道的事件

事件源类型	事件级别	事件
通道	警告	流控受限
通道	警告	通道自动扩缩容成功
通道	警告	通道自动扩缩容失败
通道	警告	通道流量异常
通道	警告	通道流量恢复

- 下表显示了事件源类型为用户的事件。

表 7-2 事件源类型为用户的事件

事件源类型	事件级别	事件
用户	警告	配额异常

- 下表显示了事件源类型为转储任务的事件。

表 7-3 事件源类型为转储任务的事件

事件源类型	事件级别	事件
转储任务	正常	转储任务恢复
转储任务	警告	转储任务异常



7.2 订阅事件通知



用户通过订阅DIS的事件通知，这样便能在特定通道或转储任务发生管理、监控或安全事件时收到通知消息。

创建订阅

- 步骤1** 使用注册账户登录[DIS控制台](#)。
- 步骤2** 单击“事件管理”。
- 步骤3** 在“事件管理”页面中单击“订阅 > 创建订阅”。
- 步骤4** 在“订阅设置”区域，设置订阅基本信息及事件过滤。
“订阅事件”区域显示系统根据订阅设置筛选出的事件。

表 7-4 订阅参数

参数名	参数解释
启用消息通知	设置是否开启事件订阅。  表示开启事件订阅，  表示关闭事件订阅，默认为关闭状态。关闭后停止发送已订阅事件的通知消息，不会删除该订阅。
订阅名称	设置订阅事件的名称。 <ul style="list-style-type: none"> • 名称只能包含大写字母、小写字母、数字、-和_，且必须由大写字母、小写字母或数字开头。 • 名称长度为1~64字符。

参数名	参数解释
订阅通道	<p>设置是否开启订阅指定通道的告警功能。</p> <p> 表示开启订阅指定通道告警功能， 表示关闭订阅指定通道告警功能，默认为关闭状态。</p> <p>开启后，配置通道名称，可以订阅指定通道的告警，避免接收其它通道不必要的告警。</p>
订阅类型	<p>支持SMN通知和DIS通道。</p> <p>说明</p> <ul style="list-style-type: none"> 当“订阅类型”设置为“SMN通知”，请参见步骤5选择消息通知主题。 当“订阅类型”设置为“DIS通道”，请参见5选择通道。

步骤5 在“SMN主题”下拉框中，选择合适的消息通知主题。

用户可以根据需要，通过以下操作新建消息通知主题。

1. 单击“去创建SMN主题”，系统将跳转到消息通知服务的“主题”页面，用户可通过单击页面右上方的“创建主题”来新建主题，具体请参见《消息通知服务用户指南》中的“创建主题”章节。
2. 在创建的主题行，单击“更多 > 设置主题策略”，“可发布消息的服务”处勾选“DIS”，使消息通知服务发布DIS消息。
3. 在创建的主题行，单击“添加订阅”，向该主题添加订阅，具体请参见《消息通知服务用户指南》中的“添加订阅”章节。

步骤6 单击“确认”，完成创建订阅。

----结束

修改订阅

步骤1 在“事件管理”页面中单击“订阅”。

步骤2 在指定订阅名称所在行的“操作”列，单击“更多 > 修改”。

步骤3 在“订阅设置”页面，选择要更改的参数项进行修改。具体修改方法参见创建订阅中的[步骤4](#)~[步骤6](#)。

----结束

删除订阅

步骤1 在“事件管理”页面中单击“订阅”。

步骤2 在指定订阅名称所在行的“操作”列，单击“更多 > 删除”，弹出确认删除对话框，如[图7-1](#)所示。

图 7-1 确认删除



步骤3 单击“确定”，删除该订阅。

----结束

7.3 查看事件


介绍用户如何查找通道或转储任务发生的事件。

步骤1 在“事件管理”页面中单击“事件”，默认显示当前所有通道或转储任务已发生的事件。

步骤2 在事件列表右上方的下拉列表中通过选择不同的筛选条件搜索事件。可从事件级别和事件源两个维度进行筛选。

- 在下拉列表中选择“所有事件级别”、“正常”或“警告”。
- 在下拉列表中选择“事件源”，在输入框中输入通道或转储任务名称，例如“demo”。

步骤3 单击 ，显示筛选后的事件查询结果。

步骤4 单击“事件”右侧的 ，选择事件名称，例如“转储任务恢复”，可过滤对应的事件。

----结束

8 使用 CES 监控 DIS

8.1 支持的监控指标

功能说明

本节定义了数据接入服务上报云监控的监控指标的命名空间，监控指标列表和维度，用户可以通过云监控检索数据接入服务产生的监控指标和告警信息。

命名空间

SYS.DAYU

监控指标

DIS通道支持的监控指标如[表8-1](#)所示。

表 8-1 DIS 的监控指标

指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
总输入流量	该指标用于统计指定时间范围内，通道上传数据量。 单位：byte/s。	≥ 0 bytes/s	通道	1分钟
总输出流量	该指标用于统计指定时间范围内，通道下载数据量。 单位：byte/s。	≥ 0 bytes/s	通道	1分钟
总输入记录数	该指标用于统计指定时间范围内，通道上传记录数。 单位：Count/s。	≥ 0 Count/s	通道	1分钟

指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
总输出记录数	该指标用于统计指定时间范围内，通道下载记录数。 单位：Count/s。	≥ 0 Count/s	通道	1分钟
上传请求成功数	该指标用于统计指定时间范围内，通道上传请求成功次数。 单位：Count/s。	≥ 0 Count/s	通道	1分钟
下载请求成功数	该指标用于统计指定时间范围内，通道下载请求成功次数。 单位：Count/s。	≥ 0 Count/s	通道	1分钟
上传请求平均处理时间	该指标用于统计指定时间范围内，通道上传请求平均时延。 单位：ms。	0~50ms	通道	1分钟
下载请求平均处理时间	该指标用于统计指定时间范围内，通道下载请求平均时延。 单位：ms。	0~50ms	通道	1分钟
因流控拒绝的上传请求数	该指标用于统计指定时间范围内，通道由于流控而拒绝的上传请求数。 单位：Count/s。	0~1Count/s	通道	1分钟
因流控拒绝的下载请求数	该指标用于统计指定时间范围内，通道由于流控而拒绝的下载请求数。 单位：Count/s。	0~1Count/s	通道	1分钟

维度

Key	Value
stream_id	实时数据接入

8.2 设置告警规则

操作场景

通过设置DIS通道告警规则，用户可自定义监控目标与通知策略，及时了解DIS通道运行状况，从而起到预警作用。

设置DIS通道的告警规则包括设置告警规则名称、监控对象、监控指标、告警阈值、监控周期和是否发送通知等参数。本节介绍了设置DIS通道告警规则的具体方法。

操作步骤

- 步骤1** 登录管理控制台。
- 步骤2** 选择“管理与监管 > 云监控服务 CES”。
- 步骤3** 在云监控服务的左侧导航树栏，选择“告警 > 告警规则”，在页面右侧单击“创建告警规则”。
- 步骤4** 根据界面提示设置DIS通道的告警规则，当前仅支持“自定义创建”。
- 步骤5** 设置完成后，单击“立即创建”。当符合规则的告警产生时，系统会自动进行通知。

📖 说明

更多关于DIS通道监控规则的信息，请参见《[云监控用户指南](#)》。

----结束

8.3 查看监控指标

操作场景

云监控服务可以对数据接入服务通道的运行状态进行日常监控。您可以通过云监控管理控制台，直观地查看各项监控指标。

由于监控数据的获取与传输会花费一定时间，因此，云监控显示的是当前时间5~10分钟前的状态。如果您的数据接入服务通道刚刚创建完成，请等待5~10分钟后查看监控数据。

前提条件


- 数据接入服务通道正常运行。

📖 说明

已删除的通道，云监控将默认该通道不存在，并在监控列表中删除，不再对其进行监控，但告警规则需要用户手动清理。

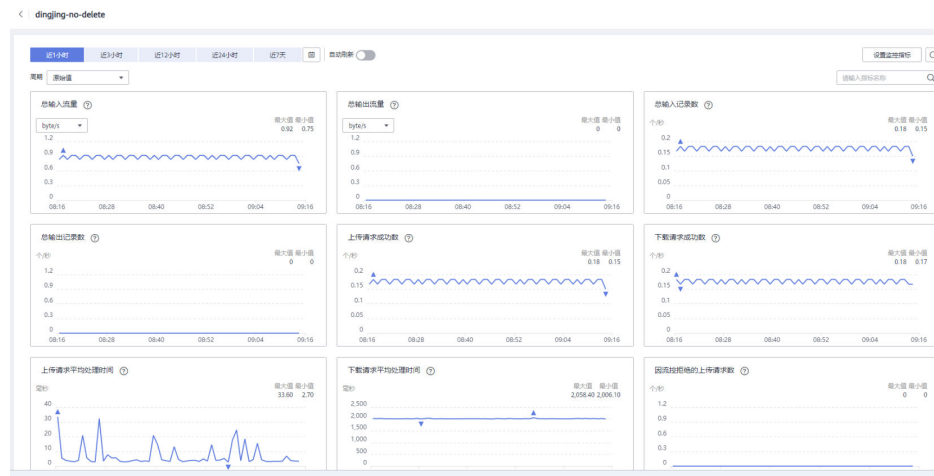
- 已在云监控页面设置告警规则，具体操作请参见[设置告警规则](#)。


操作步骤

- 步骤1** 使用注册账户登录[DIS控制台](#)。
- 步骤2** 单击管理控制台左上角的，选择区域和项目。
- 步骤3** 在左侧列表栏中选择“通道管理”。
- 步骤4** 单击需要查看监控信息的通道名称。进入监控页面。
- 步骤5** 在“通道监控”页签内单击“查看更多指标详情”，系统跳转至云监控服务的监控指标页面。

步骤6 在监控页面，可查看所有监控指标的小图。

图 8-1 查看监控指标



步骤7 单击小图右上角的 ，可进入大图模式查看。

可查看不同监控指标“近1小时”、“近3小时”、“近12小时”等周期的原始监控数据曲线图。您可以选择是否开启“自动刷新”功能，云监控服务提供了“60秒”自动刷新周期。

----结束