ALIBABA CLOUD

阿里云

DataWorks

最佳实践

文档版本：20220708

阿里云

法律声明

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

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

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

目录

1.数据迁移

1.1. IoT数据自动化同步至云端解决方案

物联网（IoT）是一个基于互联网、传统电信网等的信息承载体，它让所有能够被独立寻址的普通物理对象形成互相连通的网络。本文将为您介绍IoT数据自动化同步至云端的解决方案。

背景信息

物联网（The Internet of Things，简称IoT）是指通过信息传感器等各种装置和技术，实时采集任何需要的信息。通过各类网络的接入，实现物与物、物与人的连接，实现对物品和过程的智能化感知、识别和管理。

物联网、大数据和云计算作为当前第三次信息化浪潮的代表技术，将在未来形成广泛的影响。物联网专注于物物相连，大数据专注于数据的价值化，云计算则为大数据和物联网提供计算资源等服务支持。

大数据是物联网体系的重要组成部分。物联网的体系结构包括设备、网络、平台、分析、应用和安全，其中分析部分的主要内容为大数据分析。大数据分析是大数据完成数据价值化的重要手段之一，而进行大数据分析的第一步是让数据成功上云。

解决方案

IoT数据自动化同步至云端解决方案主要包括存储原始数据和同步数据至分析系统两部分。

IoT设备大量的数据通常以半结构化的形式存储。例如，使用OSS存储原始信息为CSV文件。

但同步至大数据系统或传统数据库的数据，需要使用专业的数据同步系统。下图为您展示使用DataWorks数据集成完成OSS数据同步至大数据系统的解决方案流程。

1. 新建离线同步节点，详情请参见通过向导模式配置离线同步任务。
2. 选择数据来源OSS进行读取，详情请参见配置OSS Reader。
3. 选择数据去向进行写入，本文以写入MaxCompute为例，您也可以写入其它类型的数据源中。

配置自动化流程

使用OSS读取CSV文件时，需要配置读取的文件名（Object前缀）。通常IoT会不停生成数据并存储为CSV文件，如果您手动配置同步任务以读取IoT数据至云端，会较为复杂且不易实现。下文将为您介绍每5分钟生成一份CSV文件的情况下，如何自动同步数据至云端（MaxCompute）。

该解决方案需要注意的问题如下：

- OSS上的文件需要按时周期性生成。

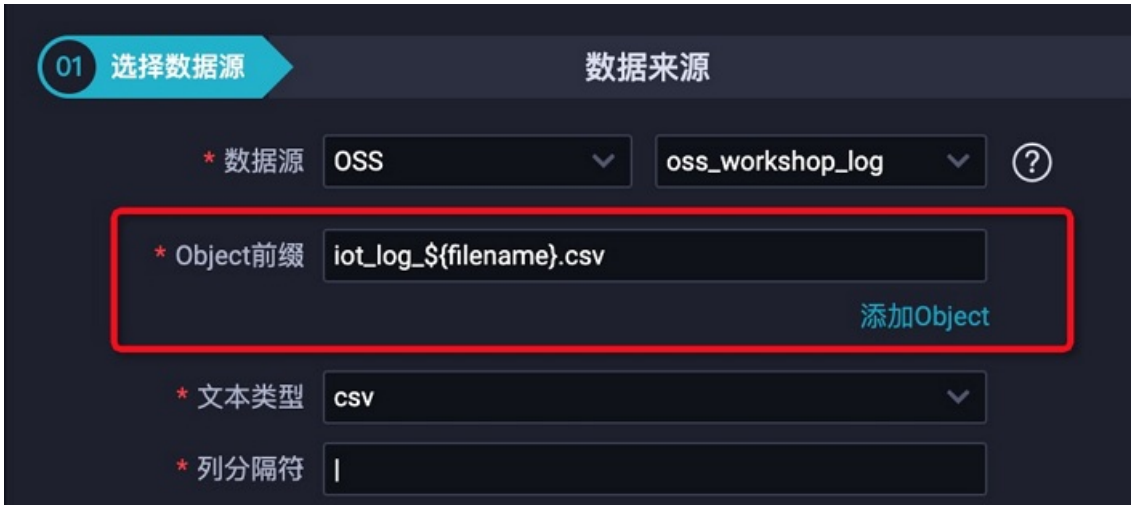
DataWorks具备按照定时时间进行周期调度的特点，您可以设置DataWorks同步任务的调度周期为OSS生成文件的周期。例如，OSS上的文件每15分钟生成一份，设置DataWorks同步任务的调度周期为每15分钟调度一次。

- 生成的文件名需要使用时间戳来命名。

  OSS同步任务在读取文件时，需要使用时间戳对文件进行命名。DataWorks通过参数变量来动态生成文件名称，以确保和OSS上的文件名称保持一致。

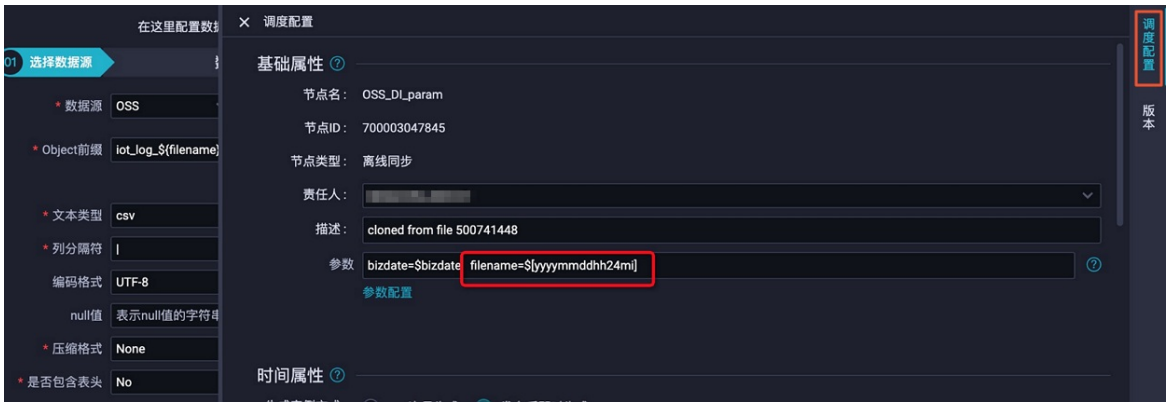
  > **说明** 推荐您使用yyyymmddhhmm等时间戳作为文件名的一部分，例如iot_log_201911062315.csv。

1. 登录DataWorks控制台，单击相应工作空间后的**进入数据集成**。
2. 新增OSS数据源和MaxCompute数据源，详情请参见配置OSS数据源和配置MaxCompute数据源。
3. 单击当前页面左上角的图标，选择**全部产品 > 数据开发**，新建业务流程，详情请参见通过脚本模式配置离线同步任务。
4. 新建离线同步节点，详情请参见通过脚本模式配置离线同步任务。
5. 在离线同步节点的编辑页面，选择数据来源，并使用参数变量作为文件名。



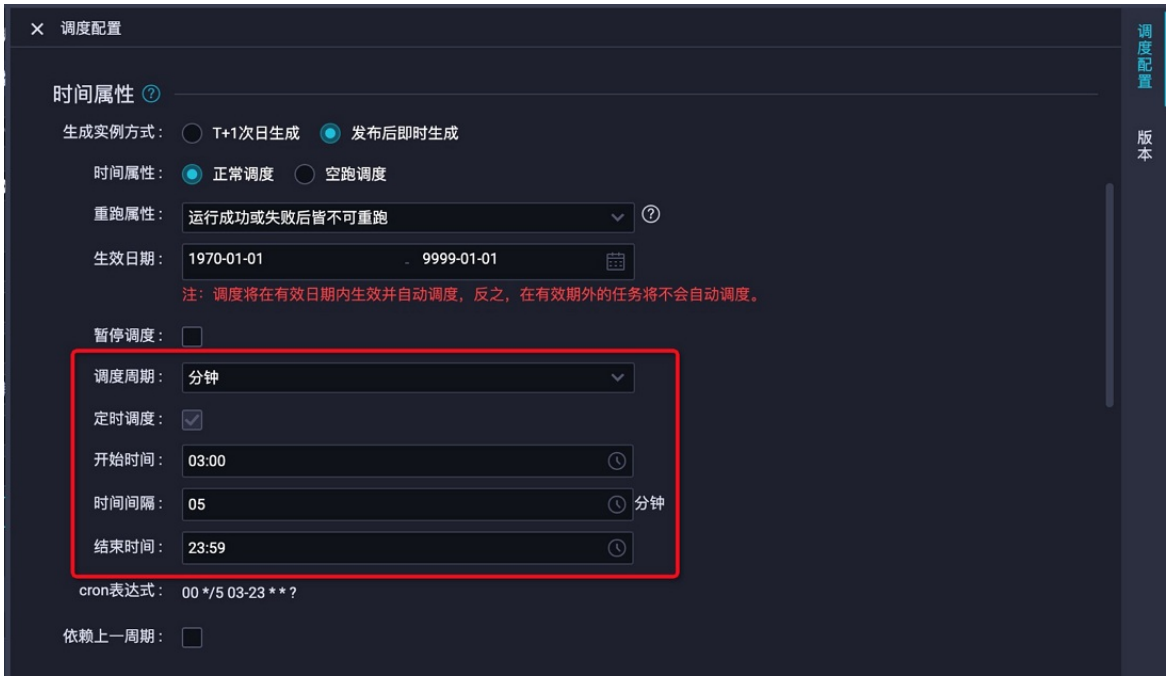
如上图所示，将文件名的时间戳部分作为变量，使用${...}格式的参数代替。您可以自定义参数名称，示例为filename。

单击右侧的**调度配置**，在**基础属性 > 参数**中为上述自定义参数赋值为 `filename=$[yyyymmddhh24mi]`，详情请参见调度参数概述。



此处自定义变量$[yyyymmddhh24mi]的含义为精确到分的时间戳。例如201911062315（2019年11月6日23点15分）、202005250843（2020年5月25日08点43分）和201912012207（2019年12月1日22点7分）。

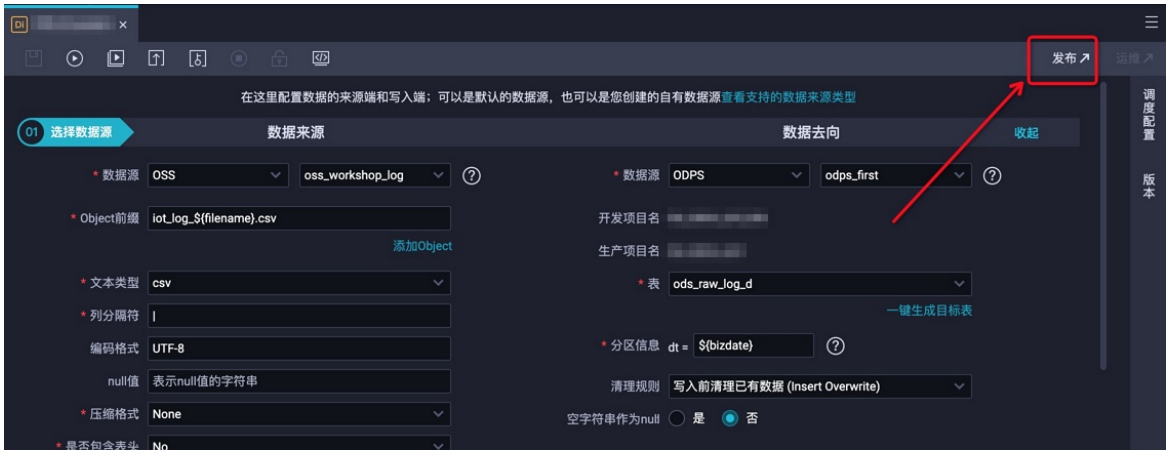
6. 在**调度配置 > 时间属性**下，配置**调度周期**。



选择**调度周期**为**分钟**，根据自身需求设置**开始时间**、**时间间隔**和**结束时间**。

> 注意　时间间隔务必和参数时间戳、任务命名范围保持一致。例如OSS上的文件每15分钟生成一份，则时间间隔同样设置为15分钟。

7. 提交并发布节点，详情请参见发布管理。



8. 任务发布成功后，单击右上角的**运维**，进入**周期任务**或**周期实例**页面，查看生成的任务或实例是否符合需求。详情请参见查看并管理周期任务。

1.2. 通过数据集成导入数据至Elasticsearch

本文为您介绍如何通过数据集成导入离线Elasticsearch数据。

前提条件

1. 准备阿里云账号，并创建账号的访问密钥。详情请参见开通DataWorks。
2. 开通MaxCompute，自动产生一个默认的MaxCompute数据源，并使用主账号登录DataWorks。
3. 创建工作空间，您可以在工作空间中协作完成业务流程，共同维护数据和任务等。详情请参见创建工作空间。

   > **说明** 如果您需要通过子账号创建数据集成任务，请赋予其相应的权限。详情请参见准备RAM用户和角色及成员管理：空间级。

4. 准备好相关的数据源，详情请参见数据源配置。

新建离线同步节点

1. 登录DataWorks控制台。
2. 在左侧导航栏，单击**工作空间列表**。
3. 选择工作空间所在地域后，单击相应工作空间后的**进入数据集成**。
4. 在**数据集成 > 首页**页面，单击**新建同步任务**，进入**数据开发**页面。
5. 在**新建节点**对话框中，输入**节点名称**并选择**目标文件夹**。

   > **说明**
   > - 节点名称的长度不能超过128个字符。
   > - 此处的**目标文件夹**为创建的业务流程，具体操作请参见创建业务流程。

6. 单击**提交**。

配置离线同步节点

1. 成功创建离线同步节点后，单击工具栏中的**转换脚本**。

2. 单击**提示**对话框中的**确认**，即可进入脚本模式进行开发。
3. 参照下文的脚本模板，根据自身需求进行配置。

```
{
"configuration": {
"setting": {
  "speed": {
    "concurrent": "1", //作业并发数。
    "mbps": "1" //作业速率上限。
  }
```

```
},
"reader": {
  "parameter": {
    "connection": [
      {
        "table": [
          "`es_table`" //源端表名。
        ],
        "datasource": "px_mysql_OK" //数据源名，建议和添加的数据源名保持一致。
      }
    ],
    "column": [ //源端表的列名。
      "col_ip",
      "col_double",
      "col_long",
      "col_integer",
      "col_keyword",
      "col_text",
      "col_geo_point",
      "col_date"
    ],
    "where": "", //过滤条件。
  },
  "plugin": "mysql"
},
"writer": {
  "parameter": {
    "cleanup": true, //是否在每次导入数据到Elasticsearch时清空原有数据，全量导入或重建索引时，需要设置为true，同步增量时必须为false。
    "accessKey": "nimda", //如果使用了X-PACK插件，需要填写password；如果未使用，则填空字符串即可。阿里云Elasticsearch使用了X-PACK插件，需要填写password。
    "index": "datax_test", // Elasticsearch的索引名称，如果之前没有，插件会自动创建。
    "alias": "test-1-alias", //数据导入完成后写入别名。
    "settings": {
      "index": {
        "number_of_replicas": 0,
        "number_of_shards": 1
      }
    },
    "batchSize": 1000, //每次批量数据的条数。
    "accessId": "default", //如果使用了X-PACK插件，需要填写username；如果未使用，则填空字符串即可。阿里云Elasticsearch使用了X-PACK插件，需要填写username。
    "endpoint": "http://xxx.xxxx.xxx:xxxx", //Elasticsearch的连接地址，可以在控制台查看。
    "splitter": ",", //如果插入数据是array，则使用指定分隔符。
    "indexType": "default", //Elasticsearch中相应索引下的类型名称。
    "aliasMode": "append", //数据导入完成后增加别名的模式，append（增加模式），exclusive（只留这一个）。
    "column": [ //Elasticsearch中的列名，顺序和Reader中的Column顺序一致。
      {
        "name": "col_ip",//对应于TableStore中的属性列：name。
        "type": "ip"//文本类型，采用默认分词。
      },
      {
        "name": "col_double",
```

```
        "type": "string"
      },
      {
        "name": "col_long",
        "type": "long"
      },
      {
        "name": "col_integer",
        "type": "integer"
      },
      {
        "name": "col_keyword",
        "type": "keyword"
      },
      {
        "name": "col_text",
        "type": "text"
      },
      {
        "name": "col_geo_point",
        "type": "geo_point"
      },
      {
        "name": "col_date",
        "type": "date"
      }
    ],
    "discovery": false//是否自动发现，设置为true。
  },
  "plugin": "elasticsearch"//Writer插件的名称：ElasticsearchWriter，无需修改。
}
},
"type": "job",
"version": "1.0"
}
```

4. 单击图标后，再单击图标。

> **说明**
>
> - Elasticsearch仅支持以脚本模式导入数据。
> - 保存同步任务后，直接单击图标，任务会立刻运行。
>
>   您也可以单击图标，提交同步任务至调度系统中，调度系统会按照配置属性在从第2天开始自动定时执行。

后续步骤

如果您需要使用其它类型的数据源配置同步任务，请参见配置Reader和配置Writer模块的文档。

1.3. 日志服务通过数据集成投递数据

本文将以LogHub数据同步至MaxCompute为例，为您介绍如何通过数据集成功能同步LogHub数据至数据集成已支持的目的端数据源（例如MaxCompute、OSS、OTS、RDBMS和DataHub等）。

前提条件

- 准备好相关的数据源，详情请参见数据源配置。
- 准备需要同步的来源表与目标表。

背景信息

日志服务支持以下数据同步场景：

- 跨地域的LogHub与MaxCompute等数据源的数据同步。
- 不同阿里云账号下的LogHub与MaxCompute等数据源间的数据同步。
- 同一阿里云账号下的LogHub与MaxCompute等数据源间的数据同步。
- 公共云与金融云账号下的LogHub与MaxCompute等数据源间的数据同步。

以B账号进入数据集成配置同步任务，将A账号的LogHub数据同步至B账号的MaxCompute为例，跨阿里云账号的特别说明如下：

1. 使用A账号的AccessKey ID和AccessKey Secret创建LogHub数据源。

   此时B账号可以同步A账号下所有日志服务项目的数据。

2. 使用A账号下的RAM用户A1的AccessKey ID和AccessKey Secret创建LogHub数据源。
   - A账号为RAM用户A1赋予日志服务的通用权限，即 `AliyunLogFullAccess` 和 `AliyunLogReadOnlyAccess` ，详情请参见创建RAM用户及授权。
   - A账号给RAM用户A1赋予日志服务的自定义权限。

     主账号A进入**RAM控制台 > 权限管理 > 权限策略管理**页面，单击**新建授权策略**。

     相关的授权请参见访问控制RAM和RAM子用户访问。

     根据下述策略进行授权后，B账号通过RAM用户A1只能同步日志服务project_name1以及project_name2的数据。

```
{
"Version": "1",
"Statement": [
{
"Action": [
"log:Get*",
"log:List*",
"log:CreateConsumerGroup",
"log:UpdateConsumerGroup",
"log:DeleteConsumerGroup",
"log:ListConsumerGroup",
"log:ConsumerGroupUpdateCheckPoint",
"log:ConsumerGroupHeartBeat",
"log:GetConsumerGroupCheckPoint"
],
"Resource": [
"acs:log:*:*:project/project_name1",
"acs:log:*:*:project/project_name1/*",
"acs:log:*:*:project/project_name2",
"acs:log:*:*:project/project_name2/*"
],
"Effect": "Allow"
}
]
}
```

新建LogHub数据源

1. 登录DataWorks控制台，单击相应工作空间后的**进入数据集成**。
2. 单击左侧导航栏中的**数据源**，即可跳转至**工作空间管理 > 数据源管理**页面。
3. 在**数据源管理**页面，单击右上角的**新增数据源**。
4. 在**新增数据源**对话框中，选择数据源类型为**LogHub**。
5. 填写**新增LogHub数据源**对话框中的配置。

参数	描述
数据源名称	数据源名称必须以字母、数字、下划线组合，且不能以数字和下划线开头。
数据源描述	对数据源进行简单描述，不得超过80个字符。
LogHub Endpoint	LogHub的Endpoint，格式为 `http://example.com` 。详情请参见服务入口。
Project	输入项目名称。
AccessKey ID	访问密钥中的AccessKey ID，您可以进入控制台的**用户信息管理**页面进行复制。
AccessKey Secret	访问密钥中的AccessKey Secret，相当于登录密码。

6. 单击**测试连通性**。

7. 连通性测试通过后，单击**完成**。

新建离线同步节点

1. 在**数据源**页面，单击左上角的图标，选择**全部产品 > DataStudio（数据开发）**。
2. 在**数据开发**页面，鼠标悬停至图标，单击**业务流程**。
3. 在**新建业务流程**对话框中，输入**业务流程名称**和**描述**，单击**新建**。
4. 展开业务流程，右键单击**数据集成**，选择**新建 > 离线同步**。
5. 在**新建节点**对话框中，输入**节点名称**，并选择**目标文件夹**。
6. 单击**提交**，进入离线节点编辑页面。

通过向导模式配置同步任务

1. 在离线节点编辑页面，选择数据来源。

参数	描述
数据源	输入LogHub数据源的名称。
Logstore	导出增量数据的表的名称。该表需要开启Stream，可以在建表时开启，或者使用UpdateTable接口开启。
日志开始时间	数据消费的开始时间位点，为时间范围（左闭右开）的左边界，为yyyyMMddHHmmss格式的时间字符串（例如20180111013000）。该参数可以和DataWorks的调度时间参数配合使用。
日志结束时间	数据消费的结束时间位点，为时间范围（左闭右开）的右边界，为yyyyMMddHHmmss格式的时间字符串（例如20180111013010）。该参数可以和DataWorks的调度时间参数配合使用。
批量条数	一次读取的数据条数，默认为256。

说明 您可以进行数据预览，此处仅选择LogHub中的几条数据展现在预览框。由于您在进行同步任务时，会指定开始时间和结束时间，会导致预览结果和实际的同步结果不一致。

2. 选择MaxCompute数据源及目标表。
3. 选择字段的映射关系。
4. 在**通道控制**中配置作业速率上限和脏数据检查规则。
5. 确认当前节点的配置无误后，单击左上角的**保存**。
6. 运行离线同步节点。

   您可以通过以下两种方式运行离线同步节点：

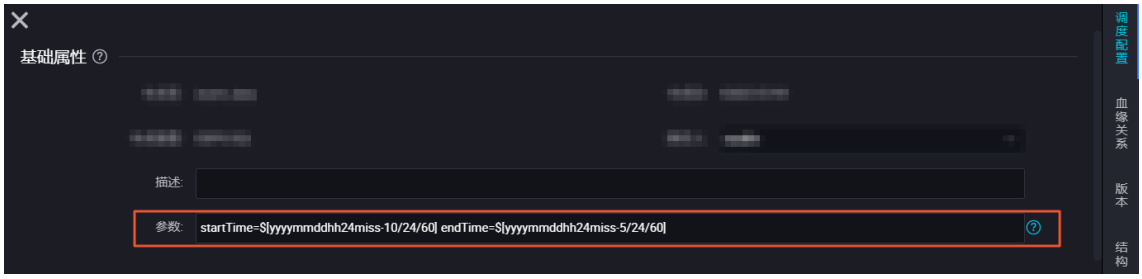
   - 直接运行（一次性运行）

     单击节点编辑页面工具栏中的**运行**图标，直接在页面运行。

     > **说明** 运行之前需要配置自定义参数的具体取值。

   - 调度运行

     单击节点编辑页面工具栏中的**提交**图标，提交离线同步节点至调度系统，调度系统会根据配置的属性，从第2天开始自动定时运行。



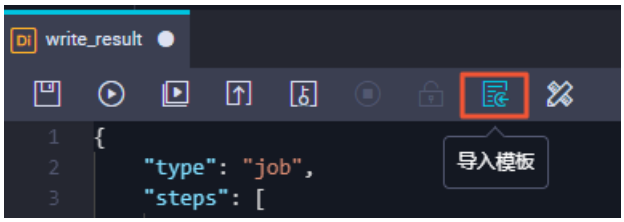
如上图所示，设置开始时间为系统前10分钟，结束时间为系统前5分钟：startTime=$[yyyymmddhh24miss-10/24/60] endTime=$[yyyymmddhh24miss-5/24/60]。

如上图所示，设置离线同步节点的调度周期为分钟，从00:00~23:59每5分钟调度一次。

通过脚本模式配置离线同步节点

1. 成功创建离线同步节点后，单击工具栏中的**转换脚本**。

2. 单击**提示**对话框中的**确认**，即可进入脚本模式进行开发。
3. 单击工具栏中的**导入模板**。



4. 在**导入模板**对话框中，选择从来源端的LogHub数据源同步至目标端的ODPS数据源的导入模板，单击**确认**。
5. 导入模板后，根据自身需求编辑代码，示例脚本如下。

```
{
"type": "job",
"version": "1.0",
"configuration": {
"reader": {
"plugin": "loghub",
"parameter": {
"datasource": "loghub_lzz",//数据源名，需要和您添加的数据源名一致。
"logstore": "logstore-ut2",//目标日志库的名字，LogStore是日志服务中日志数据的采集、存储和查询单元。
"beginDateTime": "${startTime}",//数据消费的开始时间位点，为时间范围（左闭右开）的左边界。
"endDateTime": "${endTime}",//数据消费的结束时间位点，为时间范围（左闭右开）的右边界。
"batchSize": 256,//一次读取的数据条数，默认为256。
"splitPk": "",
"column": [
"key1",
"key2",
"key3"
]
}
},
"writer": {
"plugin": "odps",
"parameter": {
"datasource": "odps_first",//数据源名，需要和您添加的数据源名一致。
"table": "ok",//目标表名。
"truncate": true,
"partition": "",//分区信息。
"column": [//目标列名。
"key1",
"key2",
"key3"
]
}
},
"setting": {
"speed": {
"mbps": 8,//作业速率上限。
"concurrent": 7//并发数。
}
}
}
}
```

1.4. DataHub通过数据集成批量导入数据

本文以Stream同步数据至DataHub的脚本模式为例，为您介绍如何通过数据集成导入离线DataHub数据。

前提条件

1. 准备阿里云账号，并创建账号的访问密钥。详情请参见开通DataWorks。
2. 开通MaxCompute，自动产生一个默认的MaxCompute数据源，并使用主账号登录DataWorks。

3. 创建工作空间，您可以在工作空间中协作完成业务流程，共同维护数据和任务等。详情请参见创建工作空间。

> **说明** 如果您需要通过子账号创建数据集成任务，请赋予其相应的权限。详情请参见准备RAM用户和角色及成员管理：空间级。

背景信息

数据集成是阿里云提供的数据同步平台。该平台具备可跨异构数据存储系统、可靠、安全、低成本、可弹性扩展等特点，可以为20多种数据源提供不同网络环境下的离线数据进出通道。数据源类型的详情请参见支持的数据源与读写插件。

本文以配置DataHub数据源为例，如果您需要使用其它类型的数据源配置同步任务，请参见配置Reader和配置Writer。

操作步骤

1. 进入**数据集成**页面。
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**工作空间列表**。
   iii. 选择工作空间所在地域后，单击相应工作空间后的**进入数据集成**。
2. 在**数据集成 > 首页**页面，单击**新建同步任务**，进入**数据开发**页面。
3. 在**新建节点**对话框中，输入**节点名称**并选择**目标文件夹**，单击**提交**。

> **说明**
> - 节点名称的长度不能超过128个字符。
> - 此处的**目标文件夹**为创建的业务流程，具体操作请参见创建业务流程。

4. 成功创建离线同步节点后，单击工具栏中的图标。
5. 单击**提示**对话框中的**确认**，即可进入脚本模式进行开发。
6. 单击工具栏中的图标。
7. 在**导入模板**对话框中，选择从来源端的Stream数据源同步至目标端的DataHub数据源的导入模板，单击**确认**。
8. 导入模板后，根据自身需求编辑代码。

```
{
"type": "job",
"version": "1.0",
"configuration": {
 "setting": {
   "errorLimit": {
     "record": "0"
   },
   "speed": {
     "mbps": "1",
     "concurrent": 1,//作业并发数。
     "throttle": false
   }
 },
 "reader": {
   "plugin": "stream",
   "parameter": {
     "column": [//源端列名。
       {
         "value": "field",//列属性。
         "type": "string"
       },
       {
         "value": true,
         "type": "bool"
       },
       {
         "value": "byte string",
         "type": "bytes"
       }
     ],
     "sliceRecordCount": "100000"
   }
 },
 "writer": {
   "plugin": "datahub",
   "parameter": {
     "datasource": "datahub",//数据源名。
     "topic": "xxxx",//Topic是DataHub订阅和发布的最小单位，您可以用Topic来表示一类或者一种流数
据。
     "mode": "random",//随机写入。
     "shardId": "0",//Shard 表示对一个Topic进行数据传输的并发通道，每个Shard会有对应的ID。
     "maxCommitSize": 524288,//为了提高写出效率，待攒数据大小达到maxCommitSize大小（单位MB）
时，批量提交到目的端。默认是1,048,576，即1MB数据。
     "maxRetryCount": 500
   }
 }
}
}
```

9. 配置完成后，分别单击和图标。

> 说明
>
> - DataHub仅支持以脚本模式导入数据。
> - 如果需要选择新模板，请单击工具栏中的图标。一旦导入新模板，原有内容会被覆盖。
> - 保存同步任务后，直接单击图标，任务会立刻运行。
>
>   您也可以单击图标，提交同步任务至调度系统中。调度系统会按照配置属性，从第2天开始定时执行。

1.5. MaxCompute跨项目迁移

本文为您介绍如何配置相同区域下不同的MaxCompute项目，以及如何实现数据迁移。

前提条件

请您首先完成教程《搭建互联网在线运营分析平台》的全部步骤，详情请参见业务场景与开发流程。

背景信息

本文使用的被迁移的原始项目为教程《搭建互联网在线运营分析平台》中的bigdata_DOC项目，您需要再创建一个迁移目标项目，用于存放原始项目的表、资源、配置和数据。

操作步骤

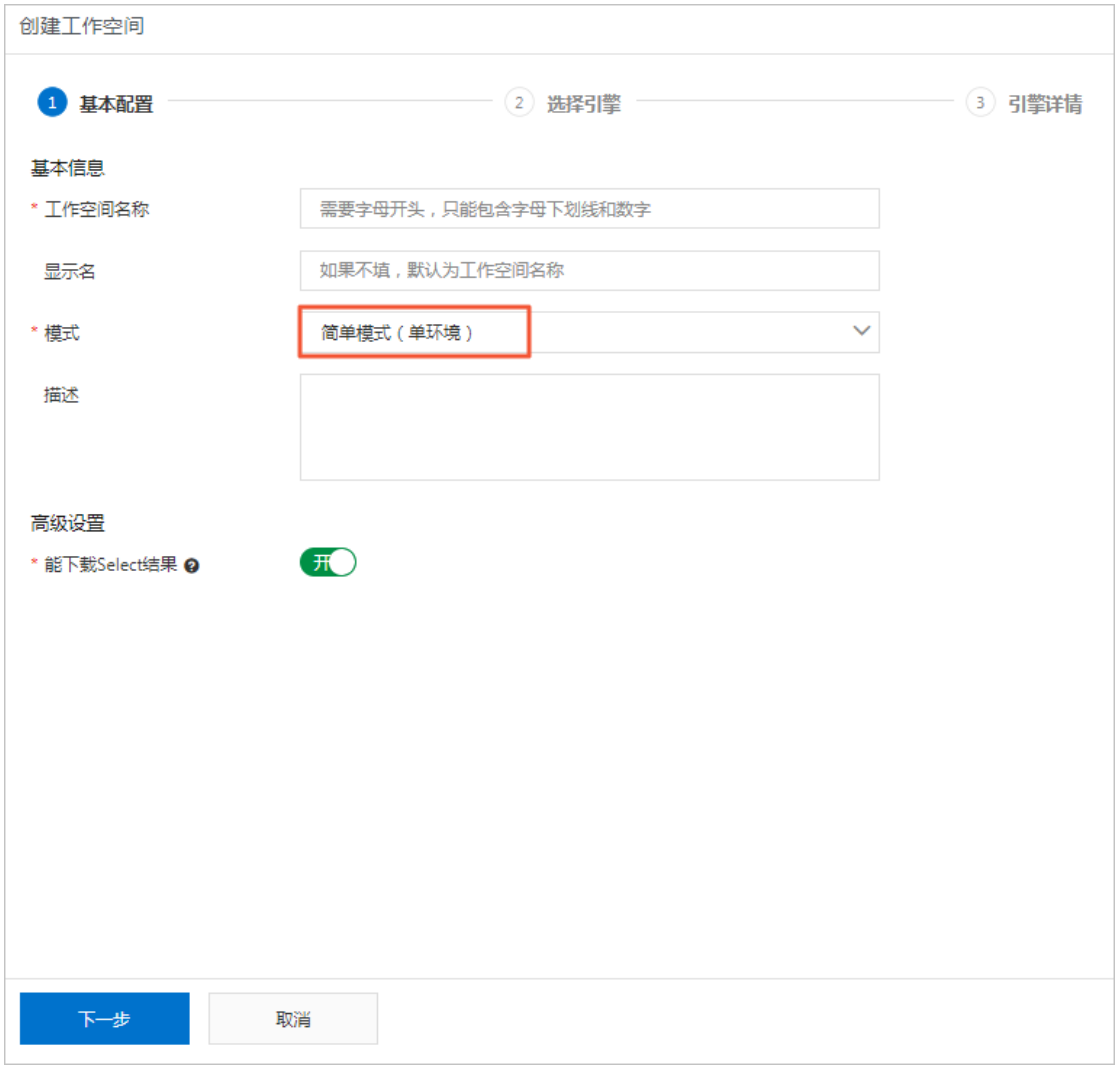
1. 创建迁移目标项目

   本文的MaxCompute项目即DataWorks的工作空间。

   i. 登录DataWorks控制台，单击左侧导航栏中的**工作空间列表**。

   ii. 选择区域为**华东1（杭州）**，单击**创建工作空间**。

   iii. 填写**创建工作空间**对话框中的**基本配置**，单击**下一步**。



分类	参数	描述
	工作空间名称	工作空间名称的长度需要在3~23个字符，以字母开头，且只能包含字母、下划线（_）和数字。
	显示名	显示名不能超过23个字符，只能字母、中文开头，仅包含中文、字母、下划线（_）和数字。

基本信息 分类	参数	描述
	模式	工作空间模式是DataWorks新版推出的新功能，分为**简单模式**和**标准模式**： ■ **简单模式**：指一个DataWorks工作空间对应一个MaxCompute项目，无法设置开发和生产环境，只能进行简单的数据开发，无法对数据开发流程以及表权限进行强控制。 ■ **标准模式**：指一个DataWorks工作空间对应两个MaxCompute项目，可以设置开发和生产两种环境，提升代码开发规范，并能够对表权限进行严格控制，禁止随意操作生产环境的表，保证生产表的数据安全。 详情请参见简单模式和标准模式的区别。
	描述	对创建的工作空间进行简单描述。
高级设置	**能下载select结果**	控制数据开发中查询的数据结果是否能够下载，如果关闭无法下载select的数据查询结果。此参数在工作空间创建完成后可以在工作空间配置页面进行修改，详情可参考文档：安全设置。

由于原始项目bigdata_DOC为简单模式，为方便起见，本文中DataWorks工作空间模式也为**简单模式（单环境）**。

工作空间名称全局唯一，建议您使用易于区分的名称，本例中使用的名称为clone_test_doc。

iv. 选择**计算引擎服务**为MaxCompute、按量付费，单击**下一步**。

v. 配置**引擎详情**，单击**创建工作空间**。

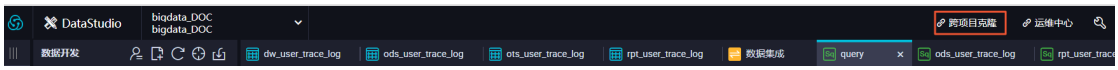
分类	参数	描述
MaxCompute	**实例显示名称**	实例显示名称不能超过27个字符，仅支持字母、中文开头，仅包含中文、字母、下划线和数字。
	MaxCompute项目名称	默认与DataWorks工作空间的名称一致。
	MaxCompute访问身份	开发环境的MaxCompute访问身份默认为**任务负责人**，不可以修改。 生产环境的MaxCompute访问身份包括**阿里云主账号**和**阿里云子账号**。
	Quota组切换	Quota用来实现计算资源和磁盘配额。

2. 跨项目克隆

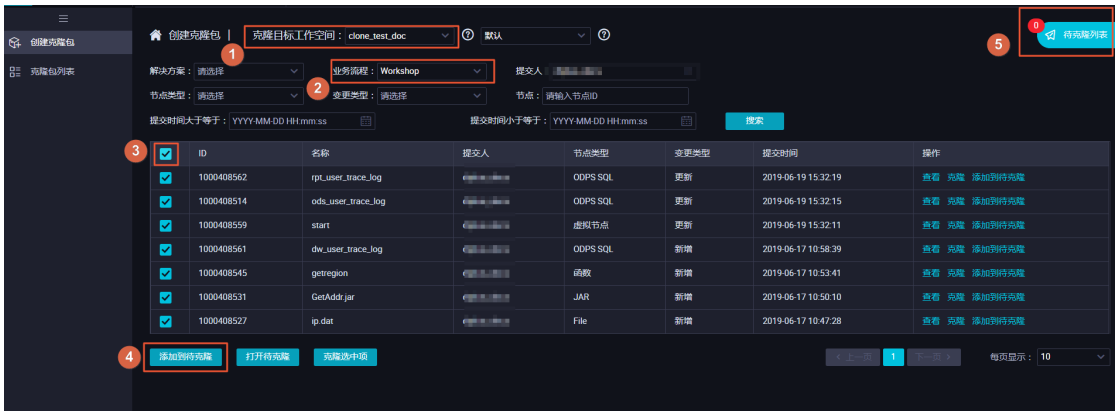
您可以通过**跨项目克隆**功能将原始项目bigdata_DOC的节点配置和资源复制到当前项目，详情请参见跨项目克隆实践。

> 说明
> - 跨项目克隆无法复制表结构与数据。
> - 跨项目克隆无法复制组合节点，需要您手动创建。

i. 单击原始项目bigdata_DOC右上角的**跨项目克隆**，跳转至相应的克隆页面。



ii. 选择**克隆目标工作空间**为clone_test_doc，**业务流程**为您需要克隆的业务流程Workshop，勾选所有节点，单击**添加到待克隆**后单击右侧的**待克隆列表**。



iii. 单击**全部克隆**，将选中的节点克隆至工作空间clone_test_DOC。

iv. 切换至您新建的项目，检查节点是否已完成克隆。

3. 新建数据表

跨项目克隆功能无法克隆您的表结构，因此您需要手动新建表。

- 对于非分区表，建议使用如下语句迁移表结构。

```
create table table_name as select * from 源库MaxCompute项目.表名 ;
```

- 对于分区表，建议使用如下语句迁移表结构。

```
create table table_name partitioned by (分区列 string);
```

新建表后请将表提交到生产环境。更多建表信息，请参见新建数据表。

4. 数据同步

跨项目克隆功能无法复制原始项目的数据到新项目，因此您需要手动同步数据，本文中仅同步表rpt_user_trace_log的数据。

i. 新建数据源。

a. 在**数据集成**页面，单击左侧导航栏上的**数据源**。

b. 在**数据源管理**页面，单击右上角**新增数据源**，并选择MaxCompute(ODPS)。

c. 填写您的**数据源名称**、ODPS项目名称、AccessKey ID、AccessKey Secret等信息，单击**完成**，详情请参见配置MaxCompute数据源。

ii. 创建数据同步任务。

a. 在**数据开发**页面右键单击您克隆的业务流程Workshop下的**数据集成**，选择**新建 > 离线同步**。

b. 编辑您新建的数据同步任务节点，填写参数如下图所示。其中数据源bigdata_DOC是您的原始项目，数据源odps_first代表您当前的新建项目，表名是您需要同步数据的表rpt_user_trace_log。完成后单击**调度配置**。

c. 单击**使用工作空间根节点**后，提交数据同步任务。

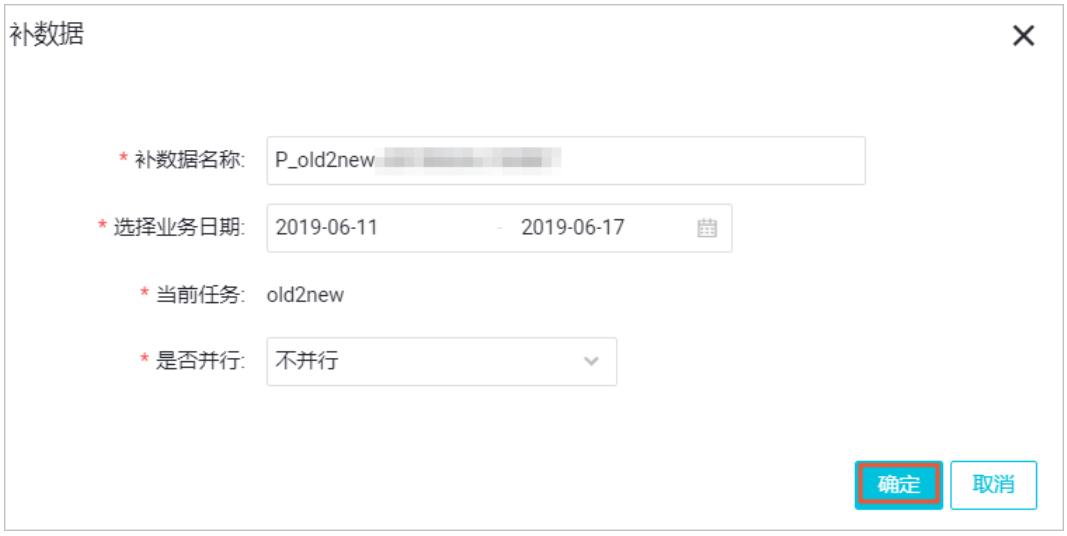
iii. 补数据

a. 单击左上角的图标，选择**全部产品 > 运维中心**。

b. 单击左侧导航栏中的**周期任务运维 > 周期任务**。

c. 右键单击您的数据同步任务，选择**补数据 > 当前节点**。

d. 本例中，需要补数据的日期分区为2019年6月11日到17日，您可以直接选择业务日期，进行多个分区的数据同步。完成设置后，单击**确定**。



e. 在**周期任务运维 > 补数据实例**页面，您可以查看补数据实例任务运行状态，显示**运行成功**则说明完成数据同步。

iv. 验证结果

您可以在**业务流程 > 数据开发**中新建**ODPS SQL**类型节点，执行如下语句查看数据是否完成同步。

```
select * from rpt_user_trace_log where dt BETWEEN '20190611' and '20190617';
```

1.6. Hadoop数据迁移MaxCompute最佳实践

本文为您介绍如何通过DataWorks数据同步功能，迁移HDFS数据至MaxCompute，或从MaxCompute迁移数据至HDFS。无论您使用Hadoop还是Spark，均可以与MaxCompute进行双向同步。

前提条件

- 开通MaxCompute并创建项目

  本文以在华东1（杭州）区域创建项目bigdata_DOC为例。详情请参见开通MaxCompute和DataWorks。

- 搭建Hadoop集群

  进行数据迁移前，您需要保证Hadoop集群环境正常。本文使用阿里云EMR服务自动化搭建Hadoop集群，详情请参见创建集群。

  本文使用的EMR Hadoop版本信息如下：

  - EMR版本：EMR-3.11.0
  - 集群类型：HADOOP

- 软件信息：HDFS2.7.2 / YARN2.7.2 / Hive2.3.3 / Ganglia3.7.2 / Spark2.2.1 / HUE4.1.0 / Zeppelin0.7.3 / Tez0.9.1 / Sqoop1.4.6 / Pig0.14.0 / ApacheDS2.0.0 / Knox0.13.0

Hadoop集群使用经典网络，区域为华东1（杭州），主实例组ECS计算资源配置公网及内网IP，高可用选择为否（非HA模式）。

操作步骤

1. 数据准备
   i. Hadoop集群创建测试数据
      a. 进入EMR控制台界面，在项目下新建作业doc。本例中HIVE建表语句如下。 关于EMR上新建作业更多信息请参见作业编辑。

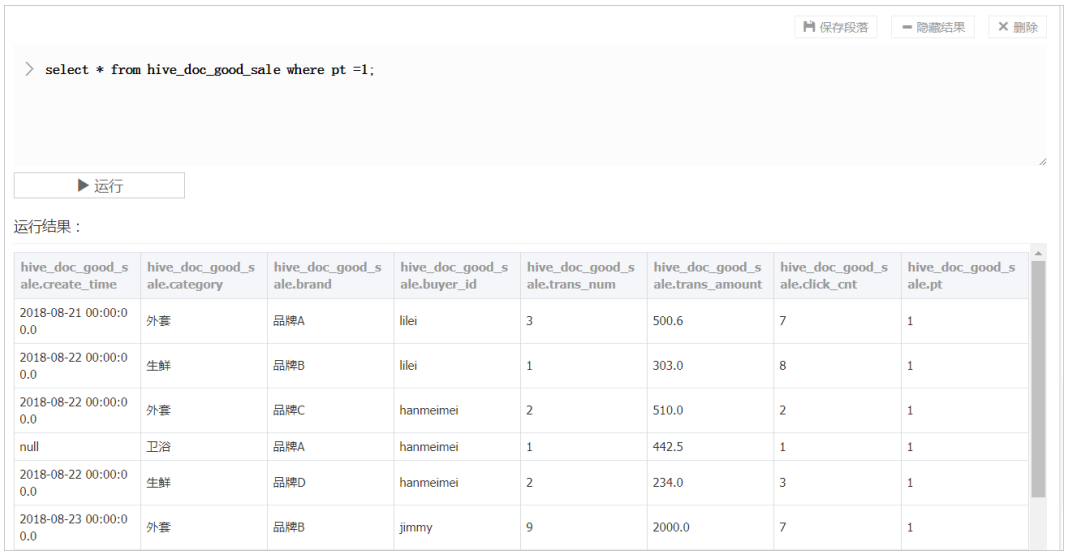
```
CREATE TABLE IF NOT
EXISTS hive_doc_good_sale(
   create_time timestamp,
   category STRING,
   brand STRING,
   buyer_id STRING,
   trans_num BIGINT,
   trans_amount DOUBLE,
   click_cnt BIGINT
   )
   PARTITIONED BY (pt string) ROW FORMAT
DELIMITED FIELDS TERMINATED BY ',' lines terminated by '\n';
```

      b. 单击**运行**，出现 `Query executed successfully` 提示，则说明成功在EMR Hadoop集群上创建了表hive_doc_good_sale。

      c. 插入测试数据。您可以选择从OSS或其他数据源导入测试数据，也可以手动插入少量的测试数据。本文中手动插入数据如下。

```
insert into
hive_doc_good_sale PARTITION(pt =1 ) values('2018-08-21','外套','品牌A','lilei',3,500.6,7),('2018-08-22','生鲜','品牌B','lilei',1,303,8),('2018-08-22','外套','品牌C','hanmeimei',2,510,2),(2018-08-22,'卫浴','品牌A','hanmeimei',1,442.5,1),('2018-08-22','生鲜','品牌D','hanmeimei',2,234,3),('2018-08-23','外套','品牌B','jimmy',9,2000,7),('2018-08-23','生鲜','品牌A','jimmy',5,45.1,5),('2018-08-23','外套','品牌E','jimmy',5,100.2,4),('2018-08-24','生鲜','品牌G','peiqi',10,5560,7),('2018-08-24','卫浴','品牌F','peiqi',1,445.6,2),('2018-08-24','外套','品牌A','ray',3,777,3),('2018-08-24','卫浴','品牌G','ray',3,122,3),('2018-08-24','外套','品牌C','ray',1,62,7) ;
```

d. 完成插入数据后，您可以执行 `select * from hive_doc_good_sale where pt =1;` 语句，检查Hadoop集群表中是否已存在数据可以用于迁移。



hive_doc_good_sale.create_time	hive_doc_good_sale.category	hive_doc_good_sale.brand	hive_doc_good_sale.buyer_id	hive_doc_good_sale.trans_num	hive_doc_good_sale.trans_amount	hive_doc_good_sale.click_cnt	hive_doc_good_sale.pt
2018-08-21 00:00:00.0	外套	品牌A	lilei	3	500.6	7	1
2018-08-22 00:00:00.0	生鲜	品牌B	lilei	1	303.0	8	1
2018-08-22 00:00:00.0	外套	品牌C	hanmeimei	2	510.0	2	1
null	卫浴	品牌A	hanmeimei	1	442.5	1	1
2018-08-22 00:00:00.0	生鲜	品牌D	hanmeimei	2	234.0	3	1
2018-08-23 00:00:00.0	外套	品牌B	jimmy	9	2000.0	7	1

ii. 利用DataWorks新建目标表

a. 登录DataWorks控制台，单击相应工作空间后的**进入数据开发**。

b. 进入**DataStudio（数据开发）**页面，右键单击工作流程，选择**新建 > MaxCompute > 表**。

c. 在**新建表**对话框中，填写**表名**，并单击**提交**。

d. 进入**新建表**页面，选择**DDL模式**。

e. 在**DDL模式**对话框中输入建表语句，单击**生成表结构**，并确认操作。本示例的建表语句如下所示。

```
CREATE TABLE IF NOT EXISTS hive_doc_good_sale(
   create_time string,
   category STRING,
   brand STRING,
   buyer_id STRING,
   trans_num BIGINT,
   trans_amount DOUBLE,
   click_cnt BIGINT
   )
   PARTITIONED BY (pt string) ;
```

在建表过程中，需要考虑HIVE数据类型与MaxCompute数据类型的映射，当前数据映射关系请参见数据类型映射表。

由于本文使用DataWorks进行数据迁移，而DataWorks数据同步功能暂不支持TIMESTAMP类型数据。因此在DataWorks建表语句中，将create_time设置为STRING类型。

上述步骤同样可通过odpscmd命令行工具完成，命令行工具安装和配置请参见使用客户端（odpscmd）连接。

```
odps@ bigdata_DOC>CREATE TABLE IF NOT EXISTS hive_doc_good_sale(
   create_time timestamp,
   category STRING,
   brand STRING,
   buyer_id STRING,
   trans_num BIGINT,
   trans_amount DOUBLE,
   click_cnt BIGINT
   )
   PARTITIONED BY (pt string) ;
                >               >               >               >
        >               >               >               >
 >
ID = 20180906110540873gev1bpim
OK
odps@ bigdata_DOC>drop table hive_doc_good_sale;
Confirm to "drop table hive_doc_good_sale;" (yes/no)? yes

ID = 20180906110825180gxh66292
```

> **说明** 考虑到部分HIVE与MaxCompute数据类型的兼容问题，建议在odpscmd客户端上执行以下命令。
>
> ```
> set odps.sql.type.system.odps2=true;
> set odps.sql.hive.compatible=true;
> ```

f. 单击**提交到生产环境**，完成表的创建。

g. 完成建表后，单击左侧导航栏中的**表管理**，即可查看当前创建的MaxCompute表。

2. 数据同步

i. 新建自定义资源组

由于MaxCompute项目所处的网络环境与Hadoop集群中的数据节点（data node）网络通常不可达，您可以通过自定义资源组的方式，将DataWorks的同步任务运行在Hadoop集群的Master节点上（Hadoop集群内Master节点和数据节点通常可达）。

a. 查看Hadoop集群数据节点

a. 登录EMR控制台，单击**集群管理**。

b. 选择集群名称，并在左侧导航栏上单击**主机列表**。

您也可以通过单击上图中Master节点的ECS ID，进入ECS实例详情页。然后单击远程连接进入ECS，执行 `hadoop dfsadmin -report` 命令查看data node。

```
DFS Used%: 0.05%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0

-------------------------------------------------
Live datanodes (2):

Name: 10.31.122.189:50010 (emr-worker-1.cluster-74503)
Hostname: emr-worker-1.cluster-74503
Decommission Status : Normal
Configured Capacity: 333373341696 (310.48 GB)
DFS Used: 155725824 (148.51 MB)
Non DFS Used: 325541888 (310.46 MB)
DFS Remaining: 332892073984 (310.03 GB)
DFS Used%: 0.05%
DFS Remaining%: 99.86%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Thu Sep 06 19:41:01 CST 2018


Name: 10.81.78.209:50010 (emr-worker-2.cluster-74503)
Hostname: emr-worker-2.cluster-74503
Decommission Status : Normal
Configured Capacity: 333373341696 (310.48 GB)
DFS Used: 155725824 (148.51 MB)
Non DFS Used: 325451776 (310.38 MB)
DFS Remaining: 332892164096 (310.03 GB)
DFS Used%: 0.05%
DFS Remaining%: 99.86%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Thu Sep 06 19:41:02 CST 2018
```

本示例的data node只具有内网地址，很难与DataWorks默认资源组互通，所以需要设置自定义资源组，将master node设置为执行DataWorks数据同步任务的节点。

b. 新建任务资源组

a. 在DataWorks控制台上，进入**数据集成 > 自定义资源组管理**页面，单击右上角的**新增自定义资源组**。

> **说明** 目前仅专业版及以上版本方可使用此入口。

b. 添加服务器时，需要输入ECS UUID和机器IP等信息（对于经典网络类型，需要输入服务器名称。对于专有网络类型，需要输入服务器UUID）。目前仅DataWorks V2.0华东2（上海）支持添加经典网络类型的调度资源，对于其他区域，无论您使用的是经典网络还是专有网络类型，在添加调度资源组时都请选择专有网络类型。

机器IP需要填写master node公网IP（内网IP有可能不可达）。ECS的UUID需要进入master node管理终端，通过命令dmidecode | grep UUID获取（如果您的hadoop集群并非搭建在EMR环境上，也可以通过该命令获取）。

c. 添加服务器后，需要保证master node与DataWorks网络可达。如果您使用的是ECS服务器，需要设置服务器安全组。

- 如果您使用的内网IP互通，请参见场景示例：ECS自建数据库的安全组配置。
- 如果您使用的是公网IP，可以直接设置安全组公网出入方向规则。本文中设置公网入方向放通所有端口（实际应用场景中，为了您的数据安全，强烈建议设置详细的放通规则）。

d. 完成上述步骤后，按照提示安装自定义资源组agent。当前状态显示为**可用**时，则新增自定义资源组成功。

如果状态为**不可用**，您可以登录master node，执行 `tail -f/home/admin/alisatasknode/logs/heartbeat.log` 命令查看DataWorks与master node之间心跳报文是否超时。

ii. 新建数据源

DataWorks新建工作空间后，默认数据源odps_first。因此只需要添加Hadoop集群数据源。更多详情请参见配置HDFS数据源。

a. 进入**数据集成**页面，单击左侧导航栏中的**数据源**。

b. 在**数据源管理**页面，单击右上角的**新增数据源** 。

c. 在**新增数据源**页面中，选择数据源类型为**HDFS**。

d. 填写HDFS数据源的各配置项。

配置	说明
数据源名称	数据源名称必须以字母、数字、下划线组合，且不能以数字和下划线开头。
数据源描述	对数据源进行简单描述，不得超过80个字符。
适用环境	可以选择**开发**或**生产**环境。 **说明** 仅标准模式工作空间会显示此配置。
DefaultFS	对于EMR Hadoop集群而言，如果Hadoop集群为HA集群，则此处地址为 `hdfs://emr-header-1的IP:8020` 。如果Hadoop集群为非HA集群，则此处地址为 `hdfs://emr-header-1的IP:9000` 。 本实验中的emr-header-1与DataWorks通过公网连接，因此此处填写公网IP并放通安全组。

e. 完成配置后，单击**测试连通性**。

f. 测试连通性通过后，单击**完成**。

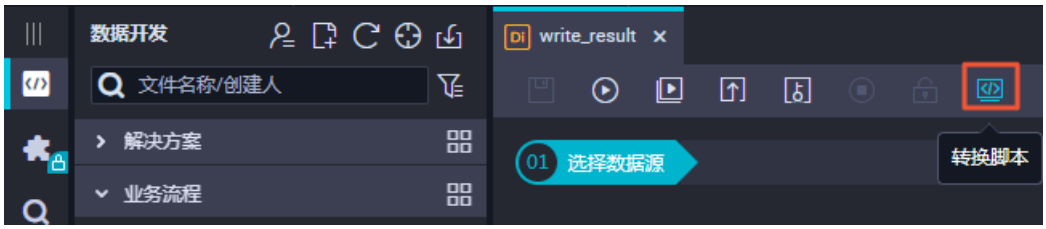
> **说明** 如果EMR Hadoop集群设置网络类型为专有网络，则不支持连通性测试。

iii. 配置数据同步任务

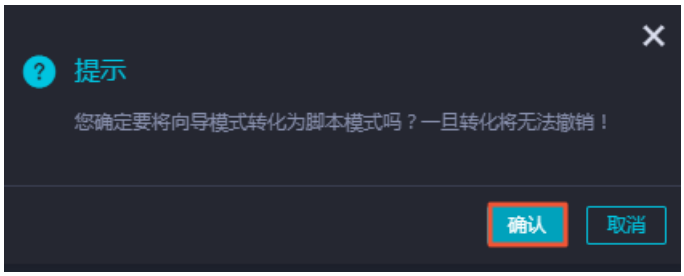
a. 进入**数据开发**页面，在左侧菜单栏顶部选择**新建 > 数据集成 > 离线同步**。

b. 在**新建节点**对话框中，输入**节点名称**，单击**提交**。

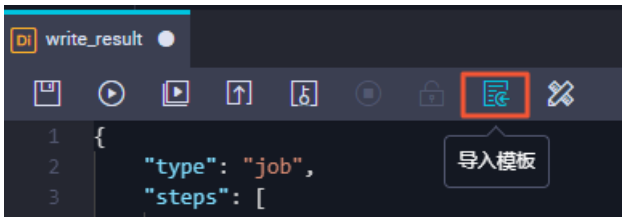
c. 成功创建数据同步节点后，单击工具栏中的**转换脚本**按钮。



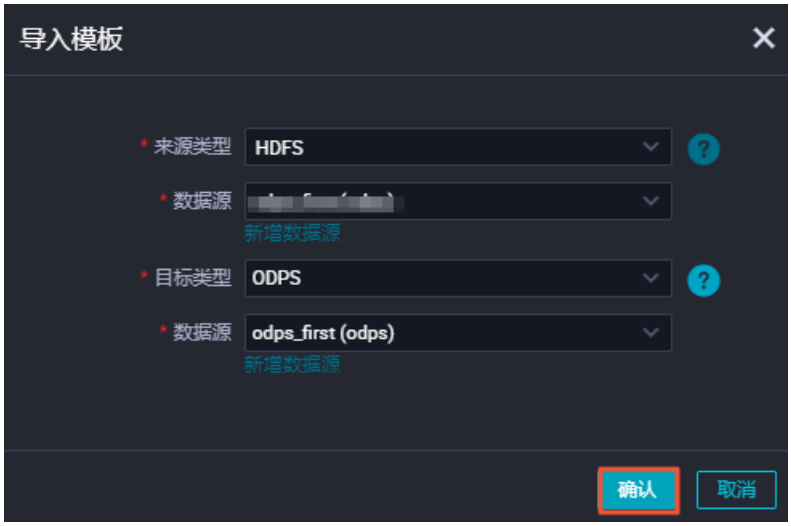
d. 单击**提示**对话框中的**确认**，即可进入脚本模式进行开发。



e. 单击工具栏中的**导入模板**按钮。



f. 在**导入模板**对话框中，选择**来源类型**、**数据源**、**目标类型**及**数据源**，单击**确认**。



g. 新建同步任务完成后，通过导入模板已生成了基本的读取端配置。此时您可以继续手动配置数据同步任务的读取端数据源，以及需要同步的表信息等。本示例的代码如下所示，更多详情请参见HDFS Reader。

```
{
  "configuration": {
    "reader": {
      "plugin": "hdfs",
      "parameter": {
        "path": "/user/hive/warehouse/hive_doc_good_sale/",
```

```
        "datasource": "HDFS1",
        "column": [
          {
            "index": 0,
            "type": "string"
          },
          {
            "index": 1,
            "type": "string"
          },
          {
            "index": 2,
            "type": "string"
          },
          {
            "index": 3,
            "type": "string"
          },
          {
            "index": 4,
            "type": "long"
          },
          {
            "index": 5,
            "type": "double"
          },
          {
            "index": 6,
            "type": "long"
          }
        ],
        "defaultFS": "hdfs://121.199.11.138:9000",
        "fieldDelimiter": ",",
        "encoding": "UTF-8",
        "fileType": "text"
      }
    },
    "writer": {
      "plugin": "odps",
      "parameter": {
        "partition": "pt=1",
        "truncate": false,
        "datasource": "odps_first",
        "column": [
          "create_time",
          "category",
          "brand",
          "buyer_id",
          "trans_num",
          "trans_amount",
          "click_cnt"
        ],
        "table": "hive_doc_good_sale"
      }
```

```
    },
    "setting": {
      "errorLimit": {
        "record": "1000"
      },
      "speed": {
        "throttle": false,
        "concurrent": 1,
        "mbps": "1",
      }
    }
  },
  "type": "job",
  "version": "1.0"
}
```

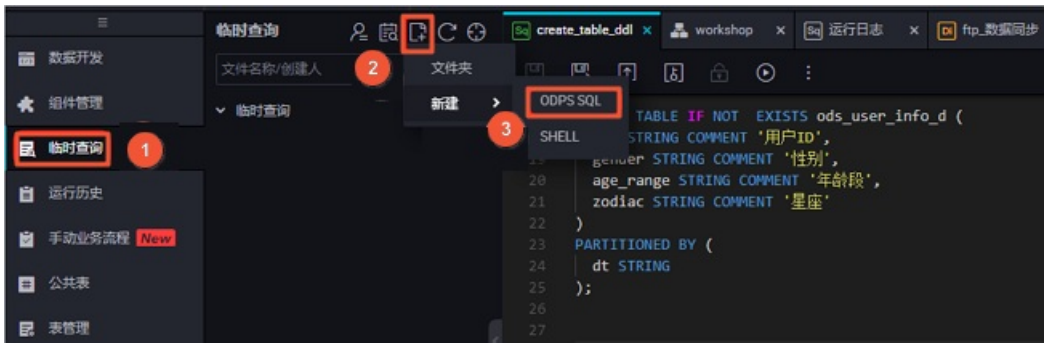
其中，path参数为数据在Hadoop集群中存放的位置。您可以在登录Master Node后，执行 `hdfs dfs -ls /user/hive/warehouse/hive_doc_good_sale` 命令确认。对于分区表，您可以不指定分区，DataWorks数据同步会自动递归到分区路径。

```
[root@emr-header-1 logs]# hdfs dfs -ls /user/hive/warehouse/hive_doc_good_sale/
Found 1 items
drwxr-x--x   - hive hadoop          0 2018-09-03 17:46 /user/hive/warehouse/hive_doc_good_sale/pt=1
```

h. 完成配置后，单击**运行**。如果提示任务运行成功，则说明同步任务已完成。如果运行失败，可以通过日志进行排查。

执行结果

1. 单击左侧导航栏中的**临时查询**。
2. 选择**新建 > ODPS SQL**。



3. 编写并执行SQL语句，查看导入hive_doc_good_sale的数据。SQL语句如下所示：

```
--查看是否成功写入MaxCompute。
select * from hive_doc_good_sale where pt=1;
```

您也可以在odpscmd命令行工具中输入 `select * FROM hive_doc_good_sale where pt =1;` ，查询表结果。

如果您想实现MaxCompute数据迁移至Hadoop，步骤与上述步骤类似，不同的是同步脚本内的reader和writer对象需要对调，具体实现脚本如下。

```
{
  "configuration": {
    "reader": {
      "plugin": "odps",
```

```
      "parameter": {
      "partition": "pt=1",
      "isCompress": false,
      "datasource": "odps_first",
      "column": [
        "create_time",
        "category",
        "brand",
      "buyer_id",
      "trans_num",
      "trans_amount",
      "click_cnt"
    ],
    "table": "hive_doc_good_sale"
    }
  },
  "writer": {
    "plugin": "hdfs",
    "parameter": {
    "path": "/user/hive/warehouse/hive_doc_good_sale",
    "fileName": "pt=1",
    "datasource": "HDFS_data_source",
    "column": [
      {
        "name": "create_time",
        "type": "string"
      },
      {
        "name": "category",
        "type": "string"
      },
      {
        "name": "brand",
        "type": "string"
      },
      {
        "name": "buyer_id",
        "type": "string"
      },
      {
        "name": "trans_num",
        "type": "BIGINT"
      },
      {
        "name": "trans_amount",
        "type": "DOUBLE"
      },
      {
        "name": "click_cnt",
        "type": "BIGINT"
      }
    ],
    "defaultFS": "hdfs://47.99.162.100:9000",
    "writeMode": "append",
    "fieldDelimiter": ","
```

```
      "fieldDelimiter": ",",
      "encoding": "UTF-8",
      "fileType": "text"
      }
    },
    "setting": {
      "errorLimit": {
        "record": "1000"
    },
    "speed": {
      "throttle": false,
      "concurrent": 1,
      "mbps": "1",
    }
    }
},
"type": "job",
"version": "1.0"
}
```

您需要参见HDFS Writer，在运行上述同步任务前，对Hadoop集群进行设置。在运行同步任务后，手动复制同步过去的文件。

1.7. 迁移ECS自建MySQL数据库至MaxCompute

本文为您介绍如何使用独享数据集成资源，将您在ECS上自建的MySQL数据库中的数据，迁移到MaxCompute。

前提条件

- 已拥有至少一个绑定专有网络VPC的ECS（请勿使用经典网络），并在ECS上安装好MySQL数据库，数据库中已创建好用户和测试数据。本文中ECS自建MySQL的测试数据创建语句如下。

```
CREATE TABLE IF NOT EXISTS good_sale(
   create_time timestamp,
   category varchar(20),
   brand varchar(20),
   buyer_id varchar(20),
   trans_num varchar(20),
   trans_amount DOUBLE,
   click_cnt varchar(20)
   );
insert into good_sale values('2018-08-21','coat','brandA','lilei',3,500.6,7),
('2018-08-22','food','brandB','lilei',1,303,8),
('2018-08-22','coat','brandC','hanmeimei',2,510,2),
('2018-08-22','bath','brandA','hanmeimei',1,442.5,1),
('2018-08-22','food','brandD','hanmeimei',2,234,3),
('2018-08-23','coat','brandB','jimmy',9,2000,7),
('2018-08-23','food','brandA','jimmy',5,45.1,5),
('2018-08-23','coat','brandE','jimmy',5,100.2,4),
('2018-08-24','food','brandG','peiqi',10,5560,7),
('2018-08-24','bath','brandF','peiqi',1,445.6,2),
('2018-08-24','coat','brandA','ray',3,777,3),
('2018-08-24','bath','brandG','ray',3,122,3),
('2018-08-24','coat','brandC','ray',1,62,7) ;
```

- 请记录好您的ECS的**私有IP**、**专有网络**和**虚拟交换机**信息。

- ECS上的安全组已放通MySQL数据库所使用的端口（默认为3306），详情请参见添加安全组规则，请记录好您的安全组名称。

- 已成功创建DataWorks工作空间。本文使用DataWorks简单模式工作空间，计算引擎为MaxCompute。请保证您的ECS与DataWorks工作空间处于同一个地域，创建方法请参见创建工作空间。
- 已完成独享数据集成资源的购买，并且绑定了ECS所在的专有网络VPC。请注意独享资源组必须与ECS同一可用区，详情请参见新增和使用独享数据集成资源组。完成绑定后，您可以在**资源组列表**查看到您的独享资源组。

- 在**专有网络绑定**处查看**专有网络**、**交换机**和**安全组**信息是否和ECS一致。

背景信息

独享资源可以保障您的数据快速、稳定地传输。您购买的独享数据集成资源和需要访问的数据源（即本文中的ECS自建MySQL数据库）必须在同地域同可用区，且和DataWorks工作空间同地域。

操作步骤

1. 在DataWorks上创建MySQL数据源。
   i. 使用主账号登录DataWorks控制台。
   ii. 在**工作空间列表**单击**进入数据集成**。

   iii. 在左侧导航栏，单击**数据源**。
   iv. 单击**数据源管理**页面右上角的**新增数据源**。
   v. 在**新增数据源**页面，单击**MySQL**。
   vi. 在**新增MySQL数据源**对话框中，配置各项参数，详情请参见配置MySQL数据源。

   本文以**连接串模式**为例，在**JDBC URL**处输入您刚刚记录的ECS私有地址和MySQL的默认端口号3306。

> **说明** 当前VPC环境下的自建MySQL数据源暂不支持**测试连通性**，因此连通性测试失败是正常现象。

   vii. 单击相应资源组后的**测试连通性**。

   数据同步时，一个任务只能使用一种资源组。您需要在每种资源组上单独测试连通性，以保证同步任务使用的数据集成资源组能够与数据源连通，否则将无法正常执行数据同步任务。详情请参见配置资源组与网络连通。

viii. 测试连通性通过后，单击**完成**。

2. 创建MaxCompute表。

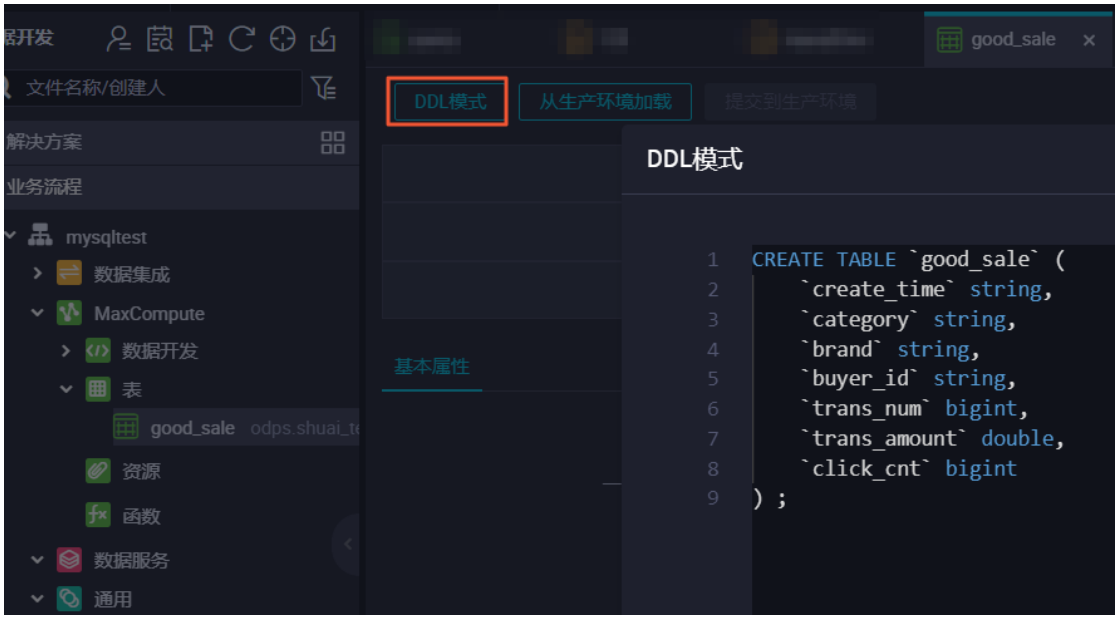
   您需要通过DataWorks创建一个表，用于接收来自MySQL的测试数据。

   i. 单击左上角的图标，选择**全部产品 > DataStudio（数据开发）**。

   ii. 新建一个**业务流程**，详情请参见创建业务流程。

   iii. 右键单击新建的业务流程，选择**新建 > MaxCompute > 表**。

iv. 输入您的MaxCompute表名称，本例中使用和MySQL数据库表一样的名称good_sale。单击**DDL模式**后，输入您的建表语句并**生成表结构**。



本例中使用的建表语句如下，请注意数据类型的转换。

```
CREATE TABLE IF NOT EXISTS good_sale(
   create_time string,
   category STRING,
   brand STRING,
   buyer_id STRING,
   trans_num BIGINT,
   trans_amount DOUBLE,
   click_cnt BIGINT
   );
```

v. 输入表的**中文名**后，单击**提交到生产环境**，完成MaxCompute表good_sale的创建。

3. 配置数据集成任务。

i. 右键单击业务流程，选择**新建 > 数据集成 > 离线同步**，创建一个数据集成任务。

ii. 选择您的数据来源为您刚添加的MySQl数据源，数据去向为默认MaxCompute数据源odps_first，单击**转换脚本**切换数据集成任务为脚本模式。

此时，如果产生报错或您无法选择数据来源的**表**，都属于正常现象，直接转换为脚本模式即可。

iii. 单击页面右侧的**数据集成资源组配置**，选中已购买的独享资源组。

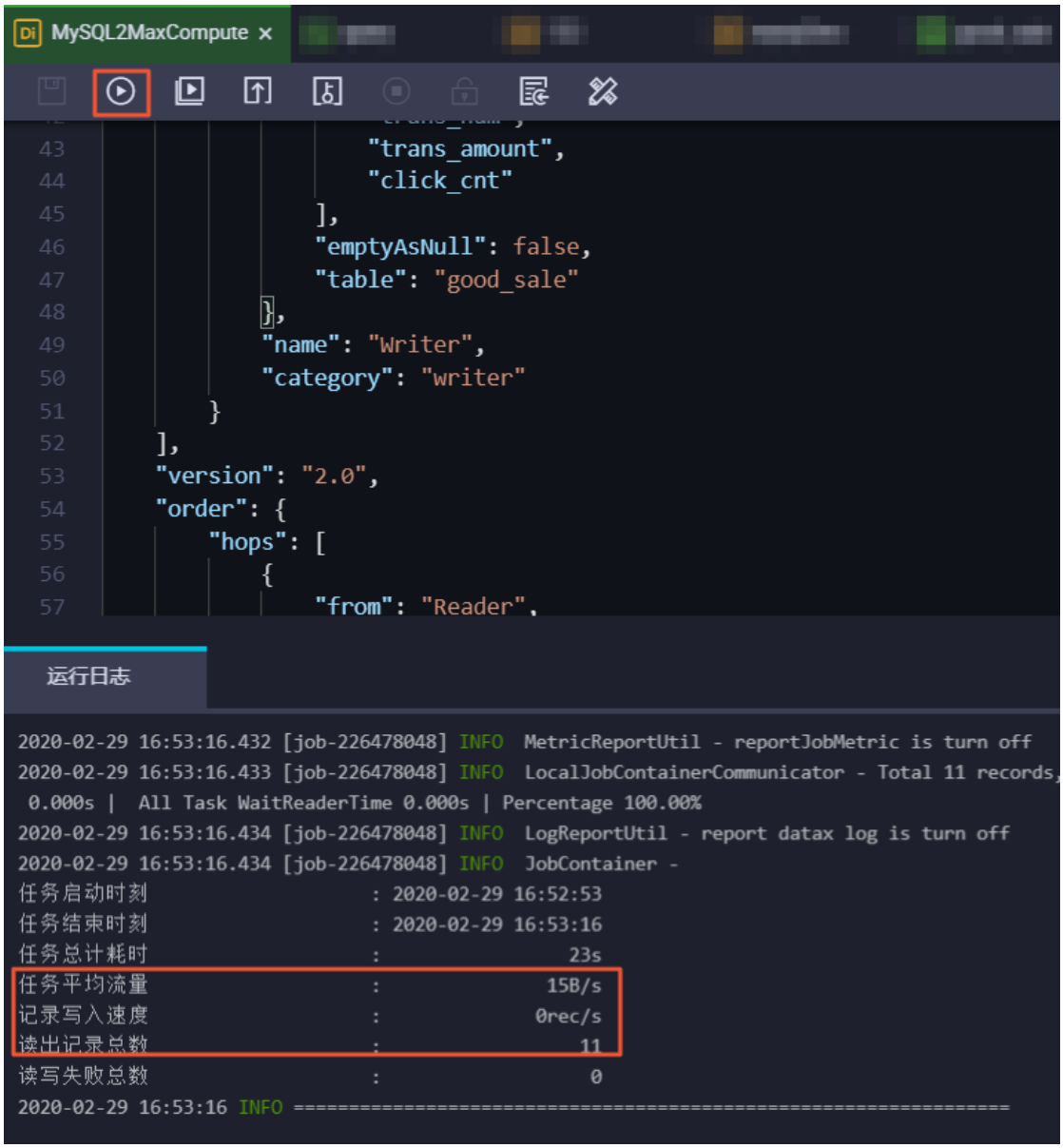
如果未切换任务资源组为数据集成独享资源，后续您的任务将无法成功运行。

iv. 填写数据集成任务脚本内容如下。

```
{
    "type": "job",
    "steps": [
        {
            "stepType": "mysql",
            "parameter": {
                "column": [//源列名
                    "create_time",
                    "category",
                    "brand",
                    "buyer_id",
                    "trans_num",
                    "trans_amount",
                    "click_cnt"
                ],
                "connection": [
                    {
                        "datasource": "shuai",//源数据源
```

```
                        "table": [
                            "good_sale"//源数据库表名，此处必须为方括号数组格式。
                        ]
                    }
                ],
                "where": "",
                "splitPk": "",
                "encoding": "UTF-8"
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "odps",
            "parameter": {
                "partition": "",
                "truncate": true,
                "datasource": "odps_first",//目标数据源
                "column": [//目标列名
                    "create_time",
                    "category",
                    "brand",
                    "buyer_id",
                    "trans_num",
                    "trans_amount",
                    "click_cnt"
                ],
                "emptyAsNull": false,
                "table": "good_sale"//目标表名
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "version": "2.0",
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
        "errorLimit": {
            "record": "0"
        },
        "speed": {
            "throttle": false,
            "concurrent": 2
        }
    }
}
```

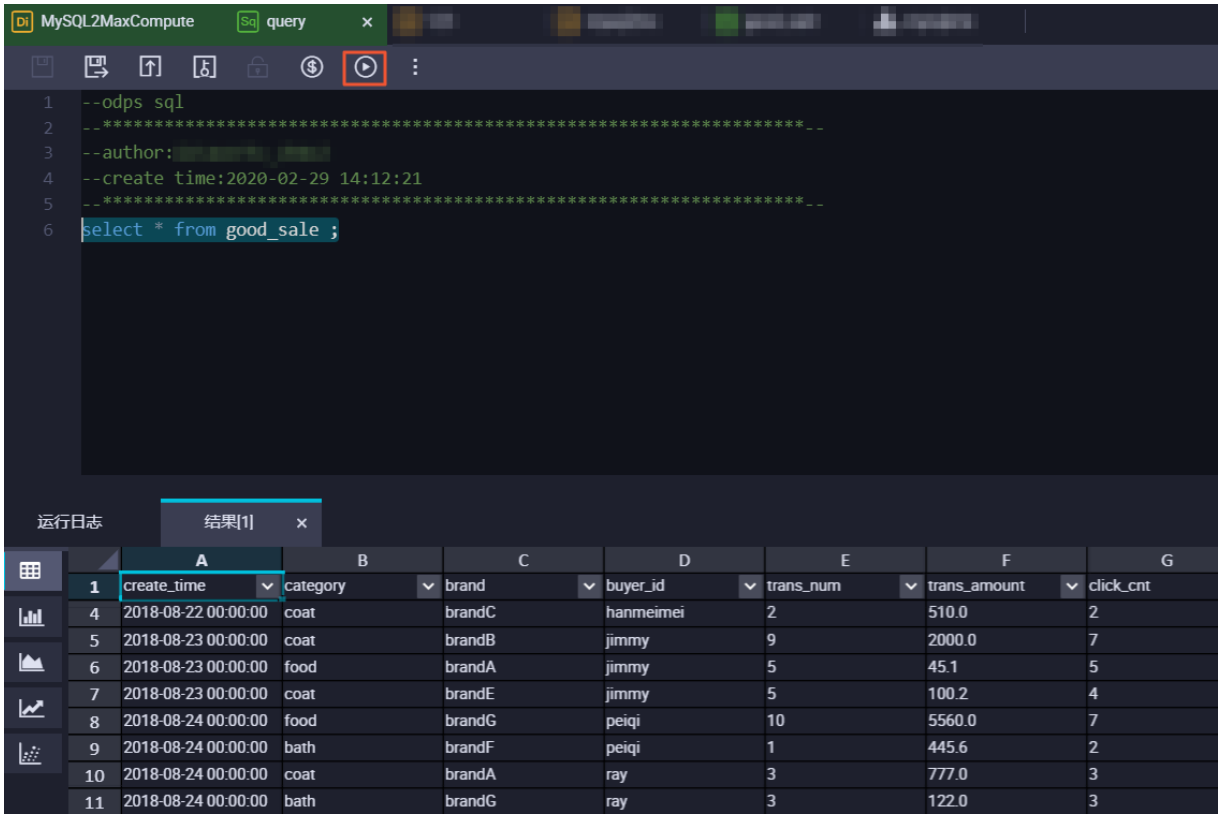
v. 单击运行，您可以在下方的**运行日志**查看数据是否已传输到MaxCompute。



执行结果

您可以新建一个**ODQP SQL**类型的节点，用于查询当前MaxCompute表中的数据。

输入您的查询语句 `select * from good_sale ;` ，单击**运行**，即可看到当前已传入MaxCompute表中的数据。



	A	B	C	D	E	F	G
1	create_time	category	brand	buyer_id	trans_num	trans_amount	click_cnt
4	2018-08-22 00:00:00	coat	brandC	hanmeimei	2	510.0	2
5	2018-08-23 00:00:00	coat	brandB	jimmy	9	2000.0	7
6	2018-08-23 00:00:00	food	brandA	jimmy	5	45.1	5
7	2018-08-23 00:00:00	coat	brandE	jimmy	5	100.2	4
8	2018-08-24 00:00:00	food	brandG	peiqi	10	5560.0	7
9	2018-08-24 00:00:00	bath	brandF	peiqi	1	445.6	2
10	2018-08-24 00:00:00	coat	brandA	ray	3	777.0	3
11	2018-08-24 00:00:00	bath	brandG	ray	3	122.0	3

1.8. 迁移Oracle数据至MaxCompute最佳实践

本文为您介绍如何通过DataWorks的数据集成功能，迁移Oracle数据至MaxCompute。

前提条件

- 准备DataWorks环境
  i. 开通MaxCompute和DataWorks。
  ii. 创建工作空间（本文以简单模式的工作空间为例）。

iii. 在DataWorks上创建业务流程。详情请参见创建业务流程。

- 准备Oracle环境

  本文中的Oracle安装在云服务器ECS上，ECS具体配置如下。为了让网络互通，您需要给ECS配置公网IP，并且配置ECS的安全组规则放通Oracle数据库的常用端口1521。关于ECS安全组配置详情请参见修改安全组规则。

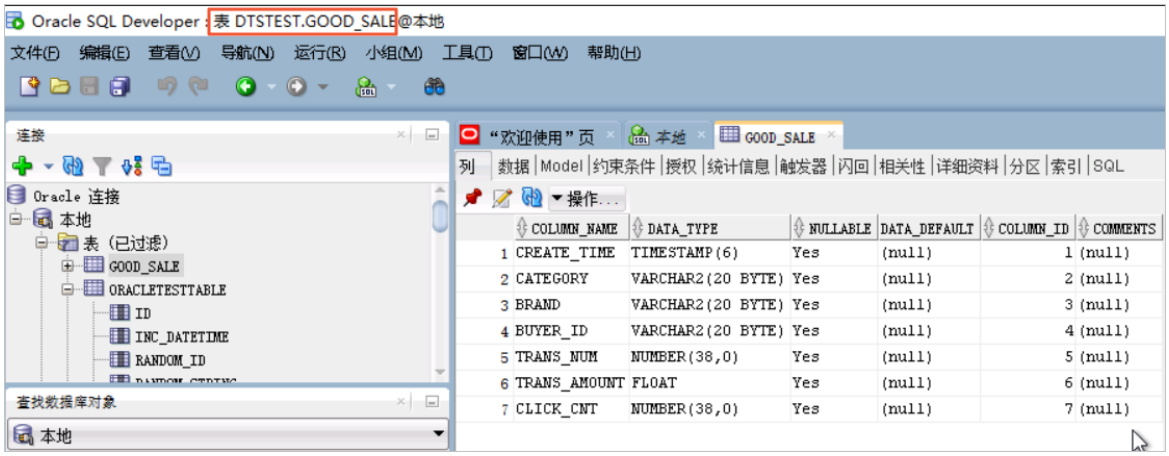
如上图所示，本文中的ECS规格为**ecs.c5.xlarge**，使用专有网络，区域为华东1（杭州）。

背景信息

本文需要使用DataWorks Oracle Reader读取Oracle中的测试数据，详情请参见Oracle Reader。

准备Oracle测试数据

1. 进入Oracle图形化操作界面，新建表DTSTEST.GOOD_SALE，主要包括create_time、category、brand、buyer_id、trans_num、trans_amount、click_cnt这7列。



2. 插入测试数据，本文中手动插入数据如下。

```
insert into good_sale values('28-12月-19','厨具','品牌A','hanmeimei','6','80.6','4');
insert into good_sale values('21-12月-19','生鲜','品牌B','lilei','7','440.6','5');
insert into good_sale values('29-12月-19','衣服','品牌C','lily','12','351.9','9');
commit;
```

3. 插入数据后，执行如下语句查看表数据。

```
select * from good_sale;
```

通过DataWorks将数据从Oracle迁移至MaxCompute

1. 进入**数据开发**页面。
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**工作空间列表**。
   iii. 选择工作空间所在地域后，单击相应工作空间后的**进入数据开发**。
2. 在**数据开发**页面创建目标表，用于接收从Oracle迁移的数据。

i. 右键单击已创建的**业务流程**，选择**新建 > MaxCompute > 表**。

ii. 在**新建表**页面，选择引擎类型并输入**表名**。

iii. 在表的编辑页面，单击**DDL模式**。

iv. 在**DDL模式**对话框，输入建表语句，单击**生成表结构**。

```
CREATE TABLE good_sale
(
    create_time     string,
    category        string,
    brand           string,
    buyer_id        string,
    trans_num       bigint,
    trans_amount    double,
    click_cnt       bigint
) ;
```

在建表过程中，需要考虑Oracle数据类型与MaxCompute数据类型的映射，Oracle Reader支持的数据类型请参见类型转换列表。

v. 单击**提交到生产环境**。

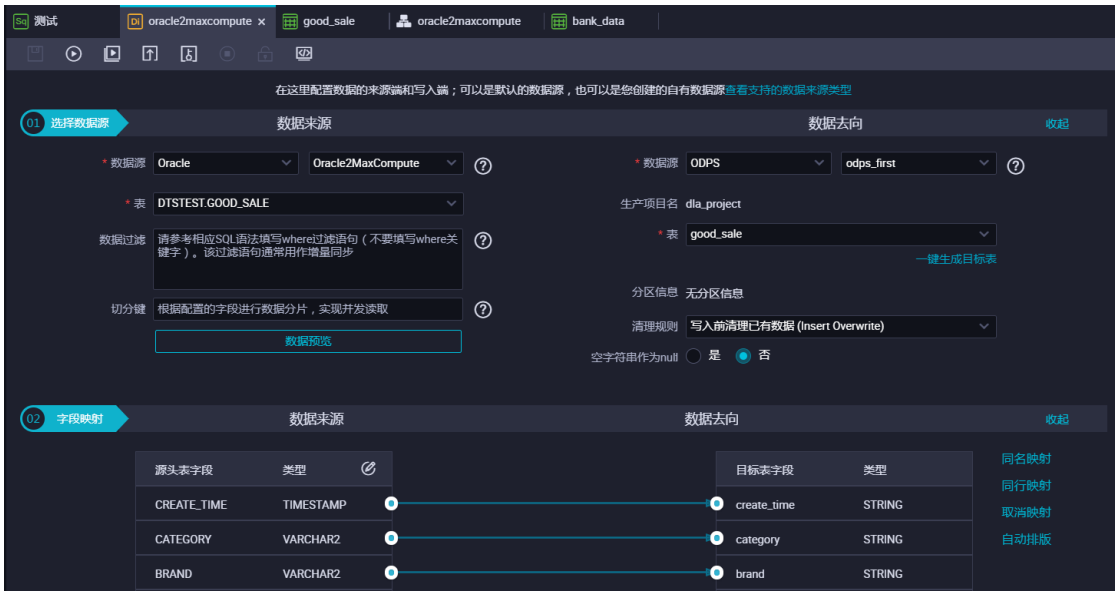
3. 新建Oracle数据源，详情请参见配置Oracle数据源。

4. 创建离线同步节点。

i.

ii.

iii. 成功创建数据同步节点后，选择**数据源**为您刚刚添加的Oracle数据源，**表**为您刚刚创建的测试表格，选择**同名映射**。其它参数保持默认配置。



iv. 单击  图标运行代码。

v. 您可以在**运行日志**查看运行结果。

验证结果

1. 右键单击业务流程，选择**新建 > MaxCompute > ODPS SQL**。
2. 在**新建节点**对话框中输入**节点名称**，并单击**提交**。
3. 在ODPS SQL节点编辑页面输入如下语句。

```
--查看是否成功写入MaxCompute。
select * from good_sale;
```

4. 单击  图标运行代码。
5. 您可以在**运行日志**查看运行结果。

1.9. Kafka数据迁移MaxCompute最佳实践

本文为您介绍如何使用DataWorks数据集成，将Kafka集群上的数据迁移至MaxCompute。

前提条件

- 开通MaxCompute和DataWorks。
- 开通DataWorks。
- 在DataWorks上完成创建业务流程，本例使用DataWorks简单模式。详情请参见创建业务流程。
- 搭建Kafka集群

  进行数据迁移前，您需要保证自己的Kafka集群环境正常。本文使用阿里云EMR服务自动化搭建Kafka集群，详细过程请参见Kafka快速入门。

  本文使用的EMR Kafka版本信息如下：

  - EMR版本：EMR-3.12.1
  - 集群类型：Kafka
  - 软件信息：Ganglia 3.7.2 ZooKeeper 3.4.12 Kafka 2.11-1.0.1 Kafka-Manager 1.3.3.16

  Kafka集群使用专有网络，区域为华东1（杭州），主实例组ECS计算资源配置公网及内网IP。

背景信息

Kafka是一款分布式发布与订阅的消息中间件，具有高性能、高吞量的特点被广泛使用，每秒能处理上百万的消息。Kafka适用于流式数据处理，主要应用于用户行为跟踪、日志收集等场景。

一个典型的Kafka集群包含若干个生产者（Producer）、Broker、消费者（Consumer）以及一个Zookeeper集群。Kafka集群通过Zookeeper管理自身集群的配置并进行服务协同。

Topic是Kafka集群上最常用的消息的集合，是一个消息存储逻辑概念。物理磁盘不存储Topic，而是将Topic中具体的消息按分区（Partition）存储在集群中各个节点的磁盘上。每个Topic可以有多个生产者向它发送消息，也可以有多个消费者向它拉取（消费）消息。

每个消息被添加到分区时，会分配一个Offset（偏移量，从0开始编号），是消息在一个分区中的唯一编号。

步骤一：准备Kafka数据

您需要在Kafka集群创建测试数据。为保证您可以顺利登录EMR集群Header主机，以及保证MaxCompute和DataWorks可以顺利和EMR集群Header主机通信，请您首先配置EMR集群Header主机安全组，放行TCP 22及TCP 9092端口。

1. 登录EMR集群Header主机地址。
   i. 进入EMR Hadoop控制台。
   ii. 在顶部导航栏，单击**集群管理**。
   iii. 在显示的页面，找到您需要创建测试数据的集群，进入集群详情页。
   iv. 在集群详情页面，单击主机列表，确认EMR集群Header主机地址，并通过SSH连接远程登录。
2. 创建测试Topic。

   执行如下命令创建测试所使用的Topic testkafka。

   ```
   [root@emr-header-1 ~]# kafka-topics.sh --zookeeper emr-header-1:2181/kafka-1.0.1 --part
   itions 10 --replication-factor 3 --topic testkafka  --create
   Created topic "testkafka".
   ```

3. 写入测试数据。

   执行如下命令，可以模拟生产者向Topic testkafka中写入数据。由于Kafka用于处理流式数据，您可以持续不断的向其中写入数据。为保证测试结果，建议写入10条以上的数据。

   ```
   [root@emr-header-1 ~]# kafka-console-producer.sh --broker-list emr-header-1:9092 --topi
   c testkafka
   >123
   >abc
   >
   ```

   您可以同时再打开一个SSH窗口，执行如下命令，模拟消费者验证数据是否已成功写入Kafka。当数据写入成功时，您可以看到已写入的数据。

   ```
   [root@emr-header-1 ~]# kafka-console-consumer.sh --bootstrap-server emr-header-1:9092 -
   -topic testkafka --from-beginning
   123
   abc
   ```

步骤二：在DataWorks上创建目标表

在DataWorks上创建目标表用以接收Kafka数据。

1. 进入**数据开发**页面。
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**工作空间列表**。
   iii. 单击相应工作空间后的**数据开发**。
2. 右键单击**业务流程**，选择**新建 > MaxCompute > 表**。
3. 在**新建表**页面，选择引擎类型并输入**表名**。
4. 单击**DDL模式**。在**DDL模式**对话框中，输入如下建表语句，单击**生成表结构**。

```
CREATE TABLE testkafka
(
 key             string,
 value           string,
 partition1      string,
 timestamp1      string,
 offset          string,
 t123            string,
 event_id        string,
 tag             string
) ;
```

其中的每一列，对应于DataWorks数据集成Kafka Reader的默认列：

- __key__表示消息的key。
- __value__表示消息的完整内容 。
- __partition__表示当前消息所在分区。
- __headers__表示当前消息headers信息。
- __offset__表示当前消息的偏移量。
- __timestamp__表示当前消息的时间戳。

您还可以自主命名，详情参见Kafka Reader。

5. 单击**提交到生产环境**并**确认**。

步骤三：同步数据

1. 新建独享数据集成资源组。

   由于当前DataWorks的默认资源组无法完美支持Kafka插件，您需要使用独享数据集成资源组完成数据同步。详情请参见新增和使用独享数据集成资源组。

2. 新建数据集成节点。

   i.

   ii.

3.

4.

5. 配置脚本，示例代码如下。

```
{
    "type": "job",
    "steps": [
        {
            "stepType": "kafka",
            "parameter": {
                "server": "47.xxx.xxx.xxx:9092",
                "kafkaConfig": {
                    "group.id": "console-consumer-83505"
                },
                "valueType": "ByteArray",
                "column": [
                    "__key__",
                    "__value__",
```

```
                    "__value__",
                    "__partition__",
                    "__timestamp__",
                    "__offset__",
                    "'123'",
                    "event_id",
                    "tag.desc"
                ],
                "topic": "testkafka",
                "keyType": "ByteArray",
                "waitTime": "10",
                "beginOffset": "0",
                "endOffset": "3"
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "odps",
            "parameter": {
                "partition": "",
                "truncate": true,
                "compress": false,
                "datasource": "odps_first",
                "column": [
                    "key",
                    "value",
                    "partition1",
                    "timestamp1",
                    "offset",
                    "t123",
                    "event_id",
                    "tag"
                ],
                "emptyAsNull": false,
                "table": "testkafka"
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "version": "2.0",
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
        "errorLimit": {
            "record": ""
        },
        "speed": {
```

```
            "throttle": false,
            "concurrent": 1,
        }
    }
}
```

您可以通过在Header主机上使用**kafka-consumer-groups.sh --bootstrap-server emr-header-1:9092 --list**命令查看group.id参数，及消费者的Group名称。

```
[root@emr-header-1 ~]#  kafka-consumer-groups.sh  --bootstrap-server emr-header-1:9092
--list
Note: This will not show information about old Zookeeper-based consumers.
_emr-client-metrics-handler-group
console-consumer-69493
console-consumer-83505
console-consumer-21030
console-consumer-45322
console-consumer-14773
```

以console-consumer-83505为例，您可以根据该参数在Header主机上使用**kafka-consumer-groups.sh --bootstrap-server emr-header-1:9092 --describe --group console-consumer-83505**命令确认beginOffset及endOffset参数。

```
[root@emr-header-1 ~]# kafka-consumer-groups.sh --bootstrap-server emr-header-1:9092 --
describe --group console-consumer-83505
Note: This will not show information about old Zookeeper-based consumers.
Consumer group 'console-consumer-83505' has no active members.
TOPIC                          PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG        CO
NSUMER-ID                                       HOST                           CLIENT-I
D
testkafka                      6          0               0               0          -
-                              -
test                           6          3               3               0          -
-                              -
testkafka                      0          0               0               0          -
-                              -
testkafka                      1          1               1               0          -
-                              -
testkafka                      5          0               0               0          -
-                              -
```

6. 配置调度资源组。
   i. 在节点编辑页面的右侧导航栏，单击**调度配置**。
   ii. 在**资源属性**区域，选择**调度资源组**为您创建的独享数据集成资源组。

   > **说明** 如果您需要将Kafka的数据周期性（例如每小时）写入MaxCompute，您可以使用beginDateTime及endDateTime参数，设置数据读取的时间区间为1小时，然后每小时调度一次数据集成任务。详情请参见Kafka Reader。

7. 单击  图标运行代码。
8. 您可以在**运行日志**查看运行结果。

后续步骤

您可以新建一个ODPS SQL节点运行SQL语句，查看从Kafka同步数据至MaxCompute是否成功。详情请参见使用临时查询运行SQL语句（可选）。

1.10. JSON数据从OSS迁移至MaxCompute

本文为您介绍如何通过DataWorks数据集成，将JSON数据从OSS迁移至MaxCompute，并使用MaxCompute内置字符串函数GET_JSON_OBJECT提取JSON信息。

前提条件

- 开通MaxCompute和DataWorks。
- 开通DataWorks。
- 在DataWorks上完成创建业务流程，本例使用DataWorks简单模式。详情请参见创建业务流程。
- 将JSON文件重命名为后缀为 `.txt` 的文件，并上传至OSS。本文中OSS Bucket地域为华东2（上海）。示例文件如下。

```
{
    "store": {
        "book": [
             {
                "category": "reference",
                "author": "Nigel Rees",
                "title": "Sayings of the Century",
                "price": 8.95
             },
             {
                "category": "fiction",
                "author": "Evelyn Waugh",
                "title": "Sword of Honour",
                "price": 12.99
             },
             {
                 "category": "fiction",
                 "author": "J. R. R. Tolkien",
                 "title": "The Lord of the Rings",
                 "isbn": "0-395-19395-8",
                 "price": 22.99
             }
          ],
          "bicycle": {
              "color": "red",
              "price": 19.95
          }
    },
    "expensive": 10
}
```

将JSON数据从OSS迁移至MaxCompute

1. 新增OSS数据源。详情请参见配置OSS数据源。
2. 在DataWorks上新建数据表，用于存储迁移的JSON数据。
   i.
   ii. 在**新建表**页面，选择引擎类型并输入**表名**。
   iii. 在表的编辑页面，单击**DDL模式**。
   iv. 在**DDL模式**对话框，输入如下建表语句，单击**生成表结构**。

   ```
   create table mqdata (mq_data string);
   ```

   v. 单击**提交到生产环境**。
3. 新建离线同步节点。
   i.
   ii.
   iii.
   iv.
   v.
   vi. 修改JSON代码后，单击按钮。

   示例代码如下。

   ```
   {
       "type": "job",
       "steps": [
           {
               "stepType": "oss",
               "parameter": {
                   "fieldDelimiterOrigin": "^",
                   "nullFormat": "",
                   "compress": "",
                   "datasource": "OSS_userlog",
                   "column": [
                       {
                           "name": 0,
                           "type": "string",
                           "index": 0
                       }
                   ],
                   "skipHeader": "false",
                   "encoding": "UTF-8",
                   "fieldDelimiter": "^",
                   "fileFormat": "binary",
                   "object": [
                       "applog.txt"
                   ]
               },
               "name": "Reader",
               "category": "reader"
           },
           {
               "stepType": "odps",
   ```

```
            "parameter": {
                "partition": "",
                "isCompress": false,
                "truncate": true,
                "datasource": "odps_first",
                "column": [
                    "mqdata"
                ],
                "emptyAsNull": false,
                "table": "mqdata"
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "version": "2.0",
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
        "errorLimit": {
            "record": ""
        },
        "speed": {
            "concurrent": 2,
            "throttle": false,
        }
    }
}
```

结果验证

1. 新建ODPS SQL节点。
   i. 右键单击业务流程，选择**新建 > MaxCompute > ODPS SQL**。
   ii. 在**新建函数**对话框中，输入**函数名称**，单击**提交**。
   iii. 在ODPS SQL节点编辑页面输入如下语句。

```
--查询表mq_data数据。
SELECT * from mqdata;
--获取JSON文件中的EXPENSIVE值。
SELECT GET_JSON_OBJECT(mqdata.MQdata,'$.expensive') FROM mqdata;
```

   iv. 单击  图标运行代码。
   v. 您可以在**运行日志**查看运行结果。

1.11. JSON数据从MongoDB迁移至MaxCompute

本文为您介绍如何通过DataWorks的数据集成功能，将从MongoDB提取的JSON字段迁移至MaxCompute。

前提条件

- 开通MaxCompute和DataWorks。
- 开通DataWorks。
- 在DataWorks上完成创建业务流程，本例使用DataWorks简单模式。详情请参见创建业务流程。

在MongoDB上准备测试数据

1. 账号准备。

   在数据库内新建用户，用于DataWorks添加数据源。本示例执行如下命令。

   ```
   db.createUser({user:"bookuser",pwd:"123456",roles:["root"]})
   ```

   新建用户名为bookuser，密码为123456，权限为root。

2. 数据准备。

   将数据上传至MongoDB数据库。本示例使用阿里云的云数据库MongoDB版，网络类型为VPC（需申请公网地址，否则无法与DataWorks默认资源组互通），测试数据如下。

```
{
    "store": {
        "book": [
            {
                "category": "reference",
                "author": "Nigel Rees",
                "title": "Sayings of the Century",
                "price": 8.95
                },
            {
                "category": "fiction",
                "author": "Evelyn Waugh",
                "title": "Sword of Honour",
                "price": 12.99
                },
            {
                "category": "fiction",
                "author": "J. R. R. Tolkien",
                "title": "The Lord of the Rings",
                "isbn": "0-395-19395-8",
                "price": 22.99
                }
                    ],
        "bicycle": {
            "color": "red",
            "price": 19.95
                }
                    },
        "expensive": 10
            }
```

3. 在MongoDB的DMS控制台，本示例使用的数据库为admin，集合为userlog。执行如下命令，查看已上传的数据。

```
db.userlog.find().limit(10)
```

通过DataWorks将JSON数据从MongoDB迁移至MaxCompute

1. 登录DataWorks控制台。
2. 在DataWorks上创建目标表。用以接收从MongoDB迁移的数据。
   i. 右键单击已创建的**业务流程**，选择**新建 > MaxCompute > 表**。
   ii. 在**新建表**页面，选择引擎类型并输入**表名**。
   iii. 在表的编辑页面，单击**DDL模式**。
   iv. 在**DDL模式**对话框，输入建表语句，单击**生成表结构**。

   ```
   create table mqdata (mqdata string);
   ```

   v. 单击**提交到生产环境**。
3. 新增MongoDB数据源，详情请参见配置MongoDB数据源。
4. 创建离线同步节点。
   i.

ii.

iii.

iv.

v.

vi. 输入如下脚本。

```
{
    "type": "job",
    "steps": [
    {
        "stepType": "mongodb",
        "parameter": {
            "datasource": "mongodb_userlog",//数据源名称。
            "column": [
                {
                "name": "store.bicycle.color", //JSON字段路径，本例中提取color值。
                "type": "document.String" //非一层子属性以最终获取的类型为准。假如您选取的JSON字段为一级字段，例如本例中的expensive，则直接填写string即可。
                }
              ],
            "collectionName": "userlog"    //集合名称。
            },
        "name": "Reader",
        "category": "reader"
        },
        {
            "stepType": "odps",
            "parameter": {
            "partition": "",
            "isCompress": false,
            "truncate": true,
            "datasource": "odps_first",
            "column": [
            "mqdata"  //MaxCompute表列名。
            ],
            "emptyAsNull": false,
            "table": "mqdata"
            },
            "name": "Writer",
            "category": "writer"
            }
            ],
            "version": "2.0",
            "order": {
            "hops": [
            {
            "from": "Reader",
            "to": "Writer"
            }
            ]
            },
            "setting": {
            "errorLimit": {
```

```
            "record": ""
            },
            "speed": {
            "concurrent": 2,
            "throttle": false,
            }
            }
        }
```

vii. 单击图标运行代码。

viii. 您可以在**运行日志**查看运行结果。

验证结果

1. 右键单击业务流程，选择**新建 > MaxCompute > ODPS SQL**。
2. 在**新建节点**对话框中输入**节点名称**，并单击**提交**。
3. 在ODPS SQL节点编辑页面输入如下语句。

```
SELECT * from mqdata;
```

4. 单击图标运行代码。
5. 您可以在**运行日志**查看运行结果。

1.12. Elasticsearch数据迁移至MaxCompute

本文为您介绍如何通过DataWorks数据同步功能，迁移阿里云Elasticsearch集群上的数据至MaxCompute。

前提条件

- 已开通MaxCompute服务。

  开通指导，详情请参见开通MaxCompute。
- 已开通DataWork服务。

  开通指导，详情请参见开通DataWorks。
- 在DataWorks上已完成创建业务流程。

  本例使用DataWorks简单模式，详情请参见创建业务流程。
- 已搭建阿里云Elasticsearch集群。

  进行数据迁移前，您需要保证自己的阿里云Elasticsearch集群环境正常。搭建阿里云Elasticsearch集群的详细过程，请参见快速入门。

  本示例中阿里云Elasticsearch的具体配置如下：
  - 地域：华东2（上海）
  - 可用区：上海可用区B
  - 版本：5.5.3 with Commercial Feature

背景信息

Elasticsearch是一个基于Lucene的搜索服务器，它提供了一个多用户分布式的全文搜索引擎。Elasticsearch是遵从Apache开源条款的一款开源产品，是当前主流的企业级搜索引擎。

阿里云Elasticsearch提供Elasticsearch 5.5.3 with Commercial Feature、6.3.2 with Commercial Feature、6.7.0 with Commercial Feature及商业插件X-pack服务，致力于数据分析、数据搜索等场景服务。在开源Elasticsearch基础上提供企业级权限管控、安全监控告警、自动报表生成等功能。

操作步骤

1. 在Elasticsearch上创建源表。详情请参见通过DataWorks将MaxCompute数据同步至Elasticsearch。
2. 在MaxCompute上创建目标表。
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**工作空间列表**。
   iii. 在**工作空间列表**页面，单击相应工作空间后的**数据开发**。
   iv. 在**数据开发**页面，右键单击目标工作流程，选择**新建 > MaxCompute > 表**。
   v. 在弹出的**新建表**对话框中，填写**表名**，并单击**提交**。

      > **说明**
      > - 表名必须以字母开头，不能包含中文或特殊字符。
      > - 如果绑定多个实例，则需要选择MaxCompute引擎实例。

   vi. 在表的编辑页面，单击**DDL模式**。
   vii. 在**DDL模式**对话框，输入如下建表语句，单击**生成表结构**。

```
create table elastic2mc_bankdata
(
age             string,
job             string,
marital         string,
education       string,
default         string,
housing         string,
loan            string,
contact         string,
month           string,
day of week     string
);
```

   viii. 单击**提交到生产环境**。
3. 同步数据。
   i.
   ii.
   iii.
   iv.
   v.

vi. 配置脚本。

示例代码如下。代码释义请参见Elasticsearch Reader。

```
{
    "type": "job",
    "steps": [
        {
            "stepType": "elasticsearch",
            "parameter": {
                "retryCount": 3,
                "column": [
                    "age",
                    "job",
                    "marital",
                    "education",
                    "default",
                    "housing",
                    "loan",
                    "contact",
                    "month",
                    "day_of_week",
                    "duration",
                    "campaign",
                    "pdays",
                    "previous",
                    "poutcome",
                    "emp_var_rate",
                    "cons_price_idx",
                    "cons_conf_idx",
                    "euribor3m",
                    "nr_employed",
                    "y"
                ],
                "scroll": "1m",
                "index": "es_index",
                "pageSize": 1,
                "sort": {
                    "age": "asc"
},
                "type": "elasticsearch",
                "connTimeOut": 1000,
                "retrySleepTime": 1000,
                "endpoint": "http://es-cn-xxxx.xxxx.xxxx.xxxx.com:9200",
                "password": "xxxx",
                "search": {
                    "match_all": {}
                },
                "readTimeOut": 5000,
                "username": "xxxx"
            },
            "name": "Reader",
            "category": "reader"
        },
        {
```

```
            "stepType": "odps",
            "parameter": {
                "partition": "",
                "truncate": true,
                "compress": false,
                "datasource": "odps_first",
                "column": [
                    "age",
                    "job",
                    "marital",
                    "education",
                    "default",
                    "housing",
                    "loan",
                    "contact",
                    "month",
                    "day_of_week",
                    "duration",
                    "campaign",
                    "pdays",
                    "previous",
                    "poutcome",
                    "emp_var_rate",
                    "cons_price_idx",
                    "cons_conf_idx",
                    "euribor3m",
                    "nr_employed",
                    "y"
                ],
                "emptyAsNull": false,
                "table": "elastic2mc_bankdata"
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "version": "2.0",
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
        "errorLimit": {
            "record": "0"
        },
        "speed": {
            "throttle": false,
            "concurrent": 1,
            "dmu": 1
        }
```

```
    }
}
```

> **说明** 您可以在创建的阿里云Elasticsearch集群的**基本信息**中，查看公网地址和公网端口信息。

vii. 单击图标运行代码。

viii. 您可以在**运行日志**查看运行结果。

4. 查看结果。

i. 右键单击业务流程，选择**新建 > MaxCompute > ODPS SQL**。

ii. 在**新建节点**对话框中输入**节点名称**，并单击**提交**。

iii. 在ODPS SQL节点编辑页面输入如下语句。

```
SELECT * FROM elastic2mc_bankdata;
```

iv. 单击图标运行代码。

v. 您可以在**运行日志**查看运行结果。

1.13. OTSStream配置同步任务

OTSStream插件主要用于导出Table Store增量数据，本文将为您介绍如何通过OTSStream配置同步任务。

背景信息

OTSStream插件与全量导出插件不同，增量导出插件仅支持多版本模式，且不支持指定列。增量数据可以看作操作日志，除数据本身外还附有操作信息。详情请参见OTSStream Reader。

> **说明** OTSStream配置同步任务时，请注意以下问题：
>
> - 如果配置任务为日调度，您可以读取当前时间24小时以内的数据，但会丢失当前时间前5分钟的数据。建议您配置任务为小时调度。
> - 设置的结束时间不能超过系统显示的时间，即您设置的结束时间要比运行时间早5分钟。
> - 配置日调度会出现数据丢失的情况。
> - 不可以配置周期调度和月调度。
>
> 开始时间和结束时间需要包含操作Table Store表的时间。例如，20171019162000您向Table Store插入2条数据，则开始时间设置为20171019161000，结束时间设置为20171019162600。

新增数据源

1. 登录DataWorks控制台，单击相应工作空间后的**进入数据集成**。

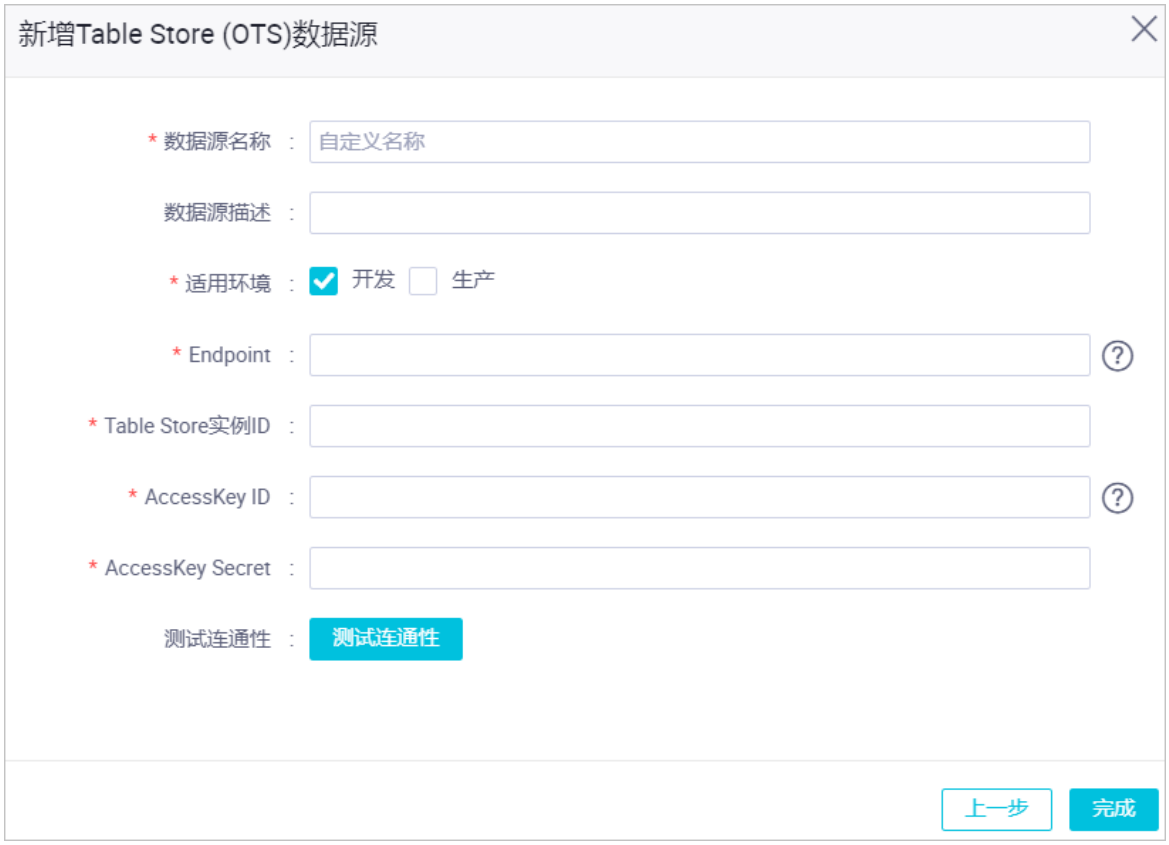
   如果您已在DataWorks的某个功能模块，请单击左上角的图标，选择**全部产品 > 数据集成**，即可跳转至**数据集成**页面。

2. 单击左侧导航栏中的**数据源**，进入**工作空间管理 > 数据源管理**页面。

3. 单击右上角的**新增数据源**。

4. 在**新增数据源**对话框中，选择数据源类型为**Table Store（OTS）** 。

5. 填写Table Store（OTS）数据源的各配置项。



参数	描述
数据源名称	数据源名称必须以字母、数字、下划线组合，且不能以数字和下划线开头。
数据源描述	对数据源进行简单描述，不得超过80个字符。
适用环境	可以选择**开发**或**生产**环境。 **说明** 仅标准模式工作空间会显示此配置。
Endpoint	Table Store服务对应的Endpoint。
Table Store实例ID	Table Store服务对应的实例ID。
AccessKey ID	访问密钥中的AccessKey ID，您可以进入**用户信息管理**页面进行复制。
AccessKey Secret	访问密钥中的AccessKey Secret，相当于登录密码。

6. 单击**测试连通性**。
7. 测试连通性通过后，单击**完成**。

通过向导模式配置同步任务

1. 在**数据源管理**页面，单击左上角的图标，选择**全部产品 > 数据集成**。

2. 单击**首页**中的**新建同步任务**。
3. 在**新建节点**对话框中，输入**节点名称**并选择**目标文件夹**，单击**提交**。
4. 进入离线同步节点的配置页面，选择数据来源。

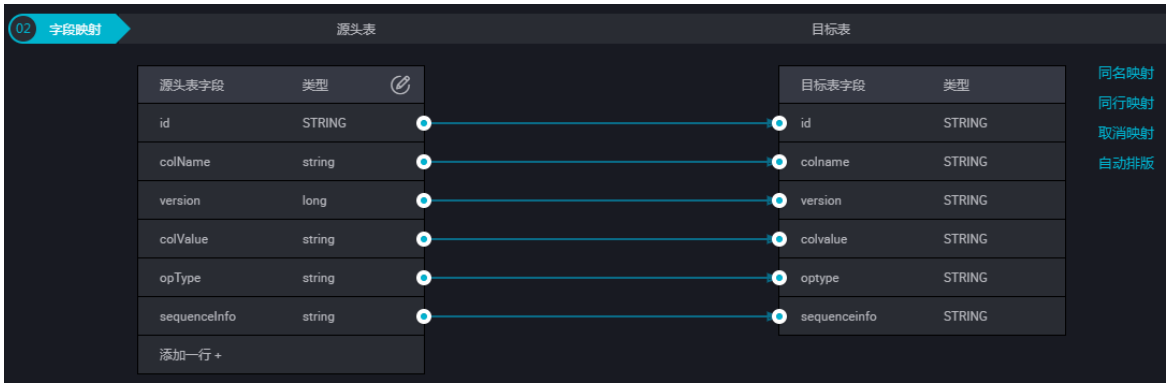
参数	描述
数据源	输入数据源的名称。
表	导出增量数据的表的名称。该表需要开启Stream，您可以在建表时开启，或使用UpdateTable接口开启。
开始时间	增量数据的时间范围（左闭右开）的左边界，格式为yyyymmddhh24miss，单位为毫秒。
结束时间	增量数据的时间范围（左闭右开）的右边界，格式为yyyymmddhh24miss，单位为毫秒。
状态表	用于记录状态的表的名称。
最大重试次数	从TableStore中读增量数据时，每次请求的最大重试次数，默认是*30*。
导出时序信息	是否导出时序信息，包括数据的写入时间等信息。

5. 选择数据去向。

参数	描述
数据源	输入配置的数据源名称。
表	选择需要同步的表。
分区信息	此处需同步的表是非分区表，所以无分区信息。
清理规则	◦ **写入前清理已有数据**：导数据之前，清空表或者分区的所有数据，相当于 `insert overwrite` 。 ◦ **写入前保留已有数据**：导数据之前，不清理任何数据，每次运行数据都是追加进去的，相当于 `insert into` 。
空字符串作为null	默认值为**否**。

6. 字段映射。

   左侧的源头表字段和右侧的目标表字段为一一对应的关系。单击**添加一行**可以增加单个字段，鼠标放至需要删除的字段上，即可单击**删除**图标进行删除。

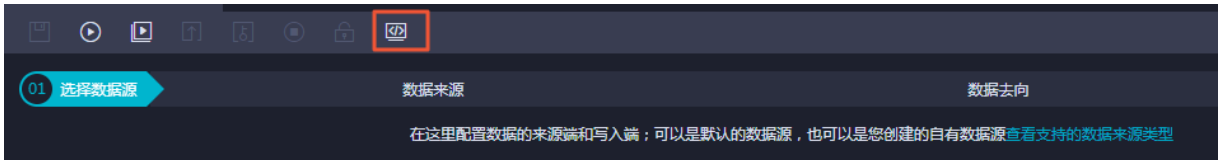


7. 通道控制。

8. 单击工具栏中的**保存**图标。
9. 单击工具栏中的**运行**图标，运行之前需要配置自定义参数。

通过脚本模式配置同步任务

如果您需要通过脚本模式配置此任务，单击工具栏中的**转换脚本**，选择**确认**即可进入脚本模式。



您可以根据自身进行配置，示例脚本如下。

```
{
  "type": "job",
  "version": "1.0",
  "configuration": {
    "reader": {
      "plugin": "otsstream",
      "parameter": {
        "datasource": "otsstream",//数据源名，需要与您添加的数据源名称保持一致。
        "dataTable": "person",//导出增量数据的表的名称。该表需要开启Stream，可以在建表时开启，或者使用UpdateTable接口开启。
        "startTimeString": "${startTime}",//增量数据的时间范围（左闭右开）的左边界，格式为yyyymmddhh24miss，单位毫秒。
        "endTimeString": "${endTime}",//运行时间。
        "statusTable": "TableStoreStreamReaderStatusTable",//用于记录状态的表的名称。
        "maxRetries": 30,//请求的最大重试次数。
        "isExportSequenceInfo": false,
      }
    },
    "writer": {
      "plugin": "odps",
      "parameter": {
        "datasource": "odps_first",//数据源名。
        "table": "person",//目标表名。
        "truncate": true,
        "partition": "pt=${bdp.system.bizdate}",//分区信息。
        "column": [//目标列名。
          "id",
          "colname",
          "version",
          "colvalue",
          "optype",
          "sequenceinfo"
        ]
      }
    },
    "setting": {
      "speed": {
        "mbps": 7,//作业速率上限。
        "concurrent": 7//并发数。
      }
    }
  }
}
```

关于运行时间参数和结束时间参数，有以下两种表现形式（配置任务时选择其中一种）：

- `"startTimeString": "${startTime}"`：增量数据的时间范围（左闭右开）的左边界，格式为yyyymmddhh24miss，单位为毫秒。

  `"endTimeString": "${endTime}"`：增量数据的时间范围（左闭右开）的右边界，格式为yyyymmddhh24miss，单位为毫秒。
- `"startTimestampMillis":""`：增量数据的时间范围（左闭右开）的左边界，单位为毫秒。

Reader插件会从statusTable中找对应startTimestampMillis的位点，从该点开始读取开始导出数据。

如果statusTable中找不到对应的位点，则从系统保留的增量数据的第1条开始读取，并跳过写入时间小于startTimestampMillis的数据。

`"endTimestampMillis":" "`：增量数据的时间范围（左闭右开）的右边界，单位为毫秒。

Reader插件startTimestampMillis位置开始导出数据后，当遇到第1条时间戳大于等于endTimestampMillis的数据时，结束导出数据，导出完成。

当读取完当前全部的增量数据时，结束读取，即使未达endTimestampMillis。

如果配置isExportSequenceInfo项为*true*，如 `"isExportSequenceInfo": true` ，则会导出时序信息，目标会多出1行，目标字段列则多1列。时序信息包含了数据的写入时间等，默认该值为false，即不导出。

1.14. 专有网络VPC的数据源连通独享数据集成资源组

本文以阿里云RDS数据库为例，为您介绍专有网络VPC的数据源如何连通独享数据集成资源组。

前提条件

- 购买RDS MySQL实例。本文以购买MySQL 5.7版本的MySQL实例为例，您可以根据业务需求进行配置。详情请参见创建RDS MySQL实例。
- 在目标实例下，创建数据库和账号。详情请参见创建数据库和账号。
- 创建专有网络VPC，详情请参见创建和管理专有网络。
- 创建交换机，详情请参见创建和管理交换机。
- 创建安全组，详情请参见创建安全组。

背景信息

如果数据源实例无外网地址，且未和数据集成资源组连通专有网络VPC环境，会导致配置数据源时，公共资源组和独享数据集成资源组测试连通性失败。您可以根据本文的操作，实现独享数据集成资源组连通专有网络VPC下的数据源。

操作步骤

1. 获取RDS实例的信息。

   创建MySQL数据源时，您需要获取RDS实例的**实例ID**、**数据库名**、**用户名**和**密码**：

   i. 登录RDS管理控制台。

   ii. 在左侧导航栏，单击**实例列表**。

   iii. 在**实例列表**页面，查看目标实例，并获取**实例ID**。

   iv. 单击目标实例后的**管理**。

   v. 在左侧导航栏，单击**数据库管理**。在显示的页面中，找到您需要建立连接的数据库，获取**数据库名**。

   vi. 在左侧导航栏，单击**账号管理**，获取绑定数据库的**账号**（即创建数据源时需要输入的**用户名**）。

2. 新增独享数据集成资源组。

   i. 登录DataWorks控制台。

   ii. 选择相应地域后，在左侧导航栏，单击**资源组列表**。

iii. 在**独享资源组**页面，单击**创建调度资源组**或**创建集成资源组**。

iv. 在**创建独享资源组**面板中，选择**资源组类型**为**独享数据集成资源组**，单击**订单号**后的**购买**，跳转至购买页面。

v. 进入购买页面后，请根据自身需求，选择相应的**地域**、**独享资源类型**、**独享数据集成资源**、**资源数量**和**计费周期**，单击**立即购买**。

本文以购买华东2（上海）地域的MySQL实例为例，此处的**地域**需要选择**华东2（上海）**。您可以根据自身需求选择**独享数据集成资源**，详情请参见独享数据集成资源组计费说明：包年包月。

vi. 确认订单信息无误后，勾选**《DataWorks独享资源（包年包月）服务协议》**，单击**去支付**。

vii. 返回**新增独享资源组**面板，配置各项参数。

参数	描述
资源组类型	资源的使用类型，此处选择**独享数据集成资源**
资源组名称	资源的名称，租户内唯一，请避免重复。 **说明** 租户即主账号，一个租户（主账号）下可以有多个用户（RAM用户）。
资源备注	对资源进行简单描述。
订单号	此处选择购买的独享资源订单。

viii. 单击**确定**。

3. 独享数据集成资源组绑定专有网络。

i. 在**资源组列表 > 独享资源组**页面，单击独享数据集成资源组后的**专有网络绑定**。

注意 绑定专有网络前，您需要进行RAM授权，让当前操作的账号可以正常访问RDS实例。

ii. 单击右上方的**新增绑定**，在**新增专有网络绑定**面板中，配置各项参数。

参数	描述
资源组名称	从**资源组名称**下拉列表中，单击需要绑定的资源组。
专有网络	从**专有网络**下拉列表中，选择需要同步的数据源所绑定的专有网络。
交换机	从**交换机**下拉列表中，选择数据源所在的交换机。
安全组	从**安全组**下拉列表中，选择相应的安全组。

iii. 单击**创建**。

4. （可选）添加专有网络VPC下的路由。

   独享数据集成资源组所在的可用区必须和数据源所在的可用区一致。如果在同一个可用区，请跳过该步骤。如果不在同一个可用区，请务必添加路由：

   i. 在**独享资源组**页面，单击独享数据集成资源组后的**专有网络绑定**。

   ii. 单击相应资源组后的**自定义路由**。

   iii. 在**自定义路由**面板中，单击**新增路由**。

iv. 在**新增路由**对话框中，配置各项参数。

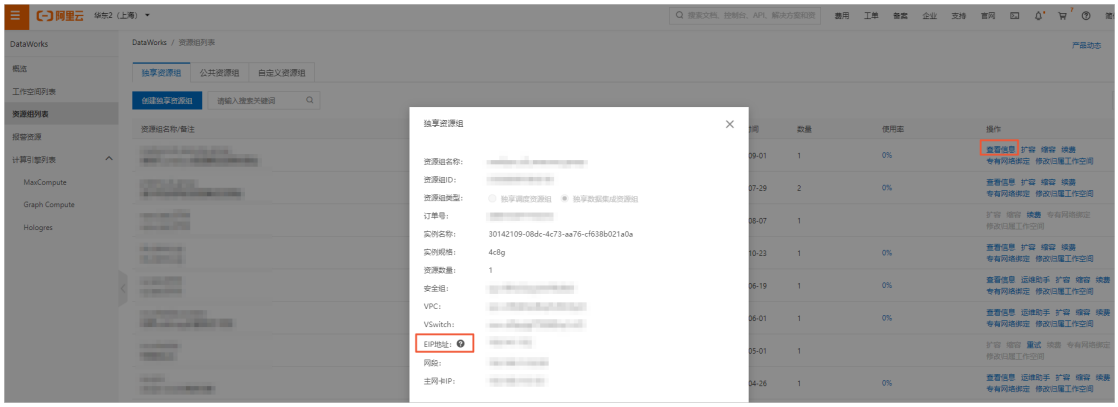
参数	描述
目的类型	本文的数据源在专有网络VPC下，请选择**VPC**。
目的VPC	数据源所在VPC的地域和名称。 **说明** 仅选择**目的类型**为VPC时，会显示该参数。
连接方式	包括**Switch**、**固定IP**和**指定网段**，本文需要选中**Switch**。
目的Switch实例	选择该VPC下的某一个交换机（Switch）作为路由目标。
路由示意图	默认不可以修改。

v. 单击**生成路由**。

5. 添加独享数据集成资源组的EIP和弹性网卡IP至RDS白名单中。

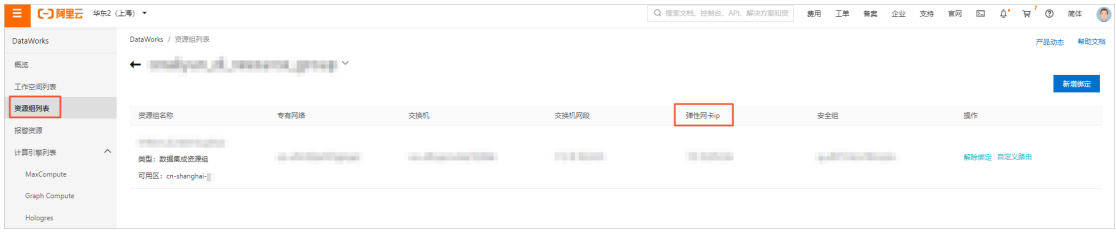
i. 在**独享资源组**页面，单击独享数据集成资源组后的**查看信息**。

ii. 在**独享资源组**对话框中，查看**EIP地址**。



iii. 在**资源组列表 > 独享资源组**页面，单击独享数据集成资源组后的**专有网络绑定**。

iv. 在资源组的详情页面，查看**弹性网卡ip**。



v. 登录RDS管理控制台。

vi. 在**实例列表**页面的左上方查看是否为目标实例所在地域。如果不是，请切换地域后，单击相应实例后的**管理**。

vii. 在左侧导航栏，单击**数据安全性**。

viii. 在**数据安全性**页面，单击**添加白名单分组**。

ix. 在**添加白名单分组**对话框中，配置各项参数，单击**添加**。

参数	描述
网络隔离模式	默认不可以修改。
分组名称	分组的名称需要符合以下规范： ■ 由小写字母、数字和下划线（_）组成。 ■ 以小写字母开头，以字母或数字结尾。 ■ 长度为2~32个字符。
组内白名单	复制独享数据集成资源组的**EIP地址**和**弹性网卡ip**。

6. 创建MySQL数据源并测试连通性。

i. 登录DataWorks控制台。

ii. 在左侧导航栏，单击**工作空间列表**。

iii. 选择工作空间所在地域后，单击相应工作空间后的**进入数据集成**。

iv. 在左侧导航栏，单击**数据源**，进入**数据源管理**页面。

v. 在**数据源管理**页面，单击右上角的**新增数据源**。

vi. 在**新增数据源**对话框中，选择数据源类型为**MySQL**。

vii. 在**新增MySQL数据源**对话框中，配置各项参数。

MySQL数据源包括**阿里云实例模式**和**连接串模式**两种类型，本文以选择**阿里云实例模式**为例。

参数	描述
数据源类型	当前选择的数据源类型为**阿里云实例模式**。
数据源名称	数据源名称必须以字母、数字、下划线（_）组合，且不能以数字和下划线（_）开头。
数据源描述	对数据源进行简单描述，不得超过80个字符。
适用环境	可以选择**开发**或**生产**环境。 **说明** 仅标准模式工作空间会显示该配置。
地区	选择购买RDS所在的地域。
RDS实例ID	输入获取的RDS实例ID。
RDS实例主账号ID	实例购买者登录DataWorks控制台，鼠标悬停至右上角的用户头像，单击**安全设置**，查看账号ID。
数据库名	输入获取的数据库名称。
用户名	输入获取的绑定该数据库的用户名。
密码	输入绑定用户的密码。

viii. 在**数据集成**页签下，单击相应资源组后的**测试连通性**。

ix. 测试连通性通过后，单击**完成**。

1.15. MySQL数据源转PolarDB数据源的OpenAPI最佳实践

本文为您介绍MySQL升级为PolarDB后，如何将之前从MySQL同步到MaxCompute的任务批量修改为从PolarDB同步到MaxCompute。

POM 依赖示例代码

```
<dependency>
        <groupId>com.aliyun</groupId>
        <artifactId>aliyun-java-sdk-core</artifactId>
        <version>4.5.20</version>
</dependency>
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-dataworks-public</artifactId>
    <version>3.4.4</version>
 </dependency>
```

Java Sdk调用示例代码

```
package com.alibaba.eas;
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.dataworks_public.model.v20200518.*;
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.profile.IClientProfile;
import java.util.List;
public class updateOfflineTask {
    public static ListFilesResponse.Data.File ListFiles(String filePath, String fileName) t
hrows Exception {
        ListFilesRequest request = new ListFilesRequest();
        request.setProjectId(1911L);
        request.setFileFolderPath(filePath);
        request.setKeyword(fileName);
        ListFilesResponse response1 = client.getAcsResponse(request);
        for(int i = 0 ; i < response1.getData().getFiles().size(); ++i) {
            return response1.getData().getFiles().get(i);
        }
        return null;
    }
    public static String GetFiles(Long fileId) throws Exception {
        GetFileRequest request = new GetFileRequest();
        request.setProjectId(1911L);
        request.setFileId(fileId);
        GetFileResponse response1 = client.getAcsResponse(request);
        return response1.getData().getFile().getContent();
    }
    public static void UpdateDISyncTask(Long fileId, String content) throws Exception {
        UpdateDISyncTaskRequest request = new UpdateDISyncTaskRequest();
        request.setProjectId(1911L);
        request.setFileId(fileId);
        request.setTaskContent(content);
        request.setTaskType("DI_OFFLINE");
        UpdateDISyncTaskResponse response1 = client.getAcsResponse(request);
    }
    public static  Long submitFile(Long fileId) throws  Exception {
        SubmitFileRequest request = new SubmitFileRequest();
```

```
        request.setProjectId(1911L);
        request.setFileId(fileId);
        SubmitFileResponse acsResponse = client.getAcsResponse(request);
        Long deploymentId = acsResponse.getData();
        return deploymentId;
    }
    public static  void getDeployment(Long deploymentId) throws Exception {
        GetDeploymentRequest request = new GetDeploymentRequest();
        request.setProjectId(1911L);
        request.setDeploymentId(deploymentId);
        GetDeploymentResponse acsResponse = client.getAcsResponse(request);
        System.out.println(acsResponse.getData().getDeployment().getStatus());
    }
    public static  Long deploy(Long fileId) throws Exception {
        DeployFileRequest request = new DeployFileRequest();
        request.setProjectId(1911L);
        request.setFileId(fileId);
        DeployFileResponse acsResponse = client.getAcsResponse(request);
        Long deploymentId = acsResponse.getData();
        return deploymentId;
    }
    public static Long listNode(String nodeName) throws Exception {
        ListNodesRequest request = new ListNodesRequest();
        request.setProjectId(1911L);
        request.setNodeName(nodeName);
        request.setProjectEnv("PROD");
        ListNodesResponse acsResponse = client.getAcsResponse(request);
        List<ListNodesResponse.Data.NodesItem> nodesItemList = acsResponse.getData().getNod
es();
        return nodesItemList.get(0).getNodeId();
    }
    public static void RunCycleDagNodes(Long nodeId) throws Exception {
        RunCycleDagNodesRequest request = new RunCycleDagNodesRequest();
        request.setIncludeNodeIds(nodeId.toString());
        request.setName("rerun_job");
        request.setParallelism(false);
        request.setProjectEnv("PROD");
        request.setRootNodeId(nodeId);
        request.setStartBizDate("2021-09-29 00:00:00");
        request.setEndBizDate("2021-09-29 00:00:00");
        request.setProjectEnv("PROD");
        RunCycleDagNodesResponse acsResponse = client.getAcsResponse(request);
    }
  /*
  把下面的配置中的 mysql 和 mysql_from_polardb 进行替换，
  {
    "type": "job",
    "version": "2.0",
    "steps": [
        {
            "stepType": "mysql",
            "parameter": {
                "envType": 0,
                "datasource": "mysql_from_polardb",
```

```
                "column": [
                    "id",
                    "name",
                    "create_time",
                    "create_user"
                ],
                "tableComment": "配置表",
                "connection": [
                    {
                        "selectedDatabase": "polardb_db1",
                        "datasource": "mysql_from_polardb",
                        "table": [
                            "lcl_test_demo"
                        ]
                    }
                ],
                "where": "",
                "splitPk": "id",
                "encoding": "UTF-8"
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "odps",
            "parameter": {
                "partition": "pt=${bizdate}",
                "truncate": true,
                "datasource": "odps_first",
                "envType": 0,
                "column": [
                    "id",
                    "name",
                    "create_time",
                    "create_user"
                ],
                "emptyAsNull": false,
                "tableComment": "配置表",
                "table": "lcl_test_demo"
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "setting": {
        "errorLimit": {
            "record": ""
        },
        "locale": "zh_CN",
        "speed": {
            "throttle": false,
            "concurrent": 2
        }
    },
```

```
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```

替换完之后的结果是

```
  {
    "type": "job",
    "version": "2.0",
    "steps": [
        {
            "stepType": "polardb",
            "parameter": {
                "envType": 0,
                "datasource": "polardb",
                "column": [
                    "id",
                    "name",
                    "create_time",
                    "create_user"
                ],
                "tableComment": "配置表",
                "connection": [
                    {
                        "selectedDatabase": "polardb_db1",
                        "datasource": "polardb",
                        "table": [
                            "lcl_test_demo"
                        ]
                    }
                ],
                "where": "",
                "splitPk": "id",
                "encoding": "UTF-8"
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "odps",
            "parameter": {
                "partition": "pt=${bizdate}",
                "truncate": true,
                "datasource": "odps_first",
                "envType": 0,
                "column": [
                    "id",
                    "name",
                    "create_time",
                    "create_user"
                ],
```

```
                "emptyAsNull": false,
                "tableComment": "配置表",
                "table": "lcl_test_demo"
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "setting": {
        "errorLimit": {
            "record": ""
        },
        "locale": "zh_CN",
        "speed": {
            "throttle": false,
            "concurrent": 2
        }
    },
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
  */
    public static String modifyContent(String content, String newStepType, String newDataso
urce){
        JSONObject jsonObject = JSON.parseObject(content);
        JSONArray steps = jsonObject.getJSONArray("steps");
        if (steps != null) {
            for (int i = 0; i < steps.size(); ++i) {
                JSONObject step = steps.getJSONObject(i);
                if (step != null && step.getString("category") != null && "reader".equals(s
tep.getString("category"))) {
                    if (step.getString("stepType") != null && "mysql".equals(step.getString
("stepType"))) {
                        step.put("stepType", newStepType);
                        JSONObject parameter = step.getJSONObject("parameter");
                        if (parameter  != null) {
                            parameter.put("datasource", newDatasource);
                            JSONArray connections = parameter.getJSONArray("connection");
                            if (connections != null) {
                                for (int j = 0; j < connections.size(); ++j) {
                                    JSONObject connection = connections.getJSONObject(j);
                                    if (connection != null) {
                                        connection.put("datasource", newDatasource);
                                    }
                                }
                            }
                        }
                    }
```

```
                }
            }
        }
        return  jsonObject.toJSONString();
    }
    static IAcsClient client;
    public static void main(String[] args) throws Exception {
        String akId = "XXX";
        String akSecret = "XXX";
        String regionId = "cn-chengdu";
        IClientProfile profile = DefaultProfile.getProfile(regionId, akId, akSecret);
        DefaultProfile.addEndpoint(regionId, "dataworks-public", "dataworks." + regionId +
".aliyuncs.com");
        client = new DefaultAcsClient(profile);
        String folderPath = "业务流程/重兴/数据集成/";
        String filename = "mysql_to_odps";
        ListFilesResponse.Data.File file = ListFiles(folderPath, filename);
        Long fileId = file.getFileId();
        System.out.println(file.getFileId());
        String content = GetFiles(fileId);
        String contentModified = modifyContent(content, "polardb", "polardb_datasource");
        //  "polardb" 代表替换替换的目标数据源，这里的实例是把 mysql 替换成 polardb
        //
        UpdateDISyncTask(file.getFileId(), contentModified);
        Long deployId = submitFile(fileId);
        getDeployment(deployId);
        Thread.sleep(10000);
        getDeployment(deployId);
        deployId = deploy(fileId);
        getDeployment(deployId);
        Thread.sleep(10000);
        getDeployment(deployId);
        Long nodeId = listNode(filename);
        RunCycleDagNodes(nodeId);
    }
}
```

2.数据开发

2.1. 设置调度依赖最佳实践

配置调度依赖时，需要根据本节点输出名称作为关联项来给任务间设置依赖关系。本文将为您介绍如何配置任务调度依赖的输入输出。

配置任务的本节点输入

您可以通过以下两种方式配置本节点输入：

- 使用代码自动解析功能，解析出任务的依赖。
- 手动输入任务依赖（手动输入父节点的**本节点的输出名称**）。

> **说明** 手动输入上游节点时，输入的是父节点的**本节点的输出名称**。如果父节点的任务名称和父节点的本节点输出名称不一致，请务必正确输入本节点输出名称。

配置上游节点时，如果自动解析出来的上游节点是一个无效的上游依赖。您可以查看解析出来的上游依赖在**父节点ID**一列是否显示有值，来识别设置的依赖是否有效。

配置任务依赖实质上是给两个节点设置依赖关系。只有真实存在的节点，才能够设置有效的依赖关系。

无效的上游依赖

无效的上游依赖通常有以下两种情况：

- 父节点不存在。

- 父节点输出不存在。

通常由于解析出来的父节点输出名称不存在，会导致上游依赖无效。本示例中，由于表project_b_name.pm_table_b没有产出任务，或该表产出任务的本节点输出的配置不正确，导致无法解析出来。

您可以通过以下两种方案解决该问题：

- 确认该表是否有产出任务。
- 确认该表产出任务的本节点输出名称，将该本节点的输出名称手动输入到依赖的上游节点中。

> **说明** 手动输入上游节点时，输入的是父节点的**本节点的输出名称**。如果父节点的任务名称和父节点的本节点输出名称不一致，请务必正确输入本节点输出名称。
>
> 例如，上游节点A的本节点输出名称是A1，下游节点B需要依赖A，此时应该在依赖上游节点的输入框中输入A1，并单击右侧的加号进行添加。

配置上游依赖

如果您的表不存在上游，您可以通过单击**使用工作空间根节点**获得上游依赖。

配置任务的本节点输出

本节点名称、本节点输出名称和本节点输出表名共用同一个名称，可以高效配置本节点输出。

- 能够快速的知道该任务操作的是哪个表。
- 能够快速知道该任务失败后造成的影响范围有多大。
- 使用自动解析配置任务依赖时，只要本节点输出符合三名合一的规则，即可提升自动解析的精准性能。

自动解析

自动解析：通过代码自动解析调度依赖关系。

自动解析的实现原理：代码中只能拿到表名，自动解析功能可以根据表名解析出对应的产出任务。

类型节点代码如下所示。

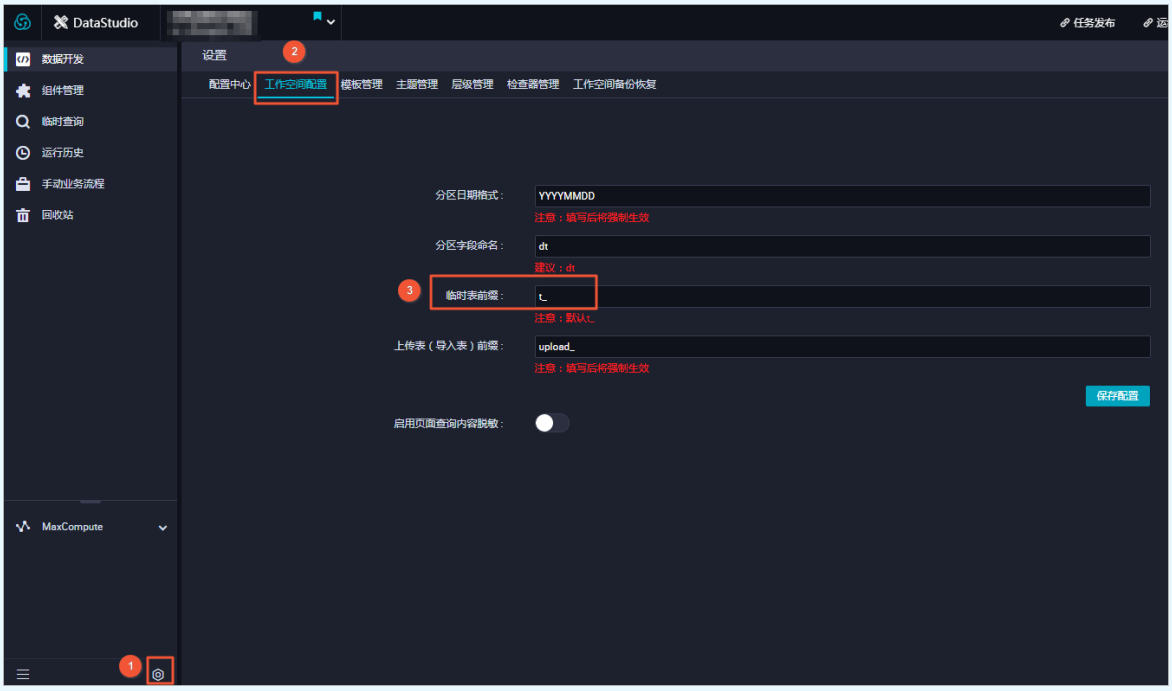
```
INSERT OVERWRITE TABLE pm_table_a SELECT * FROM project_b_name.pm_table_b ;
```

解析出来的依赖关系如下。

DataWorks会自动解析本节点，即依赖 `project_b_name` 产出 `pm_table_b` 的节点，同时本节点最终产出 `pm_table_a` ，因此父节点输出名称为 `project_b_name.pm_table_b` ，本节点的输出名称为 `project_name.pm_table_a` （本工作空间名称为test_pm_01）。

- 如果您不想使用从代码解析到的依赖，则选择**否**。
- 如果代码中有很多表是临时表：如t_开头的表为临时表。则该表不会被解析为调度依赖。可以通过项目配置，定义以什么形式开头的表为临时表。
- 如果代码中的一个表，既是产出表又是被引用表（被依赖表），则解析时只解析为产出表。
- 如果代码中的一个表，被多次引用或者被多次产出，则解析时只解析一个调度依赖关系。

> **说明** 默认情况下，表名为t_开头的会被当成临时表，自动解析不解析临时表。如果t_开头不是临时表，请联系自己的项目管理员到**工作空间配置**页面进行修改。

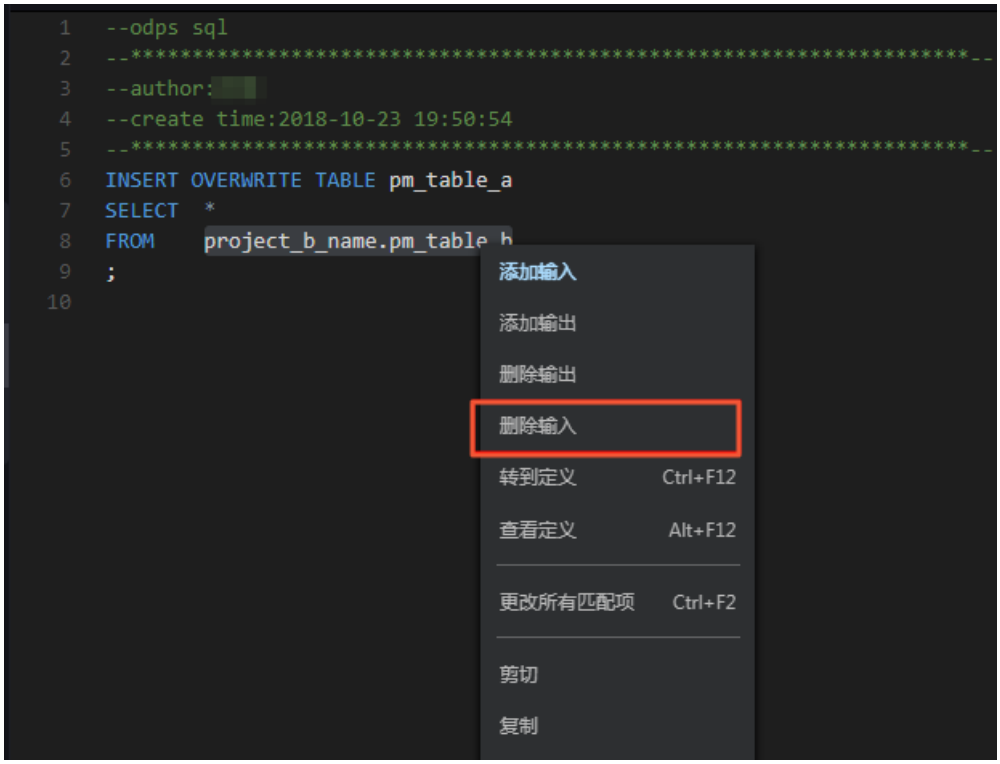


删除某表的输入输出

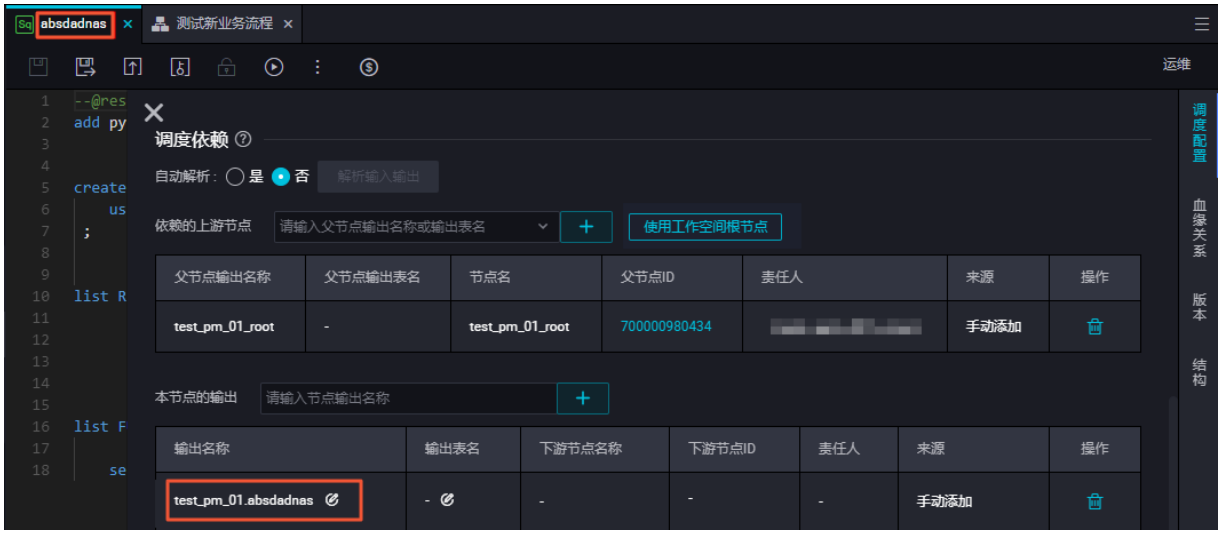
您在进行数据开发时，经常会用到静态表（数据通过本地文件上传到表中），这部分静态数据没有产出任务。

配置依赖时，您需要删除静态表的输入：如果静态表不满足t_的格式，不会被处理为临时表，此时您需要删除静态表的输入。

您在代码中右键单击表名，选择**删除输入**。



如果您是从DataWorksV1.0升级至DataWorks V2.0的用户，则您迁移过来的DataWorks任务的本节点输出默认为 `工作空间名.节点名` 。



注意事项

当任务依赖配置完成后，提交的窗口会有一个选项：当输入输出和代码血缘分析不匹配时，是否确认继续执行提交操作。

该选项的前提是您已经确认依赖关系正确。如果不能确认，则可以按照上述方法确认依赖关系。

确认依赖关系无误后，直接勾选**我确认继续执行提交操作**，并单击**确认**。

2.2. 使用MaxCompute分析IP来源最佳实践

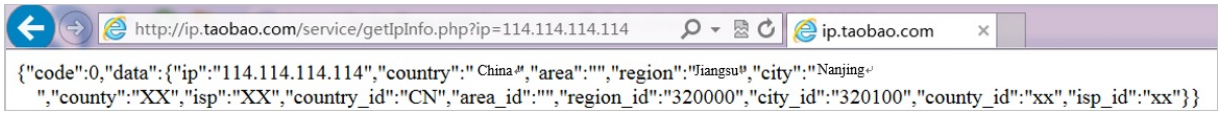
本文为您介绍如何使用MaxCompute分析IP来源，包括下载、上传IP地址库数据、编写UDF函数和编写SQL四个步骤。

前提条件

- 开通MaxCompute和DataWorks。
- 开通DataWorks。
- 在DataWorks上完成创建业务流程，本例使用DataWorks简单模式。详情请参见创建业务流程。

背景信息

淘宝IP地址库的查询接口为IP地址字串，使用示例如下。



在MaxCompute中禁止使用HTTP请求，因此目前可以通过如下三种方式实现在MaxCompute中查询IP：

- 用SQL将数据下载至本地，再发起HTTP请求查询。

  > **说明** 效率低下，且淘宝IP库查询频率需要小于10 QPS，否则拒绝请求。

- 下载IP地址库至本地，再进行查询。

> **说明** 效率低下，且不利于数据仓库等产品分析使用。

- 将IP地址库定期维护上传至MaxCompute，进行连接查询。

> **说明** 比较高效，但是IP地址库需要自己定期维护。

下载IP地址库数据

1. 获取地址库数据。本文提供示例IP地址库数据UTF-8格式的不完整的地址库demo。
2. 下载示例地址库数据至本地，示例如下。

```
0,16777215,"0.0.0.0","0.255.255.255","","","内网IP","内网IP","内网IP"
16777216,16777471,"1.0.0.0","1.0.0.255","澳大利亚","","","",""
16777472,16778239,"1.0.1.0","1.0.3.255","中国","福建省","福州市","","电信"
```

示例数据说明如下：

- 数据格式为UTF-8。
- 前四个数据是IP地址的起始地址与结束地址。前两个是十进制整数形式，后两个是点分形式。IP地址段为整数形式，以便计算IP是否属于这个网段。

> **说明** 如果您需要使用自己的IP地址，请自行下载IP地址库，具体的下载地址和使用方式请参见MaxCompute中实现IP地址归属地转换。

上传IP地址库数据

1. 在MaxCompute客户端执行如下语句，创建表ipresource存放IP地址库数据。

```
DROP TABLE IF EXISTS ipresource ;
CREATE TABLE IF NOT EXISTS ipresource
(
    start_ip BIGINT
    ,end_ip BIGINT
    ,start_ip_arg string
    ,end_ip_arg string
    ,country STRING
    ,area STRING
    ,city STRING
    ,county STRING
    ,isp STRING
);
```

2. 执行如下Tunnel命令，上传本地示例IP地址库数据至表ipresource。

```
odps@ workshop_demo>tunnel upload D:/ipdata.txt.utf8 ipresource;
```

上述命令中，*D:/ipdata.txt.utf8*为IP地址库数据本地存放路径。更多命令说明请参见Tunnel命令。

您可以执行如下语句验证数据是否上传成功。

```
--查询表中数据条数。
select count(*) from ipresource;
```

3. 执行如下SQL语句，查看表ipresource前10条的样本数据。

```
select * from ipresource limit 10;
```

返回结果如下。

```
Job Queueing...
+------------+------------+--------------+------------+---------+------+------+--------+-----+
| start_ip   | end_ip     | start_ip_arg | end_ip_arg | country | area | city | county | isp |
+------------+------------+--------------+------------+---------+------+------+--------+-----+
| 3395369026 | 3395369026 | "202.97.56.66" | "202.97.56.66" | "中国" | "湖南省" | "长沙市" | "" | "电信" |
| 3395369027 | 3395369028 | "202.97.56.67" | "202.97.56.68" | "中国" | "黑龙江省" | "" | "" | "电信" |
| 3395369029 | 3395369029 | "202.97.56.69" | "202.97.56.69" | "中国" | "安徽省" | "合肥市" | "" | "电信" |
| 3395369030 | 3395369030 | "202.97.56.70" | "202.97.56.70" | "中国" | "湖南省" | "长沙市" | "" | "电信" |
| 3395369031 | 3395369033 | "202.97.56.71" | "202.97.56.73" | "中国" | "黑龙江省" | "" | "" | "电信" |
| 3395369034 | 3395369034 | "202.97.56.74" | "202.97.56.74" | "中国" | "湖南省" | "长沙市" | "" | "电信" |
| 3395369035 | 3395369036 | "202.97.56.75" | "202.97.56.76" | "中国" | "黑龙江省" | "" | "" | "电信" |
| 3395369037 | 3395369037 | "202.97.56.77" | "202.97.56.77" | "中国" | "江苏省" | "南京市" | "" | "电信" |
| 3395369038 | 3395369038 | "202.97.56.78" | "202.97.56.78" | "中国" | "湖南省" | "长沙市" | "" | "电信" |
| 3395369039 | 3395369040 | "202.97.56.79" | "202.97.56.80" | "中国" | "黑龙江省" | "" | "" | "电信" |
+------------+------------+--------------+------------+---------+------+------+--------+-----+
```

编写UDF函数

通过编写Python UDF，将点号分割的IP地址转化为整数类型的IP地址，本示例使用DataWorks的PyODPS完成。详情请参见创建PyODPS 2节点。

1. 进入**数据开发**页面。
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**工作空间列表**。
   iii. 单击相应工作空间后的**数据开发**。
2. 新建Python资源。
   i. 右键单击业务流程，选择**新建 > MaxCompute > 资源 > Python**。
   ii. 在**新建资源**对话框中，填写**资源名称**，并勾选**上传为ODPS资源**，单击**确定**。
   iii. 在Python资源中输入如下代码。

```
from odps.udf import annotate
@annotate("string->bigint")
class ipint(object):
    def evaluate(self, ip):
        try:
            return reduce(lambda x, y: (x << 8) + y, map(int, ip.split('.')))
        except:
            return 0
```

   iv. 单击**提交**。
3. 新建函数。
   i. 右键单击已创建的业务流程，选择**新建 > MaxCompute > 函数**。
   ii. 在**新建函数**对话框中，输入**函数名称**，单击**提交**。

> **说明** 如果绑定了多个MaxCompute引擎，则需要选择MaxCompute引擎实例。

   iii. 在函数的编辑页面，配置各项参数。

参数	描述
函数类型	选择函数类型，包括**数学运算函数**、**聚合函数**、**字符串处理函数**、**日期函数**、**窗口函数**和**其他函数**。
MaxCompute引擎实例	默认不可以修改。
函数名	UDF函数名，即SQL中引用该函数所使用的名称。需要全局唯一，且注册函数后不支持修改。
责任人	默认显示。
类名	实现UDF的主类名，必填。 **说明** 当资源类型为Python时，类名格式为Python资源名称.类名（资源名称中的.py无需填写）。
资源列表	完整的文件名称，支持模糊匹配查找本工作空间中已添加的资源，必填。 多个文件之间，使用英文逗号（,）分隔。
描述	针对当前UDF作用的简单描述。
命令格式	该UDF的具体使用方法示例，例如`test`。
参数说明	支持输入的参数类型以及返回参数类型的具体说明。

参数	描述
返回值	返回值，例如1，非必填项。
示例	函数中的示例，非必填项。

4. 单击工具栏中的图标。
5. 提交函数。
   i. 单击工具栏中的图标。
   ii. 在**提交新版本**对话框中，输入**备注**。
   iii. 单击**确认**。

在SQL中使用UDF函数分析IP来源

1. 右键单击业务流程，选择**新建 > MaxCompute > ODPS SQL**。
2. 在**新建节点**对话框中输入**节点名称**，并单击**提交**。
3. 在ODPS SQL节点编辑页面，输入如下语句。

```
select * from ipresource
WHERE ipint('1.2.24.2') >= start_ip
AND ipint('1.2.24.2') <= end_ip
```

4. 单击图标运行代码。
5. 您可以在**运行日志**查看运行结果。

2.3. 在PyODPS节点中调用第三方包

本文为您介绍在依赖普通的Python脚本和开源三方包的场景下，如何使用DataWorks PyODPS节点调用第三方包。

依赖普通的Python脚本

1. 进入**数据开发**页面。
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**工作空间列表**。
   iii. 选择工作空间所在地域后，单击相应工作空间后的**进入数据开发**。
2. 创建Python资源。
   i. 在DataStudio（**数据开发**）页面，鼠标悬停至+新建图标，单击**MaxCompute > 资源 > Python**。
   您也可以展开**业务流程**目录下的目标业务流程，右键单击**MaxCompute**，选择**新建 > 资源 > Python**。

ii. 在**新建资源**对话框中，输入**资源名称**（示例为pyodps_packagetest.py），并选择**目标文件夹**。

> **注意** 资源名称只能包含中文、字母、数字、点、下划线（_）、减号（-），且必须加后缀名.py。

iii. 单击**确定**。

iv. 在新建的Python资源中，输入需要引用的第三方包的代码，示例如下。

```
# import os
# print os.getcwd()
# print os.path.abspath('.')
# print os.path.abspath('..')
# print os.path.abspath(os.curdir)
def printname():
    print 'test2'
print 123
```

v. 单击工具栏中的图标。

3. 创建PyODPS 2节点。

i. 展开**业务流程**目录下的目标业务流程，右键单击**MaxCompute**，选择**新建 > PyODPS 2**。

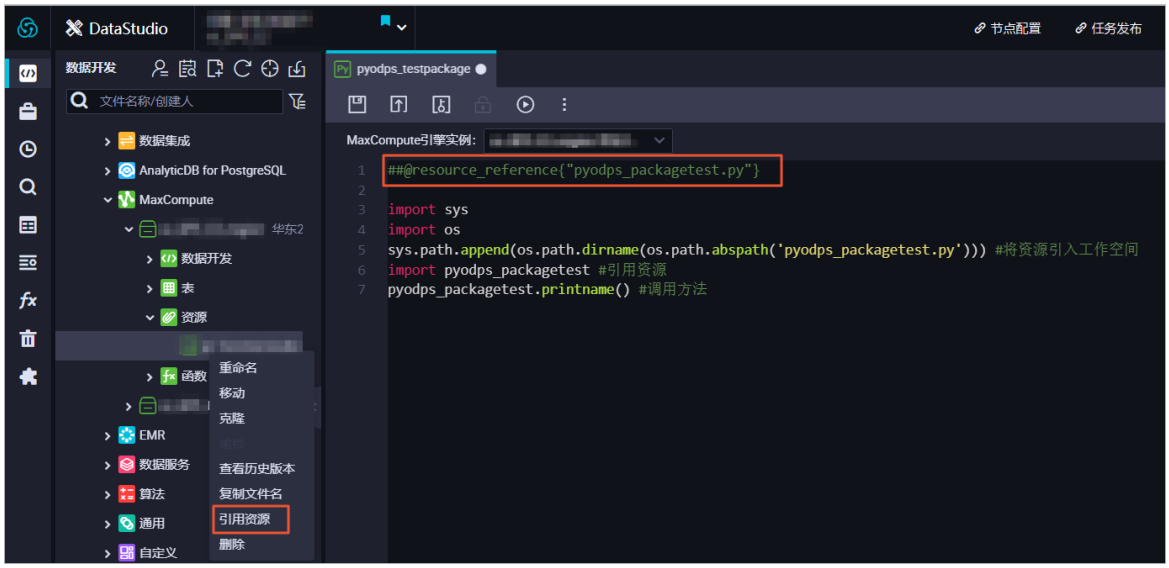
ii. 在**新建节点**对话框中，输入**节点名称**（示例为pyodps_testpackage），并选择**目标文件夹**。

> **说明** 节点名称必须是大小写字母、中文、数字、下划线（_）和小数点（.），且不能超过128个字符。

iii. 单击**提交**。

4. 打开PyODPS 2节点的编辑页面，右键单击目标Python资源名称，选择**引用资源**。

引用后，PyODPS 2节点中会自动写入引用语句 `##@resource_reference{"pyodps_packagetest.py"}` 。



5. 在PyODPS 2节点内输入引用第三方包的代码，示例如下。

```
##@resource_reference{"pyodps_packagetest.py"} #用于引用之前新建的Python资源，该语句必须添加。
import sys
import os
sys.path.append(os.path.dirname(os.path.abspath('pyodps_packagetest.py'))) #引入资源至工作空间。
import pyodps_packagetest #引用资源，资源名需要删除后缀.py。
pyodps_packagetest.printname() #调用方法。
```

6. 单击工具栏中的图标，在页面下方的**运行日志**区域查看结果。

依赖开源的三方包

如果您依赖一个开源的三方包，需要使用PIP安装，且需要满足以下条件：

- 必须使用独享调度资源组，详情请参见购买资源组（创建订单）。
- 在独享调度资源组的运维助手中安装需要的三方包，详情请参见运维助手。PyODPS节点分为PyODPS 2和PyODPS 3：
  - 如果依赖PyODPS 2节点，请执行如下命令。

    ```
    pip install <需要安装的包> -i  https://pypi.tuna.tsinghua.edu.cn/simple
    ```

    执行命令后，如果提示需要升级PIP版本，请执行如下命令。

    ```
    pip install --upgrade pip -i https://pypi.tuna.tsinghua.edu.cn/simple
    ```

  - 如果依赖PyODPS 3节点，请执行如下命令。

    ```
    /home/tops/bin/pip3 install <需要安装的包> -i https://pypi.tuna.tsinghua.edu.cn/simple
    ```

    安装了需要的三方包后，使用 `import` 命令导入对应包即可使用。例如，通过运维助手使用 `pip3 -install oss2` 语句安装了oss2这个Python依赖包后，您可以在PyODPS 3节点中使用 `import oss2` 语句导入oss2依赖包并使用。

    执行命令后，如果提示需要升级PIP版本，请执行如下命令。

    ```
    /home/tops/bin/pip3 install --upgrade pip -i https://pypi.tuna.tsinghua.edu.cn/simple
    ```

    如果使用PyODPS 3出现如下报错，请提交工单申请开启权限。

    ```
    "/home/admin/usertools/tools/cmd-0.sh:行3: /home/tops/bin/python3: 没有那个文件或目录"
    ```

2.4. 分支节点实现特定时间执行任务

本文为您介绍分支节点如何实现在特定时间执行任务。

分支节点产生背景

Cron表达式无法实现一个节点需要每个月的最后一天执行的场景，分支节点产生后，您可以套用switch-case编程模型实现该需求。详情请参见分支节点。

分支节点与其它控制节点

在**数据开发**页面，您可以看到当前版本的DataWorks支持的赋值节点、分支节点和归并节点等控制节点。

各类型控制节点的作用如下：

- **赋值节点**：您可以通过赋值节点把自己的结果传给下游，详情请参见赋值节点。

  赋值节点复用了配置节点上下文依赖的特性，在已有常量和变量节点上下文的基础上，赋值节点支持自定义的上下文输出。DataWorks会捕获或打印赋值节点的查看结果，并将该结果以outputs形式作为上下文输出参数的值，供下游节点引用。

- **分支节点**：可以决定哪些下游正常执行，详情请参见分支节点。

  分支节点复用了DataWorks上依赖关系设置的输入和输出的特性，详情请参见配置同周期调度依赖。

  对于普通节点，节点的输出仅仅是一个全局唯一的字符串。当下游需要设置依赖时，搜索这个全局唯一的字符串作为节点的输入即可挂到下游节点列表中。

  但是，对于分支节点您可以在设置下游依赖时，选择某一个条件关联的输出作为分支节点的输出。节点在成为分支节点下游的同时，也关联到了分支节点的条件上：

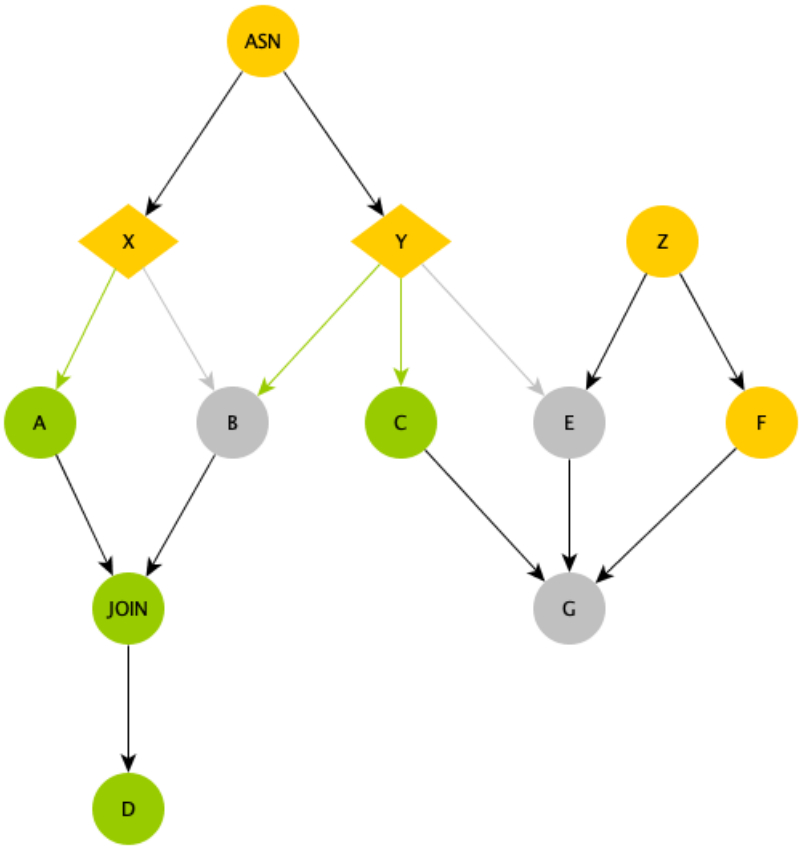
  - 满足该条件，该输出对应的下游才会被正常执行。
  - 其它未满足条件的输出对应的下游节点，会被置为空跑。

- **归并节点**：无论上游是否正常执行，本身都会正常调度。

  未被分支节点选中的分支，DataWorks会把该分支链路上所有的节点实例置为空跑实例，即一旦某个实例的上游有一个空跑实例，它本身也会变为空跑。

  DataWorks当前可以通过归并节点来阻止该空跑的属性无限制地传递下去：无论归并节点实例上游有多少个空跑的实例，归并节点都会直接成功，且不会再把下游置为空跑。

下图为存在分支节点的情况下，依赖树的逻辑关系。



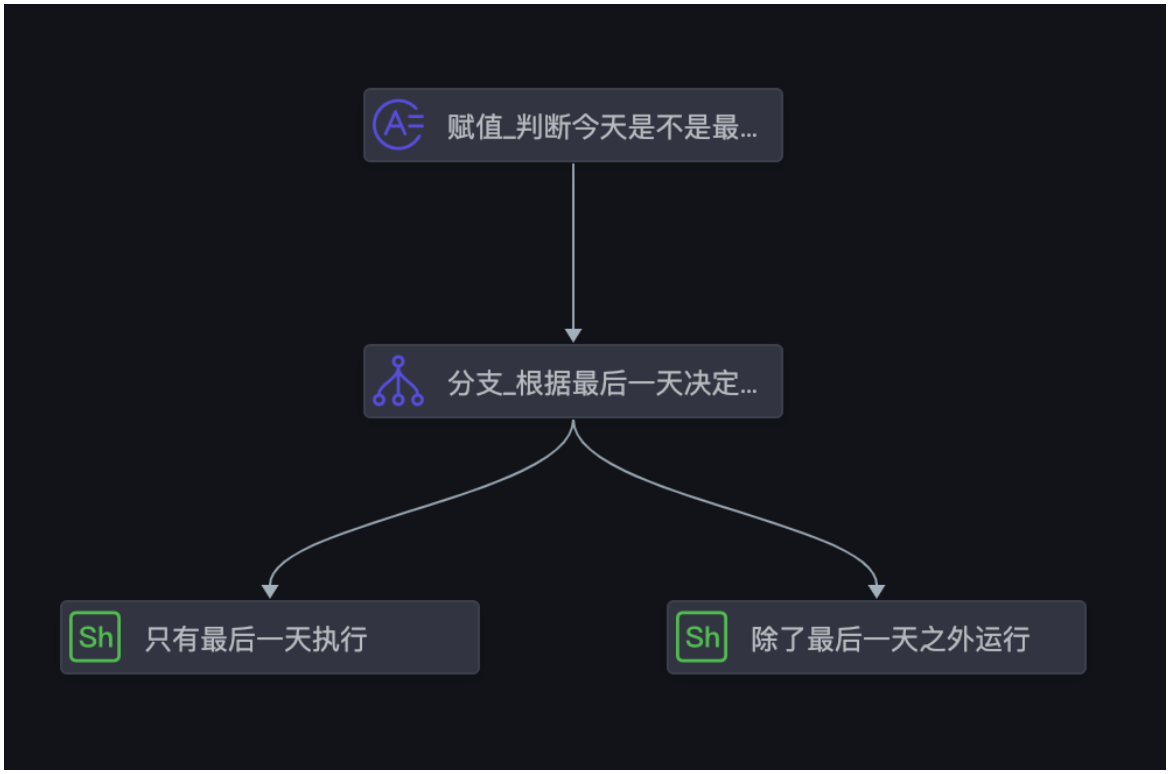
- ASN：一个赋值节点，用于对比较复杂的情况进行计算，以便选择分支节点的条件。
- X和Y：分支节点，处于赋值节点ASN下游，根据赋值节点的输出进行分支的选择。如图中绿色线条所示，X节点选择了左边的分支，Y节点选择了左边两个分支：
  - A和C节点由于在X和Y节点被选择的输出下游，因此正常执行。
  - B节点虽然在Y节点被选择的分支下游，但由于X节点未选择该输出，因此B节点被置为空跑。
  - E节点由于未被Y节点选中，因此即使有一个普通的Z节点上游，也同样被置为了空跑。
  - G节点由于上游E节点空跑，因此即使C/F都正常执行，G节点同样空跑。
  - 空跑属性什么情况下才能不再向下传递？

    JOIN节点是一个归并节点，它的特殊功能就是停止空跑属性的传递，可以看到由于D节点处于JOIN节点下游，因此B节点的空跑属性被阻断了，D节点可以开始正常跑了。

您可以通过利用分支节点配合其它控制节点，满足某个节点只有每个月最后一天运行的需求场景。

使用分支节点

1. 定义任务依赖。



i. 根节点赋值节点通过定时时间SKYNET_CYCTIME来计算当前是否为本月的最后一天。如果是，则输出1。如果不是，则输出0。该输出会被DataWorks捕获，传递给下游。

ii. 分支节点通过赋值节点的输出来定义分支。

iii. 两个Shell节点挂在分支节点下，分别执行不同的分支逻辑。

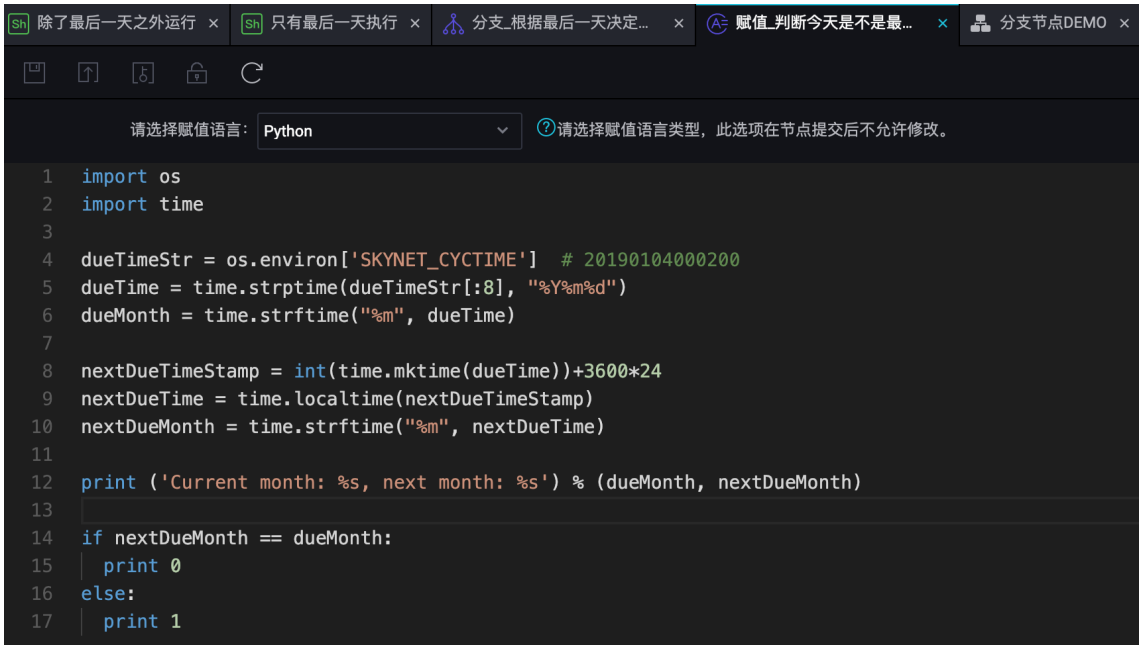
2. 定义赋值节点。

新建赋值节点时，会自带一个outputs，赋值节点支持编写SQL、SHELL和Python三种语言。

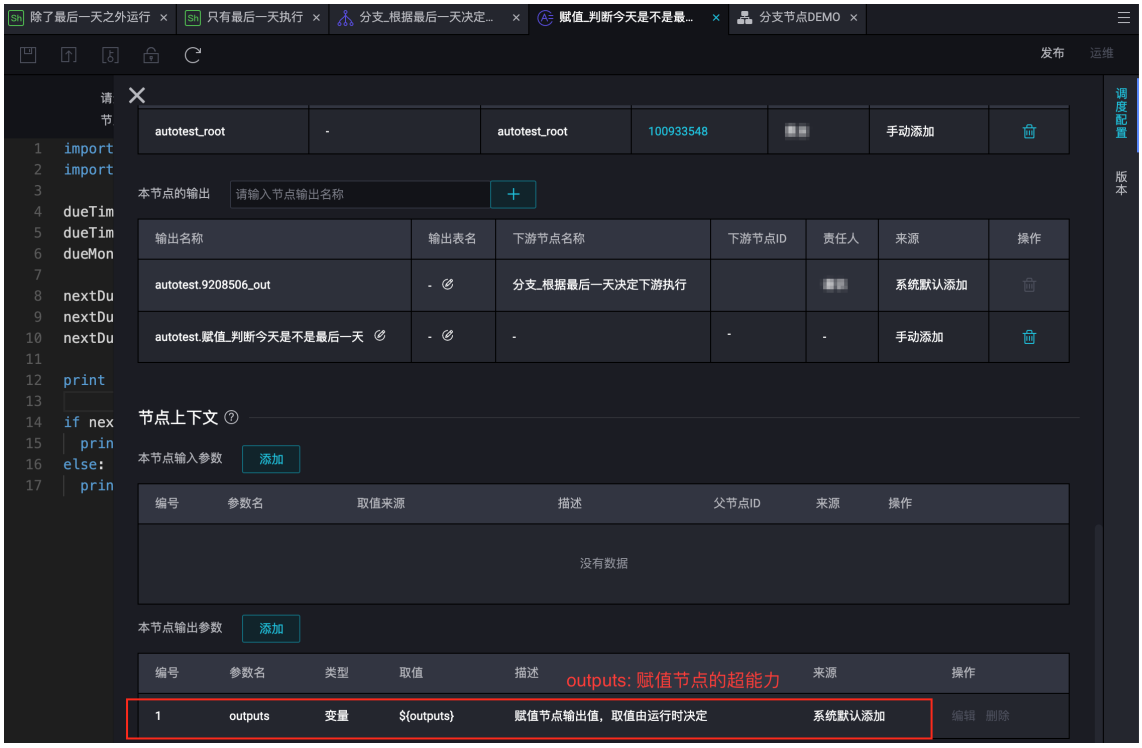
- 对于SQL类型，DataWorks捕获最后一条SELECT语句作为outputs的值。
- 对Shell和Python类型，DataWorks捕获最后一行标准输出作为outputs的值。

本文采用Python类型作为赋值节点的代码，调度属性和代码设置如下。

- 代码设置



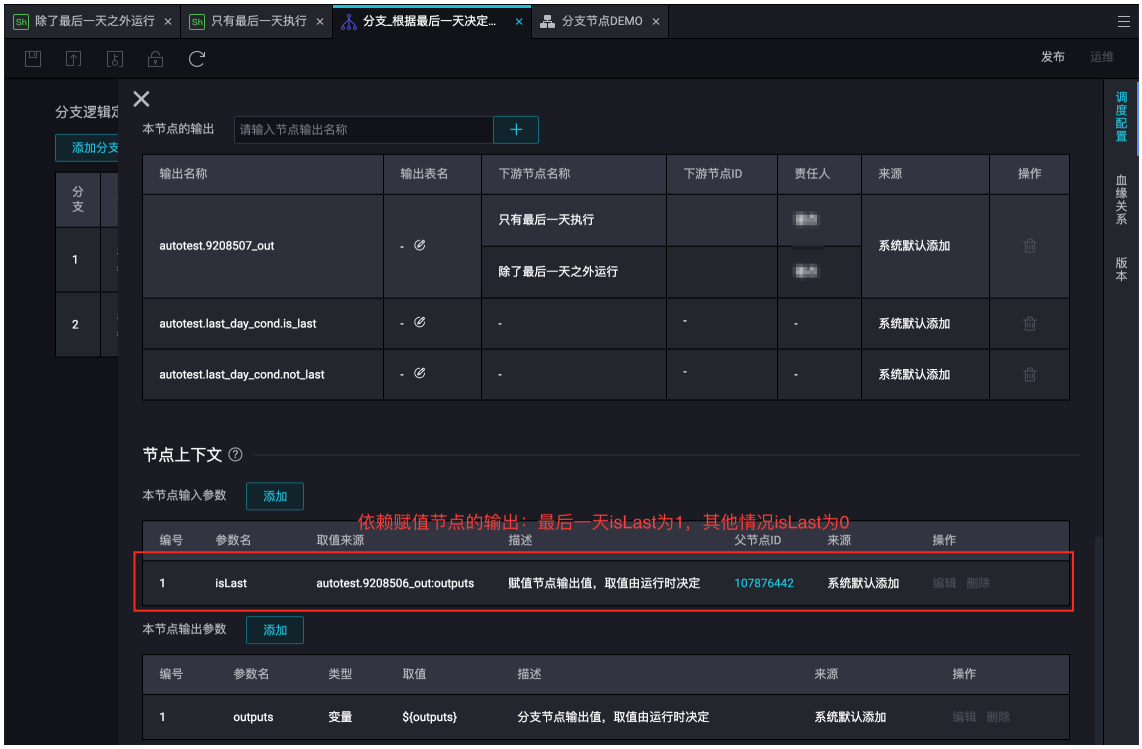
- 调度属性配置



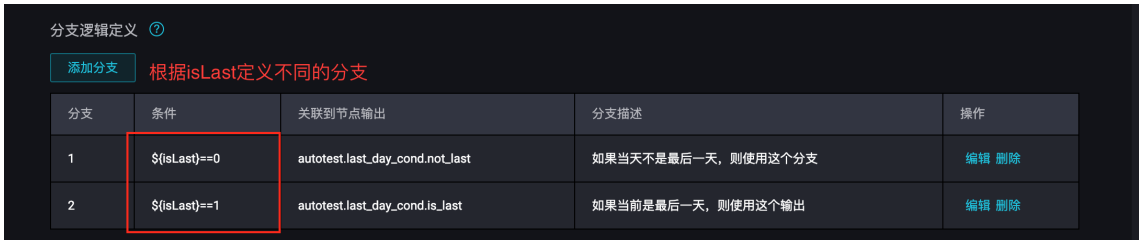
3. 定义分支。

   分支节点可以用Python语法的表达式定义条件，每个条件会绑定一个输出。当满足条件时，该输出的下游节点会被执行，而其它的节点会被置为空跑。

   - 调度配置



- 分支配置



- 调度配置生成条件绑定的输出

4. 将执行任务节点挂在不同分支下。

   分支节点有3个输出，任意选择一个输出当作输入即可。由于现在分支节点的输出关联了条件，所以要慎重选择。

   - 每月最后一天执行的节点依赖

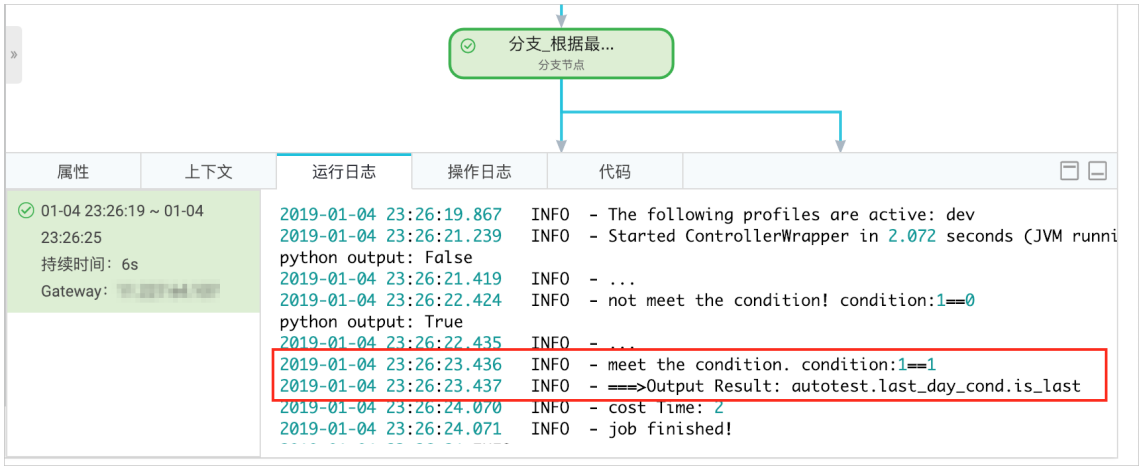
- 每月其它时间执行的节点依赖

5. 验证结果

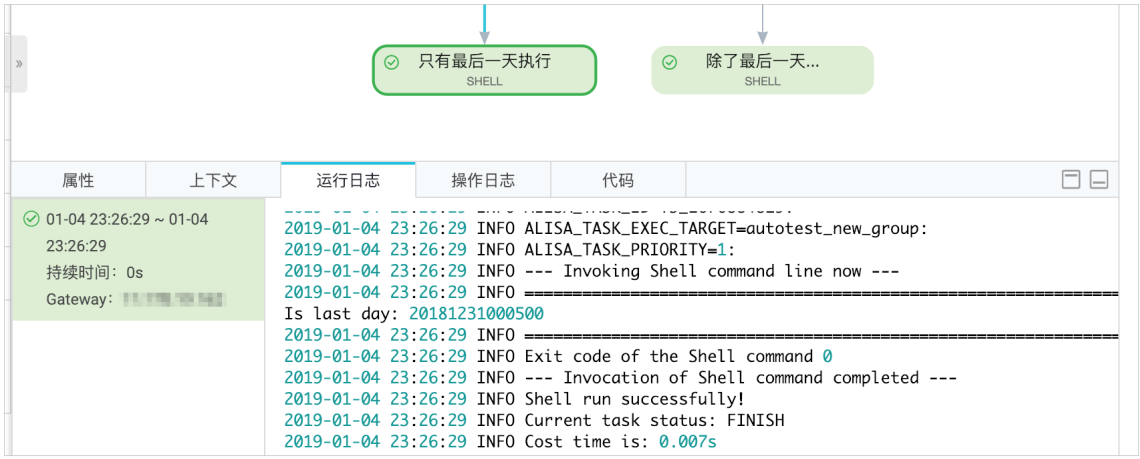
   完成上述所有配置后，您可以提交并发布任务。完成发布后，可以执行补数据来测试效果：业务日期选择2018-12-30和2018-12-31，即定时时间分别为2018-12-31和2019-01-01，则第一批补数据会触发最后一天的逻辑，第二批触发非最后一天的逻辑。两者的区别如下所示。

   业务日期为2018-12-30（即定时时间为2018-12-31）

   - 分支节点分支选择结果



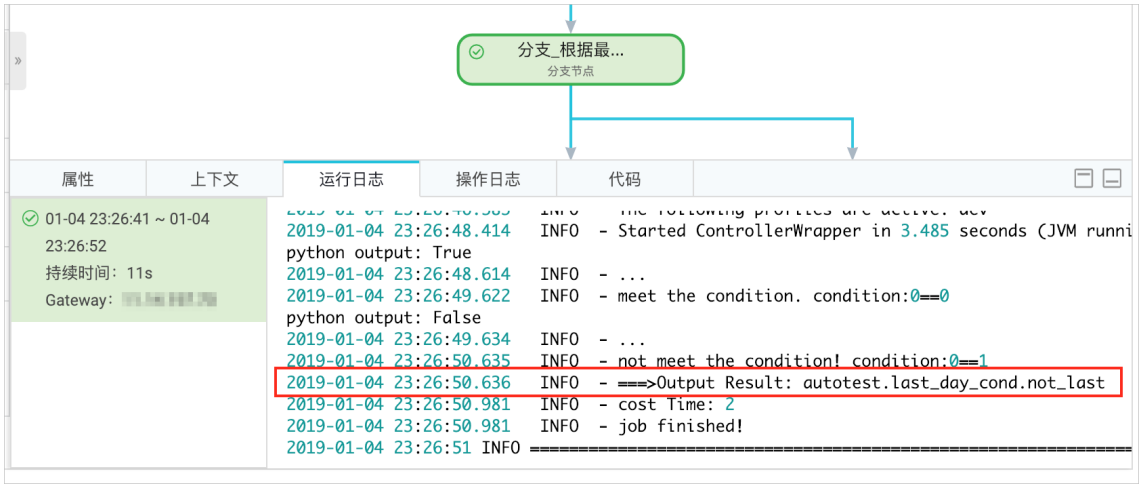
   - 节点（最后一天执行）正常执行



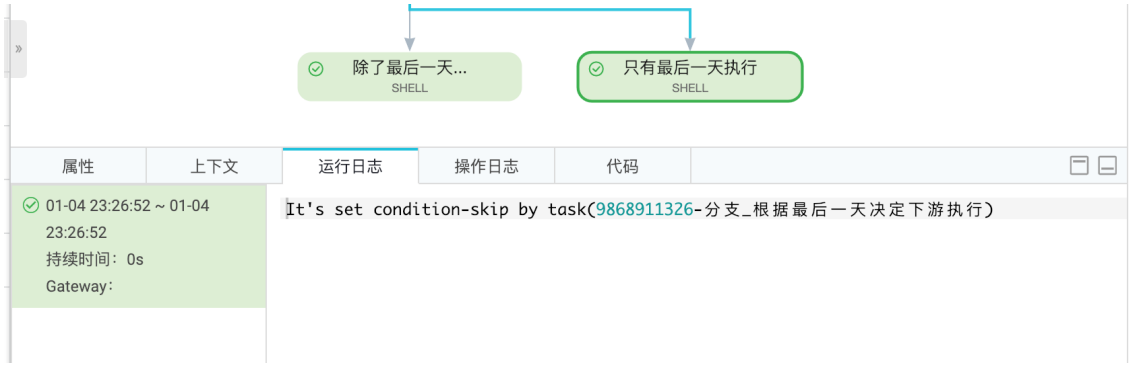
- 节点（除了最后一天之外运行）被置为空跑

业务日期为2018-12-31（即定时时间为2019-01-01）

- 分支节点分支选择结果



- 节点（最后一天执行）被置为空跑



- 节点（除了最后一天之外运行）正常执行

总结

使用分支节点要点如下：

- DataWorks捕获赋值节点的最后一条SELECT语句或者最后一行标准输出流，作为赋值节点的输出，供下游引用。
- 分支节点的每一个输出都被关联了条件，下游挂分支节点作为上游，需要了解每个输出关联的条件的意义再进行选择。
- 未被选中的分支会被置为空跑，并且空跑属性会一直向下传递，直到遇到归并节点。

2.5. DataWorks数据服务对接DataV最佳实践

DataV通过与DataWorks数据服务的对接，通过交互式分析Hologres连接DataWorks数据服务开发并生成API，快速在DataV中调用API并展现MaxCompute的数据分析结果。

数据服务对接DataV产生背景

MaxCompute是阿里巴巴集团自主研究的快速、完全托管的TB、PB和EB级数据仓库解决方案。当今社会数据收集的方式不断丰富，行业数据大量积累，导致数据规模已增长到传统软件行业无法承载的海量级别。MaxCompute服务于批量结构化数据的存储和计算，已经连续多年稳定支撑阿里巴巴全部的离线分析业务。

过去，如果您想要通过DataV展示海量数据的分析结果，需要自建一套离线数据计算自动导入MySQL的任务流程，过程繁琐且成本高。现在通过DataWorks为您提供的**数据集成 > 数据开发 > 数据服务**的全链路数据研发平台，结合MaxCompute即可快速搭建企业数仓。

DataWorks数据服务提供了快速将数据表生成API的功能，通过可视化的向导模式操作，无需代码便可快速生成API，然后通过DataV调用API并在大屏中展示数据分析结果，高效实现数仓的开发和数据的展示。

前提条件

要想实现DataWorks数据服务与DataV的对接，您需要提前准备好数据源，并开通DataV服务。

新建数据源

数据服务支持丰富的数据源类型，如下所示：

- 关系型数据库：RDS、DRDS、MySQL、PostgreSQL、Oracle和SQL Server
- 分析型数据库：AnalyticDB
- NoSQL数据库：TableStore（OTS）和MongoDB

1. 登录DataWorks控制台，单击相应工作空间后的**进入数据服务**。
2. 鼠标悬停至**+新建**，单击**新建数据源**，跳转至**工作空间管理 > 数据源**页面。
3. 单击右上角的**新增数据源**。
4. 在**新增数据源**对话框中，选择数据源类型为**Hologres**。

   本文将以Hologres数据源为例，通过Hologres数据源可以直接实时查询MaxCompute中的数据。
5. 配置**新增Hologres数据源**对话框中的参数。

参数	描述
数据源类型	当前选择的数据源类型为**Hologres**，目前仅支持阿里云实例模式。
数据源名称	数据源名称必须以字母、数字、下划线组合，且不能以数字和下划线开头。
数据源描述	对数据源进行简单描述，不得超过80个字符。
适用环境	可以选择**开发**或**生产**环境。 **说明** 仅标准模式工作空间会显示此配置。
实例ID	需要同步的Hologres实例ID。您可以进入Hologres管理控制台
数据库名数据库名	Hologres的数据库名称。
AccessKey ID	您可以单击AccessKey 管理
AccessKey Secret	您可以单击AccessKey 管理

6. 单击**测试连通性**。
7. 测试连通性通过后，单击**完成**。

新建API

数据源创建完成后，进入**数据服务**页面。本文以向导模式生成API为例，为您介绍如何新建API。

1. 单击左上角的图标，选择**全部产品 > 数据服务**，进入**数据服务**页面。

2. 鼠标悬停至 +新建，选择**生成API > 向导模式**。

3. 配置**生成API**对话框中的参数。

参数	描述
API名称	支持中文、英文、数字、下划线，且只能以英文或中文开头，4~50个字符。
API分组	API分组是指针对某一个功能或场景的API集合，也是API网关对API的最小管理单元。在阿里云API市场中，一个API分组对应于一个API商品。 您可以将鼠标悬停至**新建**图标，单击**新建分组**进行新建。
API Path	API存放的路径，例如*/user*。
协议	目前支持HTTP和HTTPS协议。
请求方式	目前支持GET和POST请求方式。
返回类型	目前仅支持JSON返回类型。
描述	对API进行简要描述。

4. 配置完成后，单击**确认**，即可进入API参数配置页面。

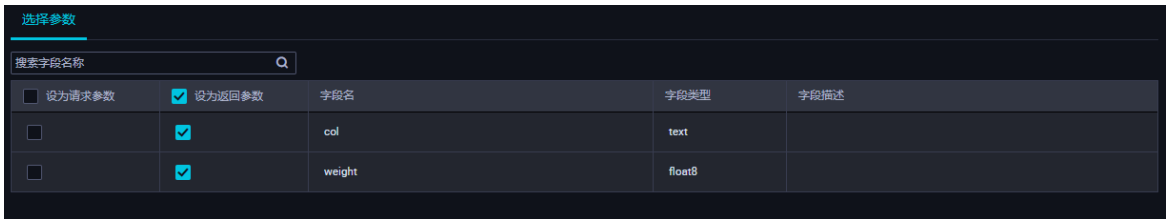
配置API参数

1. 在**选择表**模块， 选择**数据源类型**Hologres、**数据源名称**和**数据表名称**。

> 说明
> - 您需要提前在数据集成中配置好数据源，数据表下拉框支持表名搜索。
> - 创建好API后，会自动跳转至数据表配置页面，您可以直接进行配置。

2. 在**环境配置**模块，设置**内存**和**超时时间**。

3. 选择好数据表后，下方的**选择参数**模块会自动列出该表的所有字段。勾选需要**设为请求参数**和**设为返回参数**的字段，分别添加至请求参数和返回参数列表中。



4. 编辑请求参数信息。

单击页面右侧的**请求参数**，设置**参数名称**、**参数类型**、**操作符**、**是否必填**、**示例值**、**默认值**和**描述**。

5. 编辑返回参数信息。

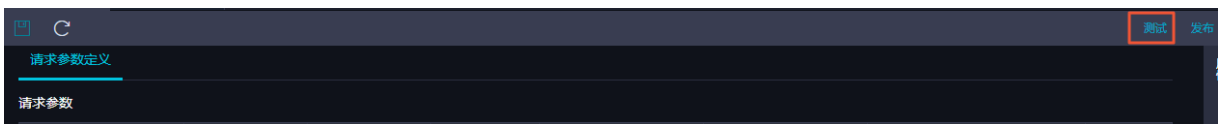
   单击页面右侧的**返回参数**，设置**参数名称**、**参数类型**、**示例值**和**描述**，并可以进行**返回结果分页**和**使用过滤器**等高级配置。

   配置过程中需要注意返回结果分页的设置：

   - 如果不开启**返回结果分页**，则API默认最多返回2,000条记录。
   - 如果返回结果可能超过2,000条，请开启**返回结果分页**功能。

测试API

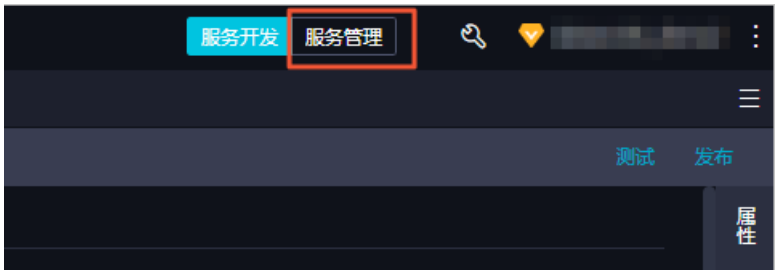
完成API参数的配置并保存后，单击右上角的**测试**，即可进入API测试环节。



填写参数值，单击**开始测试**，即可在线发送API请求，在右侧可以查看API请求详情及返回内容。如果测试失败，请仔细查看错误提示并进行相应的修改重新测试。

发布API

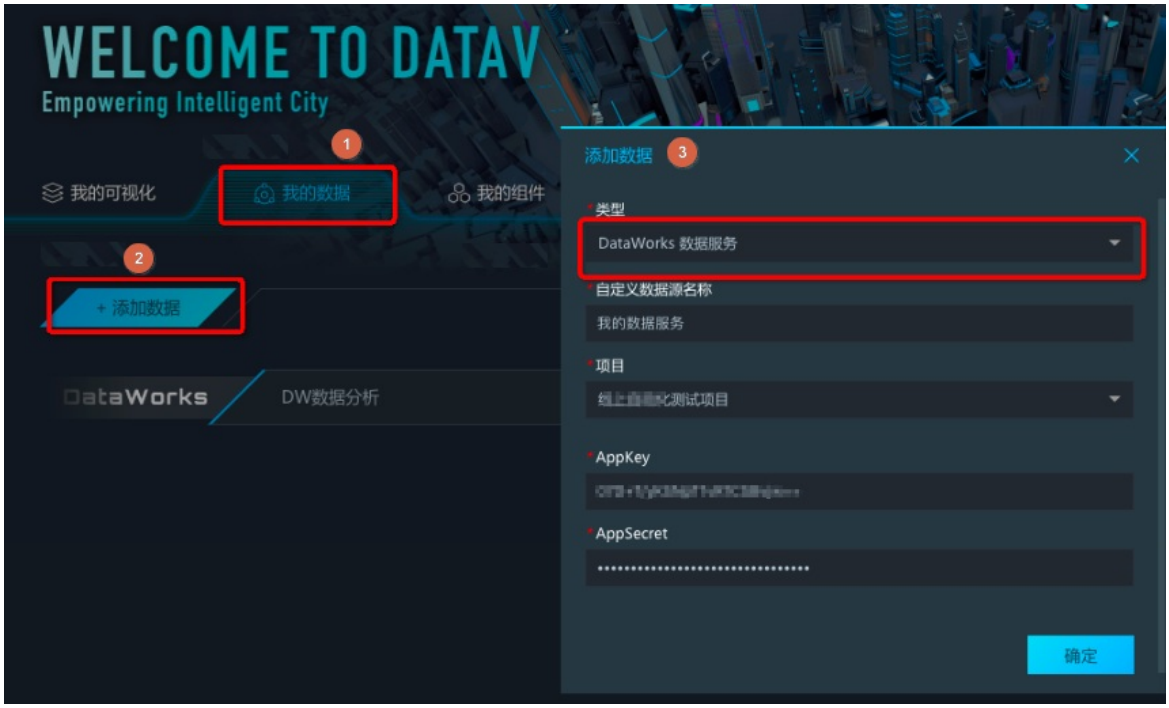
1. 完成API测试后，返回**服务开发**页面。
2. 单击**发布**，即可成功生成一个数据API。
3. 发布完成后，单击右上角**服务管理**查看API详情。



如果您需要调用API，请进入**服务管理API调用**页面，数据服务为您提供简单身份认证（AppCode）和加密签名身份认证（AppKey&AppSecret）两种认证方式，您可以自由选择。下文将为您介绍如何在DataV中进行数据服务API的调用。

添加数据服务为数据源

1. 登录DataV控制台。
2. 进入**我的数据**页面，单击**添加数据**。
3. 填写**添加数据**对话框中的配置。



参数	描述
类型	添加的数据源类型。
自定义数据源名称	数据源的显示名称，可以自由命名。
项目	选择DataWorks工作空间。

参数	描述
AppKey/AppSecret	拥有DataWorks数据服务中某一项目访问权限的账号的AppKeyID和AppSecret。 **说明** 您可以登录DataWorks数据服务控制台，进入**服务管理** > **API调用**页面进行查看。 

在大屏中调用数据服务API

1. 进入DataV控制台中的**我的可视化**页面，单击**新建可视化**。
2. 选择一个模板，单击**创建**，本文以**智能工厂**模板为例。

模板中的组件自带了静态数据，下文将以把模板中间的基本折线图改为调用上文创建好的查询成交金额增长趋势的API为例，为您介绍如何在组件中使用数据服务API。

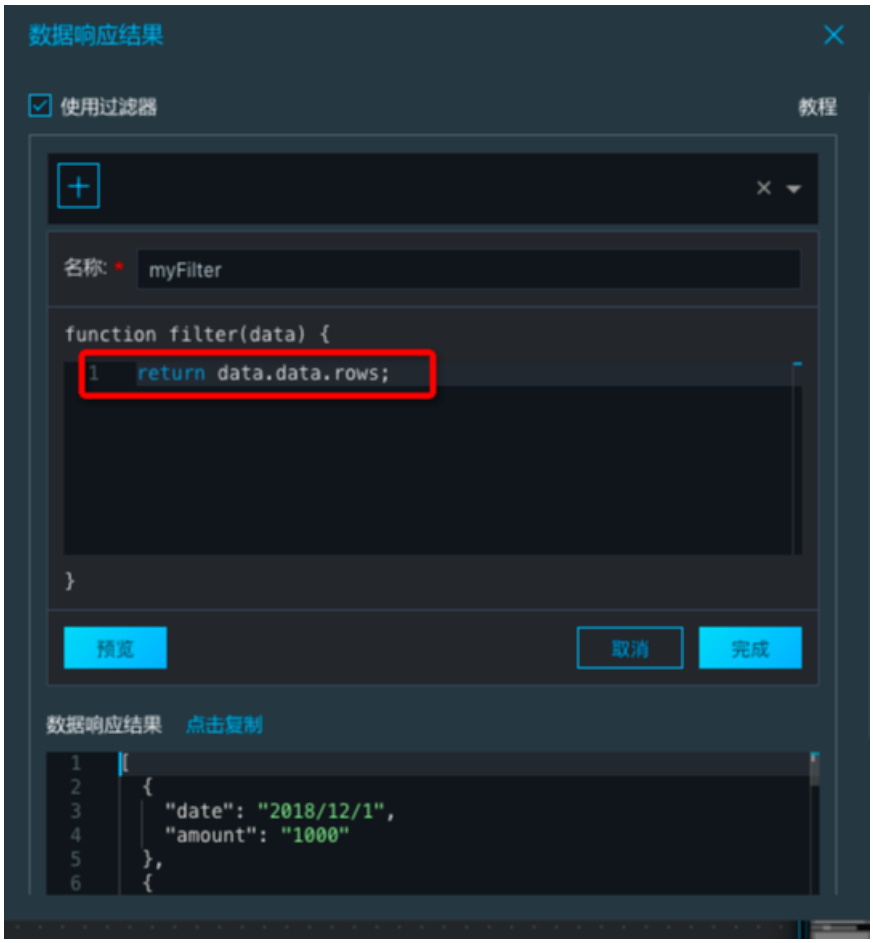
3. 选中基本折线图组件，切换到数据面板，在**数据源类型**中选择**DataWorks数据服务**。
4. 选择刚刚创建的数据源和API，并设置查询参数，本示例将**pageSize**设置为31，以查询一个月的数据。

5. 单击**查看数据响应结果**，即可查看API的查询结果。
6. 填写字段映射关系，在x中填写**date**，将日期作为横轴，在y中填写**amount**，将成交金额作为纵轴。

由上图可见，当前x和y无法匹配到字段。这是因为DataV对数据格式有一定要求，不能识别结构较深的字段，因此需要添加一个数据过滤器，过滤掉不必要的字段，在本例中直接返回rows数组即可。

7. 勾选**使用过滤器**，单击**新建**图标。此处支持编写JS代码对数据结果进行二次过滤和处理，过滤器的data参数为API返回结果JSON对象。

本示例只需返回API结果中的rows数组，因此输入 `return data.data.rows;` 在下方预览过滤后的结果，并单击**完成**。



添加过滤器后，字段便会匹配成功。

由于API返回的日期格式与组件默认的格式不一样，此时的折线图并没有正确展示，您还需要设置折线横轴的日期格式。

8. 切换至配置面板，在**x轴 > 轴标签**中选择数据种类为**时间型**，数据格式选择本API所返回的格式2016/01/01，即可看见折线图的正常展示。

至此，便完成了通过数据服务将MaxCompute表生成API，然后在DataV数据大屏中进行展示的所有操作，效果如下图所示。

注意事项

DataWorks数据服务与DataV进行无缝对接后，无需使用DataV中的API数据源去填写一个URL调用API，直接新建一个DataWorks数据服务作为数据源，便可直接选用数据服务中的API。无需每个API都设置AppKey和AppSecret认证信息，且支持通过表单填写API参数，操作快捷方便并安全可靠。

通过数据服务，您可以将MaxCompute中加工好的数据结果，直接在DataV中进行呈现，实现数据开发-数据服务-数据分析展现的全链路开发。

在开发过程中，请注意以下事项：

- DataWorks数据服务向导模式生成API仅支持单表简单条件查询，脚本模式支持您编写查询SQL语句，支持多表关联查询、函数以及复杂条件。您可以根据自己的需求灵活选择。
- 如果您要求毫秒级API查询，建议使用关系型数据库、NoSQL数据库或AnalyticDB作为数据源。
- DataV组件要求的数据格式是个数组，数据服务生成的API返回结果是带有错误码的完整JSON，因此要使用过滤器对API结果进行处理。您可以选择在DataV中添加过滤器，也可以选择直接在数据服务配置API时添加过滤器。

  通常对于未分页查询的API，直接返回data数组即可，对于分页查询的API直接返回data.rows数组。
- 如果您要在DataV的折线图或柱状图中添加多个系列，通常DataV要求每个系列的数据是一个对象，并通过字段来区分系列，此时需要注意使用过滤器进行格式转换。

2.6. 天任务依赖分钟任务最佳实践

每天00:00执行的SQL任务为天任务，依赖于每5分钟抽取一次数据的分钟任务。天任务会对当天同步任务抽取的所有数据进行计算。

前提条件

开始本实验前，您需要首先准备好以下内容：

- 请确保已拥有阿里云账号并进行实名认证，详情请参见开通DataWorks。
- 开通MaxCompute并创建工作空间，详情请参见创建工作空间。
- 通过RDS创建MySQL实例，获取RDS实例ID，并在RDS控制台添加白名单。详情请参见创建RDS MySQL实例和添加白名单。

> **说明**
> - 如果是通过自定义资源组调度RDS的数据同步任务，必须把自定义资源组的机器IP也加入RDS的白名单中。
> - 本文以MySQL数据源为例，您可以根据自身需求新增不同类型的数据源。

背景信息

在DataWorks调度系统中，下游对上游的依赖需要遵循的原则为：下游任务生成的实例会找到当天离自己最近结束的一个上游实例作为上游依赖，如果上游依赖实例运行成功，才会触发本节点实例运行。如果上游节点每天生成多个实例，则下游无法识别是哪一个实例离它最近结束，导致必须等上游当天生成的所有实例运行完成后才会运行。

因此，上游节点必须配置自依赖，SQL任务在00:00的实例才会准确依赖00:00生成的同步任务实例结束后再运行。

本实验的实现思路如下：

1. 创建一个同步节点作为上游的分钟任务，一个SQL节点作为下游的天任务。
2. 设置同步节点的调度时间为每5分钟调度一次（开始时间00:00，结束时间23:59，时间间隔5分钟）。
3. 配置同步节点**依赖上一周期 > 本节点**，以形成自依赖。
4. 设置SQL任务每天00:00调度一次。

> **说明** 由于天任务依赖分钟任务，如果分钟级任务失败，会影响天任务的执行。

操作步骤

1. 新增数据源。
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**工作空间列表**。
   iii. 选择工作空间所在地域后，单击相应工作空间后的**进入数据集成**。
   iv. 在左侧导航栏，单击**数据源**，进入**数据源管理**页面。
   v. 单击右上角的**新增数据源**，添加MySQL数据源，详情请参见配置MySQL数据源。
2. 新建业务流程。
   i. 单击当前页面左上角的☰图标，选中**全部产品 > DataStudio（数据开发）**。
   ii. 鼠标悬停至+新建图标，单击**业务流程**。
   iii. 在**新建业务流程**对话框中，输入**业务名称**和**描述**。

   > 注意 业务名称必须是大小写字母、中文、数字、下划线（_）以及小数点（.），且不能超过128个字符。

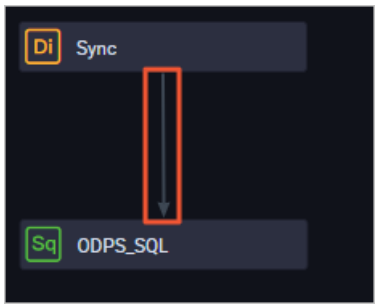
   iv. 单击**新建**。
   v. 进入业务流程开发面板，鼠标单击**离线同步**并拖拽至右侧的编辑页面。

   vi. 在**新建节点**对话框，输入**节点名称**，单击**提交**。

   以同样的方式新建一个ODPS SQL节点。

   > 注意 节点名称必须是大小写字母、中文、数字、下划线（_）以及小数点（.），且不能超过128个字符。

vii. 通过拖拽连线，设置离线同步节点为ODPS SQL节点的上游。



- 数据同步节点用来同步每5分钟调度一次的MySQL数据至MaxCompute。
- ODPS SQL节点用来汇总MaxCompute接收的数据。

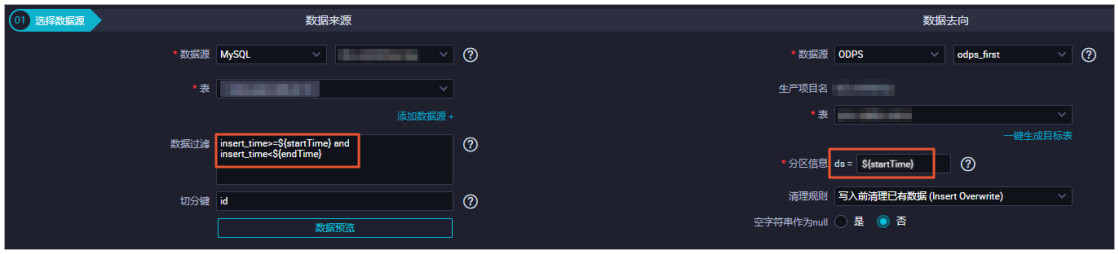
viii. 单击工具栏中的图标。

3. 配置作为分钟任务的离线同步节点。

i. 双击离线同步节点，进入该节点的编辑页面。

ii. 选择**数据来源**和**数据去向**。

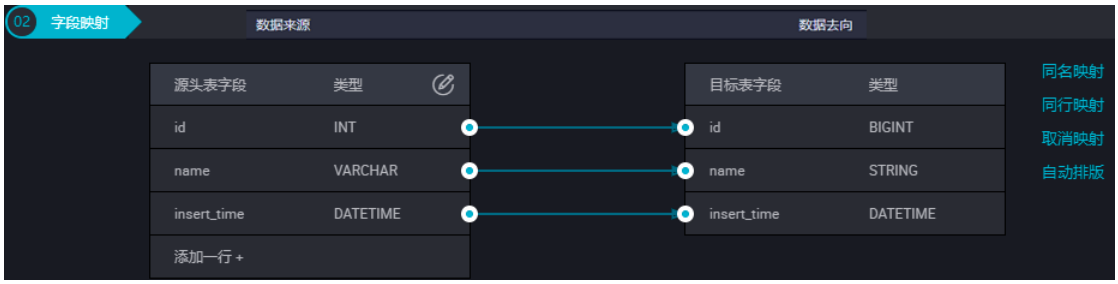
本示例为同步MySQL的数据至MaxCompute。根据过滤条件过滤每5分钟更新的数据，目标端的分区根据定时时间的前5分钟创建，以保证所有数据都写入同一天的分区中。



输入**数据过滤**为 `insert_time>=${startTime} and insert_time<${endTime}` 。

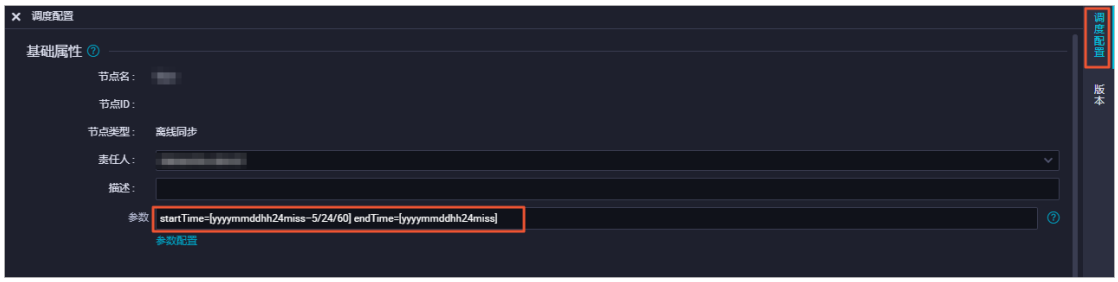
iii. 配置字段映射。

数据来源和**数据去向**均创建了id、name、insert_time三列字段。insert_time为时间列，数据可以根据时间来过滤。



iv. 单击右侧的**调度配置**。

在**基础属性**页签，为过滤条件中的参数赋值：`startTime=[yyyymmddhh24miss-5/24/60] endTime=[yyyymmddhh24miss]`，每个参数间用空格分隔。



在**时间属性**页签，设置**调度周期**为**分钟**，**开始时间**为**00:00**，每间隔5分钟调度一次。为保证一天的实例都运行完，您需要设置自依赖。

- 上述配置可以保证您一天产生的实例能够依次运行完成，并保存至MaxCompute表同一天的分区中。
- 如果您的分钟任务中有一个调度任务出错，则设置自依赖后面的实例都不会运行，需要您手动进行处理。
- 为避免出现上述问题，您可以设置数据过滤为`insert_time<${endTime}`，每次都进行全量同步，只要有成功的便可同步endTime数据。您无需设置自依赖，但会增加数据库的负担。

4. 配置作为天任务的ODPS SQL节点。

i. 双击ODPS SQL节点，进入该节点的编辑页面。

ii. 过滤workshop_odps_mi一天分区中的数据并插入至workshop_odps_dd表中。

```
insert overwrite table workshop_odps_dd partition (ds=${yestoday})
select id, name,insert_time from workshop_odps_mi where ${startTime}<=ds and ds<${e
ndTime};
```

本文以ds=20190320为例，在节点编辑页面，输入如下语句。

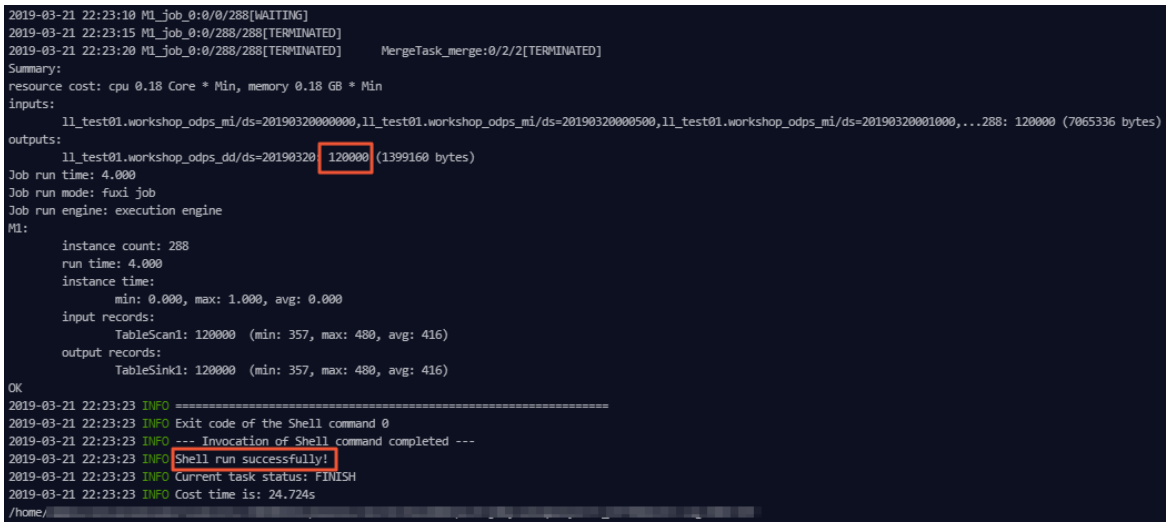
```
insert overwrite table workshop_odps_dd partition (ds=20190320)
select id, name,insert_time from workshop_odps_mi where 20190320000000<=ds and ds<2
0190321000000;
```

因为天运行定时时间在分钟任务运行结束后，所以插入workshop_odps_dd分区（ds）的时间需要和分钟任务时间在同一天，您将ds时间减1即可。

iii. 单击右侧的**调度配置**，参数赋值为 `startTime=$[yyyymmddhh24miss-1] endTime=$[yyyymmddhh24miss] yestoday=$[yyyymmdd-1]` 。

iv. 单击工具栏中的图标。

5. 单击业务流程工具栏中的图标，查看运行结果。



2.7. 通过DataWorks实现邮件外发最佳实践

本文为您介绍如何通过PyODPS节点结合独享资源组的方式，实现邮件外发的需求。

背景信息

DataWorks中的PyODPS节点和Python脚本有所区别，PyODPS节点主要用于和MaxCompute交互进行数据分析处理。DataWorks暂不支持自定义发送邮件功能，您可以通过PyODPS节点结合独享资源组的方式，实现从MaxCompute读取数据进行邮件外发的场景需求。

> **说明** TCP 25端口是默认的邮箱服务端口。出于安全考虑，云服务器ECS的25端口默认受限，独享资源组无法支持该端口，建议您使用465端口发送邮件。

通过PyODPS节点结合独享资源组的方式实现邮件外发时，独享资源组用户无法登录到对应的机器，会导致无法安装更多Python第三方模块，实现更多的功能。

操作步骤

1. 新增独享资源组。
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**资源组列表**。
   iii. 在**独享资源组**页签下，单击**新增独享资源组**。
   iv. 在**新增独享资源组**对话框中，配置各项参数，详情请参见新增和使用独享调度资源组。

   > **说明** 独享资源组和DataWorks工作空间的地域请保持一致。

   v. 单击**创建**。
2. 分配独享资源组至相应的工作空间。
   i. 单击相应资源组后的**修改归属工作空间**。
   ii. 在**修改归属**对话框中，选中要分配的工作空间。
   iii. 单击**确定**。
3. 进入**数据开发**页面。
   i. 在DataWorks控制台的左侧导航栏，单击**工作空间列表**。
   ii. 在页面左上角切换工作空间所在的地域。
   iii. 单击相应工作空间后的**进入数据开发**。
4. 新建PyODPS 2节点。
   i. 鼠标悬停至 +新建 图标，单击**MaxCompute > PyODPS 2**。

      您也可以打开相应的业务流程，右键单击**MaxCompute**，选择**新建 > PyODPS 2**。

ii. 在**新建节点**对话框中，输入**节点名称**，并选择**目标文件夹**。

> **说明** 节点名称必须是大小写字母、中文、数字、下划线（_）和小数点（.），且不能超过128个字符。

iii. 单击**提交**。

iv. 在PyODPS 2节点的编辑页面，输入如下SMTP发送代码。

```
import smtplib
from email.mime.text import MIMEText
from odps import ODPS
mail_host = '<yourHost>' //邮箱服务地址
mail_username = '<yourUserName>' //登录用户名
mail_password = '<yourPassWord>'  //登录用户密码
mail_sender = '<senderAddress>' //发件人邮箱地址
mail_receivers = ['<receiverAddress>']  //收件人邮箱地址
mail_content=""        //邮件内容
o=ODPS('access_key','access_secretkey','default_project_name',endpoint='maxcompute_
service_endpoint')
with o.execute_sql('query_sql').open_reader() as reader:
          for record in reader:
                  mail_content+=str(record['column_name'])+' '+record['column_name
']+'\n'
message = MIMEText(mail_content,'plain','utf-8')
message['Subject'] = 'mail test'
message['From'] = mail_sender
message['To'] = mail_receivers[0]
try:
          smtpObj = smtplib.SMTP_SSL(mail_host+':465')
          smtpObj.login(mail_username,mail_password)
          smtpObj.sendmail(
              mail_sender,mail_receivers,message.as_string())
          smtpObj.quit()
          print('mail send success')
except smtplib.SMTPException as e:
          print('mail send error',e)
```

或者您可以使用以下发送邮件代码的示例：

```
import smtplib
from email.mime.text import MIMEText
from odps import ODPS
mail_host = 'username@example.com'  //邮箱服务地址
mail_username = 'xxxx' //登录用户名
mail_password = 'xxx'  //登录用户密码
mail_sender = 'xxx' //发件人邮箱地址
mail_receivers = ['xxx'] //收件人邮箱地址
mail_content=""        //邮件内容
o=ODPS('access_key','access_secretkey','default_project_name',endpoint='maxcompute_
service_endpoint')
with o.execute_sql('query_sql').open_reader() as reader:
    for record in reader:
                mail_content+=str(record['column_name'])+' '+record['column_name']+
'\n'
message = MIMEText(mail_content,'plain','utf-8')
message['Subject'] = 'mail test'
message['From'] = mail_sender
message['To'] = mail_receivers[0]
try:
           smtpObj = smtplib.SMTP()
           smtpObj.connect(mail_host,587)
           smtpObj.ehlo()
           smtpObj.starttls()
           smtpObj.login(mail_username,mail_password)
           smtpObj.sendmail(
               mail_sender,mail_receivers,message.as_string())
           smtpObj.quit()
           print('mail send success')
except smtplib.SMTPException as e:
          print('mail send error',e)
```

> **说明** 使用PyODPS 2节点外发邮件时，PyODPS会将读取的数据存放至一个临时文件中，再通过邮件的形式外发。外发邮件时没有数据条数的限制。

v. 单击工具栏中的图标。

5. 提交节点。

> **注意** 提交节点前，您需要单击右侧的**调度配置**，设置**重跑属性**和**依赖的上游节点**。

i. 单击工具栏中的图标。

ii. 在**提交新版本**对话框中，输入**备注**。

iii. 单击**确认**。

如果您使用的是标准模式的工作空间，提交节点后，请单击右上角的**发布**。详情请参见发布任务。

6. 修改运行节点的资源组。

i. 在PyODPS 2节点的编辑页面，单击右上方的**运维**。

ii. 在左侧导航栏，单击**周期任务运维 > 周期任务**。

iii. 单击页面中间的箭头，展开任务列表。

iv. 单击相应节点后的**更多 > 修改资源组**

v. 在**修改资源组**对话框中，**选择资源组**。

vi. 单击**确定**。

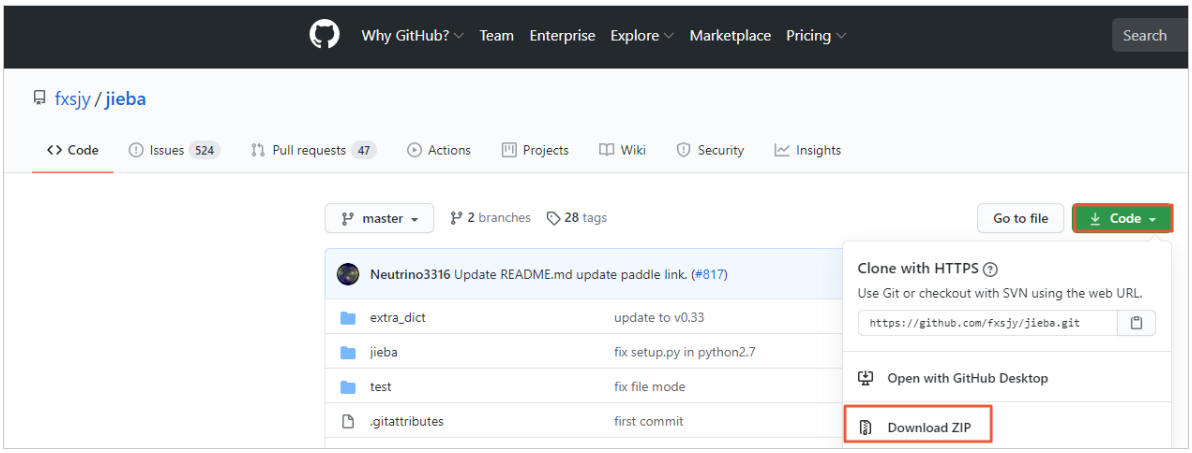
7. 测试节点，详情请参见查看并管理周期任务。

2.8. PyODPS节点实现结巴中文分词

本文为您介绍如何使用DataWorks的PyODPS类型节点，借助开源结巴中文分词包实现对中文字段的分词并写入新的表，以及如何通过闭包函数使用自定义词典进行分词。

前提条件

- 请首先确保您已经完成DataWorks工作空间的创建，本示例使用绑定多个MaxCompute计算引擎的**简单模式**工作空间，详情请参见创建工作空间。
- 请在GitHub下载开源结巴分词中文包。



背景信息

PyODPS集成了MaxCompute的Python SDK。您可以在DataWorks的PyODPS节点上直接编辑Python代码，并使用MaxCompute的Python SDK。关于PyODPS节点的详情请参见创建PyODPS 2节点。

> **注意** 本文的操作仅作为代码示例，不建议用于实际的生产环境。

操作步骤

1. 创建业务流程。

   i. 登录DataWorks控制台。

   ii. 在左侧导航栏，单击**工作空间列表**。

   iii. 选择工作空间所在地域后，单击相应工作空间后的**进入数据开发**。

   iv. 鼠标悬停至+新建图标，单击**业务流程**。

v. 在**新建业务流程**对话框中，输入**业务名称**和**描述**。

> 注意 业务名称必须是大小写字母、中文、数字、下划线（_）以及小数点（.），且不能超过128个字符。

vi. 单击**新建**。

2. 上传jieba-master.zip包。

i. 展开新建的业务流程下的**MaxCompute**，右键单击**资源**，选择**新建 > Archive**。

ii. 在**新建资源**对话框中，配置各项参数。

参数	描述
资源名称	资源的名称，无需和上传的文件名保持一致，但需要符合以下规范： ■ 资源名称仅包含中文、字母、数字、英文句号（.）、下划线（_）和短划线（-）。 ■ 资源类型为Archive时，资源名称和文件名的后缀必须一致，且后缀名包含.zip、.tgz、.tar.gz或.tar。
目标文件夹	默认当前所在文件夹的路径，您可以进行修改。
资源类型	此处选择**Archive**类型。 说明 如果该资源包已经在MaxCompute（ODPS）客户端上传过，请取消勾选**上传为ODPS资源**，否则上传会报错。
MaxCompute引擎实例	从下拉列表中选择该资源所在的计算引擎。 如果您所在的工作空间仅绑定一个实例，则不会显示该参数。
上传文件	单击**点击上传**，在本地选择已下载的文件jieba-master.zip后，单击**打开**。

iii. 在**新建资源**对话框中，单击**确定**。

iv. 单击工具栏中的图标，在**提交新版本**对话框中，输入**变更描述**。

v. 单击**确认**。

3. 创建测试数据表。

i. 展开新建的业务流程下的**MaxCompute**，右键单击**表**，选择**新建表**。

ii. 在**新建表**对话框中，输入**表名**（示例为jieba_test），并选择**MaxCompute引擎实例**。

iii. 单击**提交**。

iv. 单击**DDL模式**，输入以下建表DDL语句。

本教程准备了两列测试数据，您在后续开发过程中可以选择一列进行分词。

```
CREATE TABLE jieba_test (
    `chinese` string,
    `content` string
);
```

v. 单击**生成表结构**。

vi. 在**确认操作**对话框中，单击**确认**。

vii. 在**基本属性**区域，输入表的**中文名**。

viii. 单击**提交到生产环境**。

ix. 在**提交生产确认**对话框中，选中**我已悉知风险，确认提交**，单击**确认**。

4. 以同样的方式创建存放测试结果的数据表。

本例仅对测试数据的chinese列进行分词处理，因此结果表仅有一列。DDL语句如下所示。

```
CREATE TABLE jieba_result (
    `chinese` string
) ;
```

5. 下载分词测试数据。

6. 上传测试数据：

i. 在**数据开发**页面，单击图标。

ii. 在**数据导入向导**对话框中，输入需要导入数据的测试表jieba_test并选中，单击**下一步**。

iii. 单击**浏览**，上传您下载至本地的*jieba_test.csv*文件，单击**下一步**。

iv. 选中**按名称匹配**，单击**导入数据**。

7. 创建PyODPS 2节点。

i. 展开业务流程中的**MaxCompute**，右键单击**数据开发**，选择**新建 > PyODPS 2**。

ii. 在**新建节点**对话框中，输入**节点名称**（示例为word_split），并选择**目标文件夹**。

> **说明** 节点名称必须是大小写字母、中文、数字、下划线（_）和小数点（.），且不能超过128个字符。

iii. 单击**提交**。

iv. 在节点的编辑页面，选中**MaxCompute引擎实例**，并输入下述PyODPS代码。

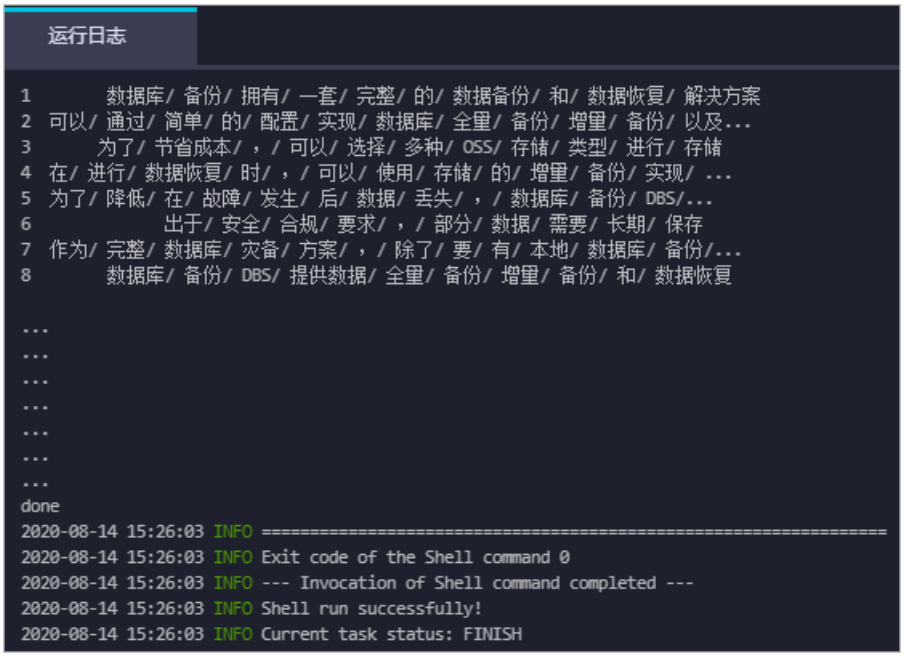
```
def test(input_var):
    import jieba
    import sys
    reload(sys)
    sys.setdefaultencoding('utf-8')
    result=jieba.cut(input_var, cut_all=False)
    return "/ ".join(result)
hints = {
    'odps.isolation.session.enable': True
}
libraries =['jieba-master.zip']  #引用您的jieba-master.zip压缩包。
iris = o.get_table('jieba_test').to_df()  #引用您的jieba_test表中的数据。
example = iris.chinese.map(test).execute(hints=hints, libraries=libraries)
print(example)  #查看分词结果，分词结构为MAP类型数据。
abci=list(example)   #将分词结果转为list类型的数据。
i = 0
for i in range(i,len(abci)):
    pq=str(abci[i])
    o.write_table('jieba_result',[pq])  #通过循环，逐条写入数据至结果表jieba_result中。
    i+=1
else:
    print("done")
```

v. 单击工具栏中的图标，保存输入的代码。

vi. 单击工具栏中的图标，在**参数**对话框中，从**调度资源组**下拉列表选择需要使用的资源组。

vii. 单击**确定**，测试PyODPS 2节点。

viii. 在页面下方的**运行日志**区域，查看结巴分词程序的运行结果。



8. 创建和运行ODPS SQL节点。

i. 展开业务流程中的**MaxCompute**，右键单击**数据开发**，选择**新建 > ODPS SQL**。

ii. 在**新建节点**对话框中，输入**节点名称**，并选择**目标文件夹**。

> **说明** 节点名称必须是大小写字母、中文、数字、下划线（_）和小数点（.），且不能超过128个字符。

iii. 单击**提交**。

iv. 在节点的编辑页面，输入SQL语句 `select * from jieba_result;` 。

v. 单击工具栏中的图标，保存输入的查询语句。

vi. 单击工具栏中的图标，在**参数**对话框中，从**调度资源组**下拉列表选择需要使用的资源组。

vii. 单击**确定**，运行查询语句。

viii. 在**成本估计**对话框中，确认**预估费用**后，单击**运行**。

ix. 在页面下方的**运行日志**区域，查看运行结果。

9. 如果开源结巴分词的词库无法满足您的需求，需要使用自定义的词典。

PyODPS自定义函数可以读取上传至MaxCompute的资源（表资源或文件资源）。此时，自定义函数需要写为闭包函数或Callable类。如果您需要引用复杂的自定义函数，则可以使用DataWorks的注册MaxCompute函数功能，详情请参见注册MaxCompute函数。

本文以使用闭包函数的方式，引用上传至MaxCompute的资源文件（即自定义词典）key_words.txt：

i. 展开业务流程中的**MaxCompute**，右键单击**资源**，选择**新建 > File**。

ii. 在**新建资源**对话框中，配置各项参数。

参数	描述
资源名称	资源的名称，仅支持中文、字母、数字、英文句号（.）、下划线（_）和短划线（-）。
目标文件夹	默认当前所在文件夹的路径，您可以进行修改。
资源类型	此处选择File类型。 本文中的资源文件，在新建时请务必选中**上传为ODPS资源**。
MaxCompute引擎实例	从下拉列表中选择该资源所在的引擎实例。

iii. 单击**确定**。

iv. 在资源的编辑页面，选择**MaxCompute引擎实例**，并输入自定义词典的内容。

词典格式：

- 一个词占一行。
- 每一行包括词、词频（可省略）和词性（可省略），使用空格分隔，且不可以颠倒顺序。

如果您从本地上传词典文件至DataWorks，则文件必须为UTF-8编码格式。

v. 单击工具栏中的图标提交资源。

vi. 创建一个PyODPS 2节点，并输入如下代码。

```
def test(resources):
    import jieba
    import sys
    reload(sys)
    sys.setdefaultencoding('utf-8')
    fileobj = resources[0]
    def h(input_var):#在嵌套函数h()中，执行词典加载和分词。
        import jieba
        jieba.load_userdict(fileobj)
        result=jieba.cut(input_var, cut_all=False)
        return "/ ".join(result)
    return h
hints = {
    'odps.isolation.session.enable': True
}
libraries =['jieba-master.zip']  #引用您的jieba-master.zip压缩包。
iris = o.get_table('jieba_test').to_df()  #引用您的jieba_test表中的数据。
file_object = o.get_resource('key_words.txt') #get_resource()引用odps资源。
example = iris.chinese.map(test, resources=[file_object]).execute(hints=hints, libr
aries=libraries) #map调用函数，并传递resources参数。
print(example)  #查看分词结果，分词结构为MAP类型数据。
abci=list(example)   #将分词结果转为list类型数据。
for i in range(i,len(abci)):
    pq=str(abci[i])
    o.write_table('jieba_result',[pq])  #通过循环，逐条写入数据至结果表jieba_result中。
    i+=1
else:
    print("done")
```

vii. 运行代码，对比引用自定义词典前后的结果。

引用自定义词典前结果如下。

运行日志

```
                                                chinese
0   数据库/ 备份/ 是/ 为/ 数据库/ 提供/ 连续/ 数据保护/ 低成本/ 的/ 备份/ 服务
1        数据库/ 备份/ 拥有/ 一套/ 完整/ 的/ 数据备份/ 和/ 数据恢复/ 解决方案
2  可以/ 通过/ 简单/ 的/ 配置/ 实现/ 数据库/ 全量/ 备份/ 增量/ 备份/ 以及...
3      为了/ 节省成本/ ，/ 可以/ 选择/ 多种/ OSS/ 存储/ 类型/ 进行/ 存储
4  在/ 进行/ 数据恢复/ 时/ ，/ 可以/ 使用/ 存储/ 的/ 增量/ 备份/ 实现/ ...
5  为了/ 降低/ 在/ 故障/ 发生/ 后/ 数据/ 丢失/ ，/ 数据库/ 备份/ DBS/...
6              出于/ 安全/ 合规/ 要求/ ，/ 部分/ 数据/ 需要/ 长期/ 保存
7  作为/ 完整/ 数据库/ 灾备/ 方案/ ，/ 除了/ 要/ 有/ 本地/ 数据库/ 备份/...
8        数据库/ 备份/ DBS/ 提供数据/ 全量/ 备份/ 增量/ 备份/ 和/ 数据恢复
```

引用自定义词典后结果如下。

```
1        数据库/ 备份/ 拥有/ 一套/ 完整/ 的/ 数据备份/ 和/ 数据恢复/ 解决方案
2  可以/ 通过/ 简单/ 的/ 配置/ 实现/ 数据库/ 全量/ 备份/ 增量/ 备份/ 以及...
3      为了/ 节省成本/ ，/ 可以/ 选择/ 多种/ OSS存储/ 类型/ 进行/ 存储
4  在/ 进行/ 数据恢复/ 时/ ，/ 可以/ 使用/ 存储/ 的/ 增量/ 备份/ 实现/ ...
5  为了/ 降低/ 在/ 故障/ 发生/ 后/ 数据/ 丢失/ ，/ 数据库/ 备份/ DBS/...
6              出于/ 安全/ 合规/ 要求/ ，/ 部分/ 数据/ 需要/ 长期/ 保存
7  作为/ 完整/ 数据库/ 灾备/ 方案/ ，/ 除了/ 要有/ 本地/ 数据库/ 备份/ 外...
8        数据库/ 备份/ DBS/ 提供数据/ 全量/ 备份/ 增量/ 备份/ 和/ 数据恢复
```

2.9. 基于AnalyticDB构建企业数仓

本文将为您介绍如何基于AnalyticDB构建企业数仓，并进行运维和元数据管理等操作。

开始本文的操作前，请首先创建工作空间，详情请参见创建工作空间。

配置AnalyticDB for MySQL 2.0数据源

1. 登录DataWorks控制台，单击相应工作空间后的**进入数据集成**。
2. 单击左侧导航栏中的**数据源**，即可跳转至**工作空间管理 > 数据源管理**页面。
3. 在**数据源管理**页面，单击右上角的**新增数据源**。
4. 在**新增数据源**对话框中，选择数据源类型为**AnalyticDB for MySQL 3.0**。
5. 配置AnalyticDB for MySQL 2.0数据源的参数，详情请参见配置AnalyticDB for MySQL 2.0数据源。

> 说明
>
> - AnalyticDB for MySQL 节点只支持使用独享调度资源组，其他资源组无法访问专有网络环境下的AnalyticDB for MySQL 实例，会出现链接超时的情况。独享调度资源组使用详情请参考文档新增和使用独享调度资源组。
> - 如果使用的是AnalyticDB for MySQL 2.0版本，通过用户AK信息进行身份验证。
> - 如果使用的是AnalyticDB for MySQL 3.0版本，通过数据库的用户名和密码进行身份验证（开通3.0版本数据库后，首先在控制台创建用户和密码）。

6. 单击**测试连通性**。
7. 测试连通性通过后，单击**完成**。

设置AnalyticDB for MySQL 3.0白名单

由于AnalyticDB for MySQL 3.0版本基于用户名密码访问，因此需要设置客户端白名单，才允许连接数据库。

1. 获取独享调度资源组白名单

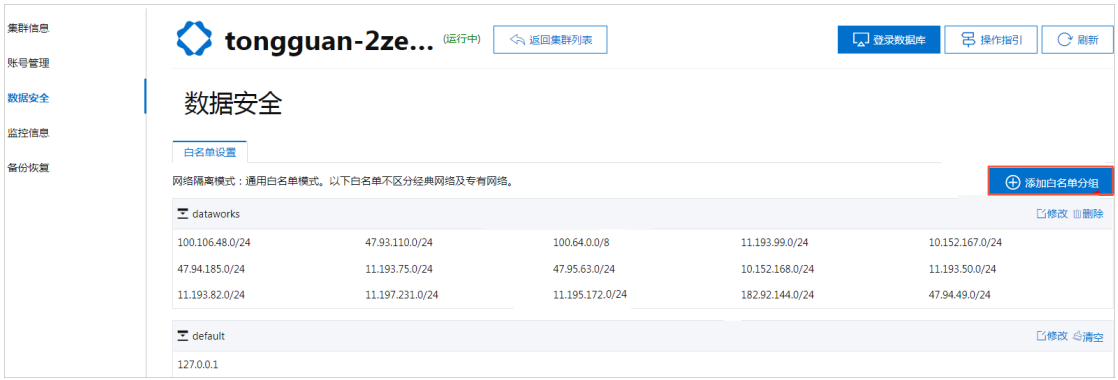
   为了能让DataWorks gateway请求AnalyticDB for MySQL 3.0，需要将独享调度资源组绑定的弹性网卡ip，详情可参考文档新增和使用独享调度资源组，将其设置为AnalyticDB for MySQL 3.0的白名单（AnalyticDB for MySQL 2.0不需要设置）。

2. 设置AnalyticDB for MySQL 3.0白名单

   i. 登录AnalyticDB for MySQL 3.0控制台，进入**集群列表 > 数据安全**页面。

   ii. 单击**添加白名单分组**，将复制的DataWorks白名单粘贴至AnalyticDB for MySQL 3.0中。

新建业务流程

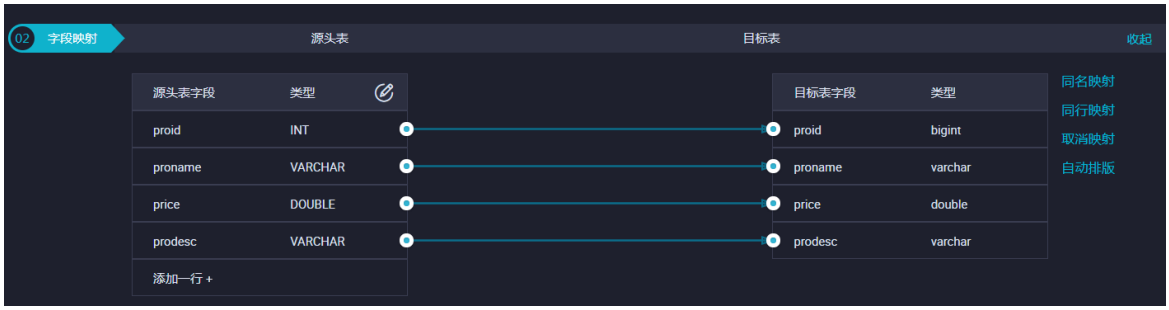
1. 单击左上角的图标，选择**全部产品 > DataStudio（数据开发）**。
2. 在**新建业务流程**对话框中，输入**业务名称**和**描述**。
3. 单击**新建**。

创建离线同步任务

1. 右键单击新建业务流程下的**数据集成**，选择**新建 > 离线同步**。
2. 在**新建节点**对话框中，输入**节点名称**，单击**提交**。
3. 设置**数据来源**和**数据去向**。

4. 选择字段的映射关系。

   左侧的源头表字段和右侧的目标表字段为一一对应关系。单击**添加一行**可以增加单个字段，鼠标放至需要删除的字段上，即可单击**删除**图标进行删除 。



5. 通道控制。

   配置作业速率上限和脏数据检查规则。

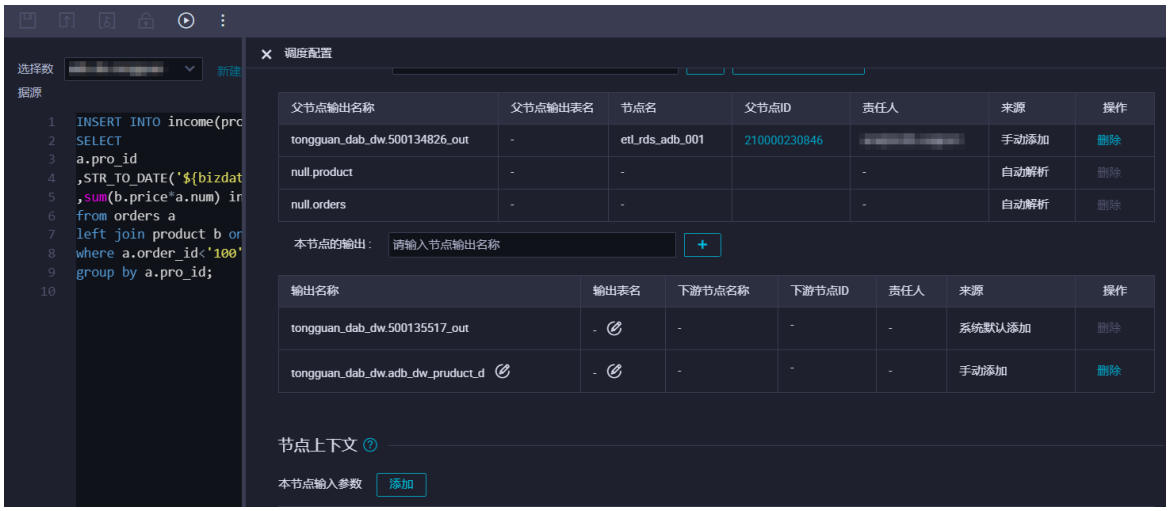
参数	描述
任务期望最大并发数	数据同步任务内，可以从源并行读取或并行写入数据存储端的最大线程数。向导模式通过界面化配置并发数，指定任务所使用的并行度。
同步速率	设置同步速率可以保护读取端数据库，以避免抽取速度过大，给源库造成太大的压力。同步速率建议限流，结合源库的配置，请合理配置抽取速率。
错误记录数	错误记录数，表示脏数据的最大容忍条数。
独享数据集成资源组	选择任务运行的机器，如果任务数比较多，使用默认资源组出现等待资源的情况，建议购买独享数据集成资源或添加自定义资源组，详情请参见新增和使用独享数据集成资源组和新增和使用自定义数据集成资源组。

6. 单击右侧的**调度配置**，为节点配置调度属性。
7. 配置完成后，单击**保存**并**提交**。

新建数据开发任务

1. 右键单击业务流程下的**自定义**，选择**新建 > AnalyticDB for MySQL**。
2. 在**新建节点**对话框中，输入**节点名称**，单击**提交**。
3. 选择相应的数据源后，根据AnalyticDB for MySQL支持的语法，编写SQL语句。通常支持DML语句，您也可以执行DDL语句。

4. 单击右侧的**调度配置**，为节点配置调度属性。



5. 配置完成后，单击**保存**按钮，将其保存至服务器。然后单击**运行**按钮，即可立即执行编辑的SQL语句。

数据运维

提交并发布新建的节点任务后，单击左上角的图标，选择**全部产品 > 运维中心**，即可进行数据运维操作。详情请参见运维中心模块的文档。

元数据管理

您可以单击左上角的图标，选择**全部产品 > 数据地图**，进行元数据管理操作。详情请参见数据地图。

2.10. 使用开放消息和POP接口检查待发布节点

DataWorks支持对成功提交至开发环境的标准模式工作空间节点，在发布到生产环境前，进行待发布状态检查。您可以通过DataWorks提供的检查器机制，根据业务需求判断目标节点是否可以继续后续的发布流程，保障能够发布符合要求的节点任务。本文为您介绍如何使用开放消息和POP接口检查待发布节点。

前提条件

- 开通企业版及以上版本的DataWorks，详情请参见版本服务计费说明。
- 开通消息队列Kafka版，详情请参见计费概述。
- 在DataWorks标准模式工作空间的数据开发页面创建的节点已成功提交。查看节点的提交状态，详情请参见数据开发与运行
- 配置CheckFileDeployment接口，详情请参见CheckFileDeployment。
- 配置GetFileVersion接口，详情请参见GetFileVersion。

背景信息

消息队列Kafka版，用于大数据领域的日志收集、监控数据聚合、流式数据处理、在线和离线分析，详情请参见消息队列Kafka。

DataWorks为您提供了开放消息功能，您可以订阅文件发布检查事件，快速获取并识别事件消息的状态变更信息。当您在DataWorks标准模式工作空间的数据开发页面创建的节点提交成功后，该节点需要发布到生产环境，如果您的DataWorks工作空间配置了检查器，则该节点将进入待发布的检查状态。DataWorks会根据检查器的配置信息将目标节点本次发布的相关信息通过Kafka消息下发至您的服务。您需要根据事件内容判断该节点是否可以继续发布，并调用CheckFileDeployment接口将节点的文件发布检查事件的检查结果返回给DataWorks。当检查校验通过后才可以继续后续的发布流程。校验流程如下图所示。

1. DataWorks数据开发页面创建的节点提交成功，该节点进入待发布状态。DataWorks根据工作空间关联的检查器发起检查流程，节点自动进入待检查状态。

   由于DataWorks的检查器未对外开放，您需要提交工单将检查器的配置信息交由管理员，由管理员统一配置。配置检查器，详情请参见配置检查器。

2. DataWorks的检查器发送文件发布检查事件至Kafka。
3. Kafka将文件发布检查事件转发至用户的服务，进行后续的审批处理。
4. 您可以根据收到的消息进行判断，审批目标节点的发布状态，并通过调用CheckFileDeployment接口，发送审批结果至DataWorks。CheckFileDeployment接口的配置信息，详情请参见CheckFileDeployment。

   您可以通过消息中携带的工作空间ID、节点ID、节点版本等信息，调用GetFileVersion接口查询本次发布的内容，并对该内容进行自动扫描，根据扫描结果判断是否可以正常发布。如果无法通过自动扫描确定是否可以正常发布，则需要使用人工审批流程，进行后续的人工审核。GetFileVersion接口的配置，详情请参见GetFileVersion。

   > **说明** 您可以登录DataWorks控制台，鼠标悬停至顶部菜单栏右侧的用户头像，单击AccessKey管理，获取调用DataWorks OpenAPI需要使用的AccessKey ID及AccessKey Secret。

5. DataWorks根据收到的检查结果，修改目标节点的最终发布状态。发布状态如下：

- OK：表示文件检查通过，可以发布。
- WARN：表示文件检查通过，但是存在警告，文件可以发布。
- FAIL：表示文件检查未通过，不能发布。

> **说明** 如果一个DataWorks工作空间关联了多个检查器，当有待发布的节点时，DataWorks的每一个检查器会启动一个检查流程，每个检查流程均会生成一个checkerInstanceId（即文件检查器所属的实例ID）。每个检查器会分别发送一条Kafka消息至用户，用户需要通过CheckFileDeployment接口，把所有检查器生成的检查结果返回给DataWorks，DataWorks将检查结果汇总，给出最终的检查状态。当所有检查器的检查结果均为通过时，最终检查结果才为通过，本次发布正常进行；只要有检查器的检查结果为不通过，则最终检查结果为不通过，本次发布会被拦截。

DataWorks的检查器机制，常用的功能场景示例如下：

- 检查目标节点的代码中是否包含表变更的语句。您可以通过变更内容，判断该变更是否会修改核心表的结构。如果涉及修改核心表的结构，则您可以拦截此次发布。
- 检查目标节点代码的SQL性能。例如，执行节点任务时，扫描全部表会降低代码性能，如果您希望执行的节点任务SQL性能较好，则可以检查代码中是否包含全表扫描的SQL语句。当包含相关语句时，您可以拦截此次发布。
- 检查目标节点中是否包含账号及密码等敏感信息，当包含敏感时，您可以拦截此次发布，保证敏感信息不被泄露。

使用开放消息和POP接口检查待发布文件的校验步骤如下：

- 开启并配置消息订阅，详情请参见开启并配置消息订阅。
- 配置检查器，详情请参见配置检查器。
- DataWorks向用户发送文件发布检查事件，详情请参见DataWorks向用户发送文件发布检查事件。
- 用户审核文件发布检查事件并返回最终检查结果，详情请参见用户审核文件发布检查事件并返回最终检查结果。

使用限制

- DataWorks仅支持企业版及以上版本使用开放消息和POP接口检查待发布节点。
- DataWorks仅支持检查标准模式工作空间中的待发布节点。

开启并配置消息订阅

1. 进入**开放平台**开启消息订阅服务。
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**开放平台**。
   iii. 在**开放消息**页签，打开**启动消息订阅**开关，即可开启消息订阅服务。

   > **说明**

2. 新建消息分类Topic。
   i. 在**开放消息**页签，单击**新建Topic**。
   ii. 在**新建Topic**对话框，输入Topic名称及描述信息。

   > **说明** 该Topic主要描述消息的主题，用于消息分类。Topic名称一旦创建，无法修改。

iii. 单击**确定**，完成Topic的创建。

3. 配置Topic的订阅信息。

i. 在所创建Topic的**操作**列，单击**订阅设置**。

ii. 在**Topic订阅内容设置**对话框，选择需要订阅的工作空间，并勾选**订阅事件类型**为**文件发布检查事件**。

> **说明** 一个Topic最多可以订阅5个工作空间。

配置检查器

由于DataWorks的检查器未对外开放，您需要提交工单将检查器的配置信息交由管理员，由管理员统一配置。

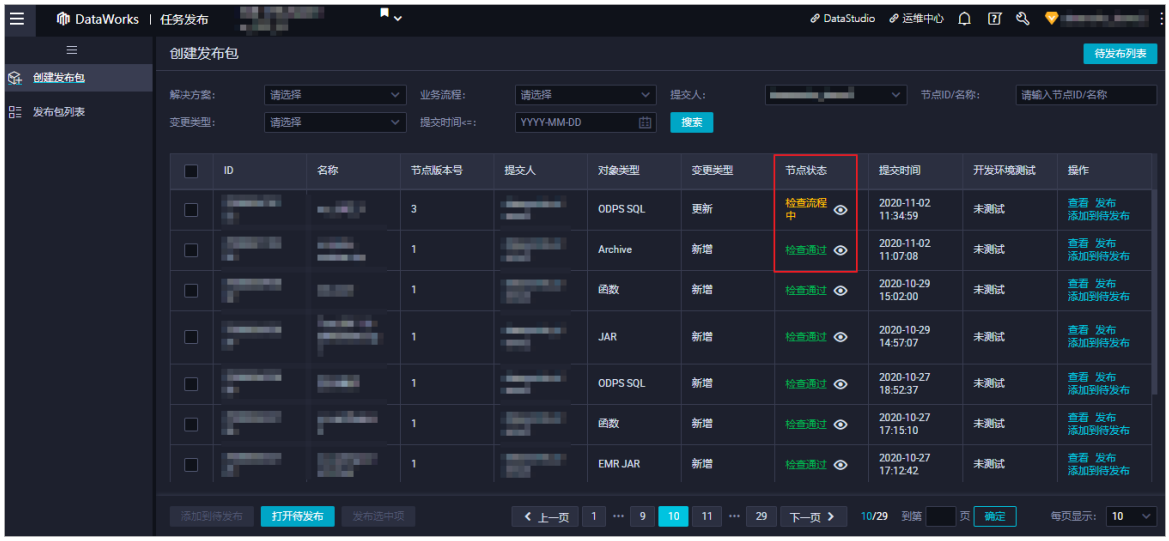
配置检查器需要提供的配置项信息如下表所示。

配置项	描述
检查器的唯一标识符	标识符自定义。检查流程中的Kafka消息会携带该标识符，方便您使用该标识符来判断所使用的检查器。 检查器的唯一标识符需要以英文字母开头，只能包含英文字母、数字及下划线（_），最多支持64个字符，并且确定后不可变更。 检查器的唯一标识符对应收到的Kafka消息中的checkerIdentify字段。Kafka消息的字段，详情请参见DataWorks向用户发送文件发布检查事件。
检查器的名称	名称自定义。用于在检查详情页，查看本次发布被哪个检查器所拦截。
检查器责任人	阿里云的账号ID。您可以登录DataWorks控制台，鼠标悬停至顶部菜单栏右侧的用户头像，查看账号ID。方便当节点任务被检查器拦截时，可以定位到对应的检查器责任人。
检查器描述信息	最多支持600个字符。

配置项	描述
检查器检查流程页面地址	如果您使用的是非全自动的检查流程，而是在检查流程发起之后，需要人为执行审批操作的检查流程，则需要提供人工审核的页面地址。 该地址会配置在检查页面的**进入详情页面**，引导用户提交审批流程。跳转地址格式为 **您所提供的人工审核页面地址**+`checkerInstanceId` 。例如，提供的人工审核的页面地址为 `https://www.aliyun.com/` ，本次检查器的实例ID（checkerInstanceId）为 `82_1235776432_3` ，则最终的跳转地址为 `https://www.aliyun.com/82_1235776432_3` 。 **checkerInstanceId**为每次自动检查待发布节点的检查器流程ID，用于串联一次发布流程，系统自动生成，不可修改。
需要拦截的文件类型列表	系统会根据配置的文件类型选择性拦截，不配置则默认拦截所有类型的文件。

如果您配置了检查器，当目标节点提交成功后，您可以执行如下操作：

1. 进入**任务发布**页面，查看节点的检查状态。进入**任务发布**页面，详情请参见发布标准模式工作空间的节点。



2. 单击目标节点状态，查看节点的检查详情。

   您可以查看DataWorks工作空间关联的哪个检查器阻碍了发布流程。如果使用的是您配置的检查器，单击目标检查器后的**进入详情页面**，则会跳转至您配置检查器时所提供的*检查器检查流程页面地址*。

DataWorks向用户发送文件发布检查事件

DataWorks向Kafka发送文件发布检查事件后，Kafka会将其转发至用户的服务，用于后续用户进行审批操作。事件的参数及描述信息格式如下。

```
{
    "isContentChange": true,
    "baseId": "**************",
    "appId": ******,
    "tenantId": ***************,
    "cloudUuid": *************,
    "type": ***,
    "fileName": "********",
    "owner": "****************",
    "projectOwner": "*************",
    "projectName": "************",
    "checkerIdentify": "*************",
    "checkerInstanceId": "***************",
    "changeType": "UPDATE",
    "useType": 0,
    "fileCreateTime": "2021-01-15 14:03:02",
    "fileId": **********,
    "connName": "*******",
    "fileVersion": 3
}
```

参数描述如下表所示。

参数	描述
isContentChange	相较于前一次文件内容是否有变动。取值如下： • true：文件内容变动。 • false：文件内容未变动。
baseId	提交节点任务的用户ID。
appId	DataWorks工作空间的ID。 您可以调用GetFileVersion获取目标节点本次发布的详细内容，appId、fileId、fileVersion可以作为GetFileVersion接口的入参信息。

参数	描述
tenantId	租户ID。
cloudUuid	调度节点的ID。
type	提交节点的节点类型。
fileName	提交节点的节点名称。
owner	提交节点对应节点责任人的阿里云账号ID。
projectOwner	DataWorks工作空间的Owner。
projectName	DataWorks工作空间的名称。
checkerIdentify	检查器的唯一标识。
checkerInstanceId	检查器实例的唯一标识。
changeType	提交节点的变更类型。取值如下： • UPDATE：节点内容有更新。 • DELETE：节点内容有删除。 • CREATE：节点有新建的内容。
useType	功能模块。例如，数据开发页面的**业务流程（0）**、**手动业务流程（2）**、**临时查询（10）**等。
fileCreateTime	提交节点的创建时间。
fileId	提交节点的节点ID。 您可以调用GetFileVersion获取目标节点本次发布的详细内容，**appId**、**fileId**、**fileVersion**可以作为GetFileVersion接口的入参信息。
connName	提交的节点所使用的数据源。
fileVersion	提交节点的版本。 您可以调用GetFileVersion获取目标节点本次发布的详细内容，**appId**、**fileId**、**fileVersion**可以作为GetFileVersion接口的入参信息。

用户审核文件发布检查事件并返回最终检查结果

当您接收到Kafka推送的文件发布检查事件后，会对该事件的内容进行审核，并调用CheckFileDeployment接口，将审核结果及检查器的检查流程页面地址返回给DataWorks，DataWorks根据收到的审核结果修改目标节点的最终发布状态。当最终检查结果为通过时，则本次发布正常进行；当最终检查结果为不通过时，则本次发布会被拦截。CheckFileDeployment接口的配置，详情请参见CheckFileDeployment。

说明 如果一个DataWorks工作空间关联了多个检查器，当有待发布的节点时，DataWorks的每一个检查器会启动一个检查流程，每个检查流程均会生成一个checkerInstanceId（即文件检查器所属的实例ID）。每个检查器会分别发送一条Kafka消息至用户，用户需要通过CheckFileDeployment接口，把所有检查器生成的检查结果返回给DataWorks，DataWorks将检查结果汇总，给出最终的检查状态。当所有检查器的检查结果均为通过时，最终检查结果才为通过，本次发布正常进行；只要有检查器的检查结果为不通过，则最终检查结果为不通过，本次发布会被拦截。

如下代码为您展示了，当用户收到Kafka发送的文件发布检查事件后，检查本次发布中是否包含表变更，并调用CheckFileDeployment接口返回审批结果至DataWorks的流程示例。

```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.dataworks_public.model.v20200518.CheckFileDeploymentRequest;
import com.aliyuncs.dataworks_public.model.v20200518.GetFileVersionRequest;
import com.aliyuncs.dataworks_public.model.v20200518.GetFileVersionResponse;
import com.aliyuncs.profile.DefaultProfile;
import com.google.gson.Gson;
import org.apache.kafka.clients.CommonClientConfigs;
import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.common.config.SaslConfigs;
import org.apache.kafka.common.config.SslConfigs;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;
public class DataStudioKafkaChecker {
    private static final String KAFKA_TOPIC; //表示在DataWorks控制台注册的Kafka消息Topic。
    private static final String DW_API_AK_ID; //调用DataWorks OpenAPI所需的AccessKey ID。
    private static final String DW_API_AK_SECRET; //调用DataWorks OpenAPI所需的AccessKey Sec
ret。
    public static void main(String[] args) {
        /*
         * 初始化并启动Kafka消息的Consumer。
         */
        KafkaConsumer<String, String> consumer; //初始化Consumer。
        //设置消费组（GROUP_ID_CONFIG）订阅的Topic，您可以订阅多个Topic。
        //如果消费组相同，则订阅的Topic也建议设置为相同的Topic。
        List<String> subscribedTopics =  new ArrayList<String>();
        //如果需要订阅多个Topic，则使用Add语法添加即可。
        //您需要登录DataWorks控制台创建Topic。
        subscribedTopics.add(KAFKA_TOPIC);
        consumer.subscribe(subscribedTopics);
        //初始化DataWorks OpenAPI的客户端（如下代码以杭州地域示例）。
        DefaultProfile.addEndpoint("cn-hangzhou", "dataworks-public", "dataworks.cn-hangzho
u.aliyuncs.com");
        IAcsClient client = new DefaultAcsClient(DefaultProfile.getProfile("cn-hangzhou", D
W_API_AK_ID, DW_API_AK_SECRET));
        Gson gson = new Gson();
        //循环消费消息。
        while (true){
```

```
            try {
                ConsumerRecords<String, String> records = consumer.poll(1000);
                //必须在下次poll之前消费完这些数据，且总耗时不得超过SESSION_TIMEOUT_MS_CONFIG。
                //建议使用单独的线程池来消费消息，并异步返回结果。
                for (ConsumerRecord<String, String> record : records) {
                    KafkaMessage kafkaMessage = gson.fromJson(record.value(), KafkaMessage.
class);
                    CheckerMessage checkerMessage = kafkaMessage.messageBody;
                    //根据消息中的信息检查本次发布的内容。
                    GetFileVersionRequest getFileVersionRequest = new GetFileVersionRequest
();
                    getFileVersionRequest.setFileId(checkerMessage.fileId);
                    getFileVersionRequest.setFileVersion(checkerMessage.fileVersion);
                    getFileVersionRequest.setProjectId(checkerMessage.appId);
                    GetFileVersionResponse getFileVersionResponse = client.getAcsResponse(g
etFileVersionRequest);
                    System.out.println(getFileVersionResponse.getData().getFileContent());
                    //检查发布节点的代码中是否包含表变更的代码。
                    //如果包含create table语句，则检查不通过
                    //如果包含alter table语句，则会产生告警。
                    //如果不包含create table或alter table语句，则检查通过。
                    //执行的具体业务规则，可以根据您的业务需求定制。
                    CheckFileDeploymentRequest checkFileDeploymentRequest = new CheckFileDe
ploymentRequest();
                    checkFileDeploymentRequest.setCheckerInstanceId(checkerMessage.checkerI
nstanceId);
                    if (getFileVersionResponse.getData().getFileContent().toLowerCase().con
tains("create table")) {
                        checkFileDeploymentRequest.setStatus("FAIL");
                    } else if (getFileVersionResponse.getData().getFileContent().toLowerCas
e().contains("alter table")) {
                        checkFileDeploymentRequest.setStatus("WARN");
                    } else {
                        checkFileDeploymentRequest.setStatus("OK");
                    }
                    client.getAcsResponse(checkFileDeploymentRequest);
                }
            } catch (Exception e) {
                try {
                    Thread.sleep(1000);
                } catch (Throwable ignore) {
                }
                e.printStackTrace();
            }
        }
    }
    private class KafkaMessage {
        public CheckerMessage messageBody;
    }
    private class CheckerMessage {
        public String owner;
        public String fileName;
        public String checkerIdentify;
        public String changeType;
```

```
        public String connName;
        public Long cloudUuid;
        public String baseId;
        public Integer type;
        public Integer useType;
        public boolean isContentChange;
        public String fileCreateTime;
        public String checkerInstanceId;
        public Long appId;
        public Long tenantId;
        public Integer fileVersion;
        public Long fileId;
    }
}
```

3.数据安全

3.1. 实现指定用户访问特定UDF最佳实践

本文为您介绍如何实现将资源（表、UDF等）设置为仅能被指定的用户访问。此方法涉及数据的加密解密算法，属于数据安全管控范畴。

前提条件

您需要提前安装MaxCompute客户端，以实现指定UDF被指定用户访问的操作。详情请参见使用客户端（odpscmd）连接。

背景信息

设置用户访问权限的常见方法有如下几种：

- Package方案，通过打包授权进行权限精细化管控。

  Package用于解决跨项目空间的数据共享及资源授权问题。通过Package授予用户开发者角色后，用户拥有所有权限，风险不可控。详情请参见基于Package跨项目访问资源。

  - 下图为DataWorks开发者角色的权限。

    ```
    odps@ :desc role role_project_dev;

    Authorization Type: Policy
    A       projects/sz_mc: *
    A       projects/sz_mc/instances/*: *
    A       projects/sz_mc/jobs/*: *
    A       projects/sz_mc/offlinemodels/*: *
    A       projects/sz_mc/packages/*: *
    A       projects/sz_mc/registration/functions/*: *
    A       projects/sz_mc/resources/*: *
    A       projects/sz_mc/tables/*: *
    A       projects/sz_mc/volumes/*:
    ```

    由上图可见，开发者角色对工作空间中的Package、Functions、Resources和Table默认有全部权限，不符合权限配置的要求。

  - 下图为通过DataWorks添加子账号并赋予开发者角色的权限。

    ```
    odps@ :show grants for RAM$y .pt@aliyun-test.com:ramtest;

    [roles]
    role_project_dev

    Authorization Type: Policy
    [role/role_project_dev]
    A       projects/sz_mc: *
    A       projects/sz_mc/instances/*: *
    A       projects/sz_mc/jobs/*: *
    A       projects/sz_mc/offlinemodels/*: *
    A       projects/sz_mc/packages/*: *
    A       projects/sz_mc/registration/functions/*: *
    A       projects/sz_mc/resources/*: *
    A       projects/sz_mc/tables/*: *
    A       projects/sz_mc/volumes/*: *
    ```

由此可见，通过打包授权和DataWorks默认的角色都不能满足特定用户访问指定UDF的需求。例如，授予子账号 `RAM$xxxxx.pt@example.com:ramtest` 开发者角色，则默认该子账号拥有当前工作空间中全部对象的所有操作权限，详情请参见用户授权。

- 在DataWorks中新建角色进行权限管控。

  进入DataWorks的**工作空间管理 > MaxCompute高级配置**页面，在左侧导航栏上单击**自定义用户角色**进行权限管控。但是在**MaxCompute高级配置**中，只能针对某个表或项目进行授权，不能对资源和UDF进行授权。

- Role Policy结合Project Policy实现指定用户访问指定UDF。

  通过Policy可以精细化地管理具体用户对特定资源的具体权限粒度。

  > **说明** 为了安全起见，建议初学者使用测试项目来验证Policy。

因此您可以通过Policy方案实现特定UDF被指定用户访问：

- 如果您不想让其他用户访问工作空间内具体的资源，在DataWorks中添加数据开发者权限后，再根据Role Policy的操作，在MaxCompute客户端将其配置为拒绝访问权限。
- 如果您需要指定用户访问指定资源，通过Role Policy在DataWorks中配置数据开发者权限后，再根据Project Policy的操作，在MaxCompute客户端将其配置为允许访问权限。

操作步骤

1. 创建默认拒绝访问UDF的角色。
   i. 在客户端输入如下命令创建角色denyudfrole。

   ```
   create role denyudfrole;
   ```

   ii. 创建Policy授权文件，如下所示。

   ```
   {
   "Version": "1", "Statement"
   [{
   "Effect":"Deny",
   "Action":["odps:Read","odps:List"],
   "Resource":"acs:odps:*:projects/sz_mc/resources/getaddr.jar"
   },
   {
   "Effect":"Deny",
   "Action":["odps:Read","odps:List"],
   "Resource":"acs:odps:*:projects/sz_mc/registration/functions/getregion"
   }
    ] }
   ```

   iii. 设置Role Policy。

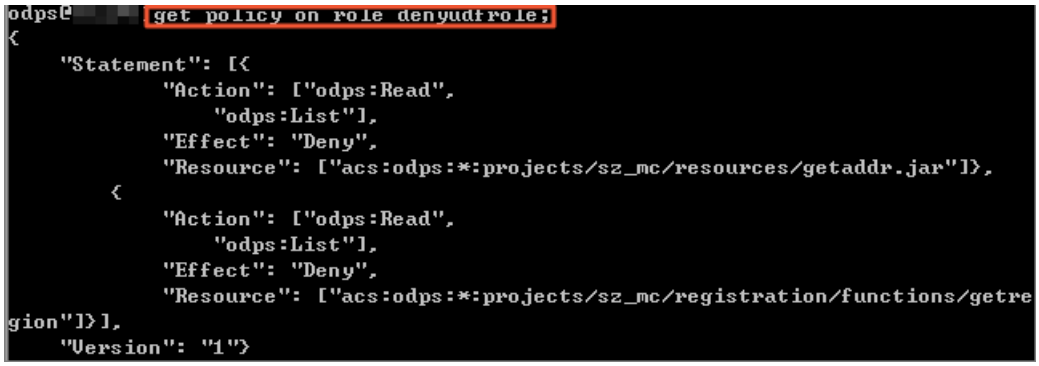
   在客户端执行如下命令，设置Role Policy文件的存放路径。

   ```
   put policy /Users/yangyi/Desktop/role_policy.json on role denyudfrole;
   ```

iv. 在客户端执行如下命令查看Role Policy。

```
get policy on role denyudfrole;
```

返回结果如下。

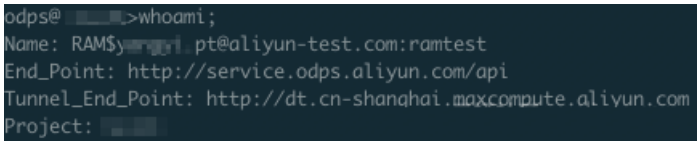


v. 在客户端执行如下命令添加子账号至role denyudfrole。

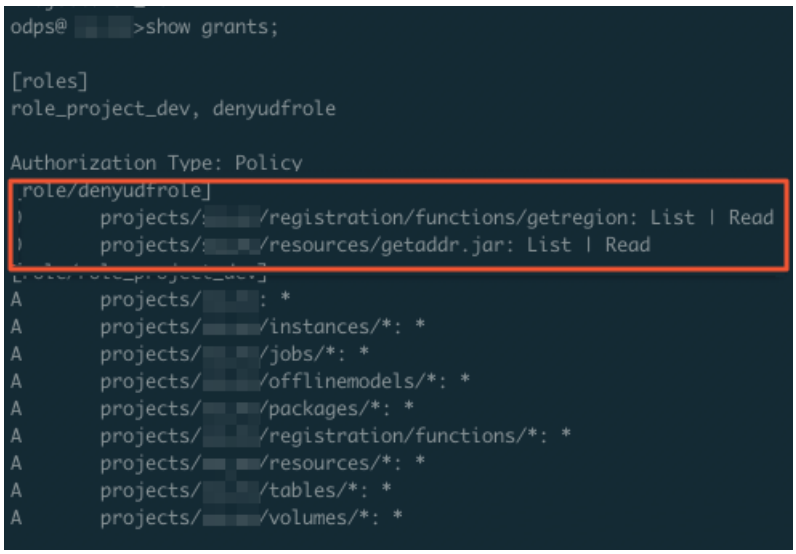
```
grant denyudfrole to RAM$xxxx.pt@example.com:ramtest;
```

2. 验证拒绝访问UDF的角色是否创建成功。

i. 登录客户端输入 `whoami;` 确认角色。

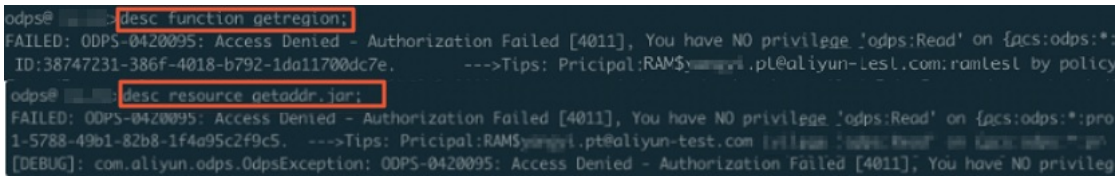


ii. 通过 `show grants;` 查看当前登录用户权限。



通过查询发现该RAM子账号有两个角色，一个是role_project_dev（即DataWorks默认的开发者角色），另一个是刚自定义创建的denyudfrole。

iii. 验证自建UDF以及依赖的包的权限。



通过上述验证发现，该子账号在拥有DataWorks开发者角色的前提下，没有自建UDF（getregion）的读权限。您还需要结合Project Policy来实现该UDF只能被指定的用户访问。

3. 配置Project Policy。

i. 编写Policy。

```
{
"Version": "1", "Statement":
[{
"Effect":"Allow",
"Principal":"RAM$yangyi.pt@example.com:yangyitest",
"Action":["odps:Read","odps:List","odps:Select"],
"Resource":"acs:odps:*:projects/sz_mc/resources/getaddr.jar"
},
{
"Effect":"Allow",
 "Principal":"RAM$xxxx.pt@example.com:yangyitest",
"Action":["odps:Read","odps:List","odps:Select"],
"Resource":"acs:odps:*:projects/sz_mc/registration/functions/getregion"
}] }
```

ii. 设置Project Policy。

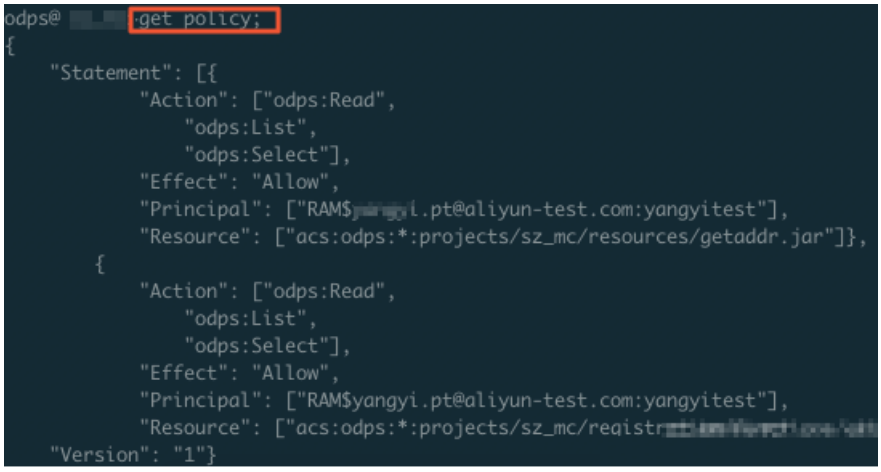
在客户端执行如下命令，设置Project Policy文件的存放路径。

```
put policy /Users/yangyi/Desktop/project_policy.json;
```

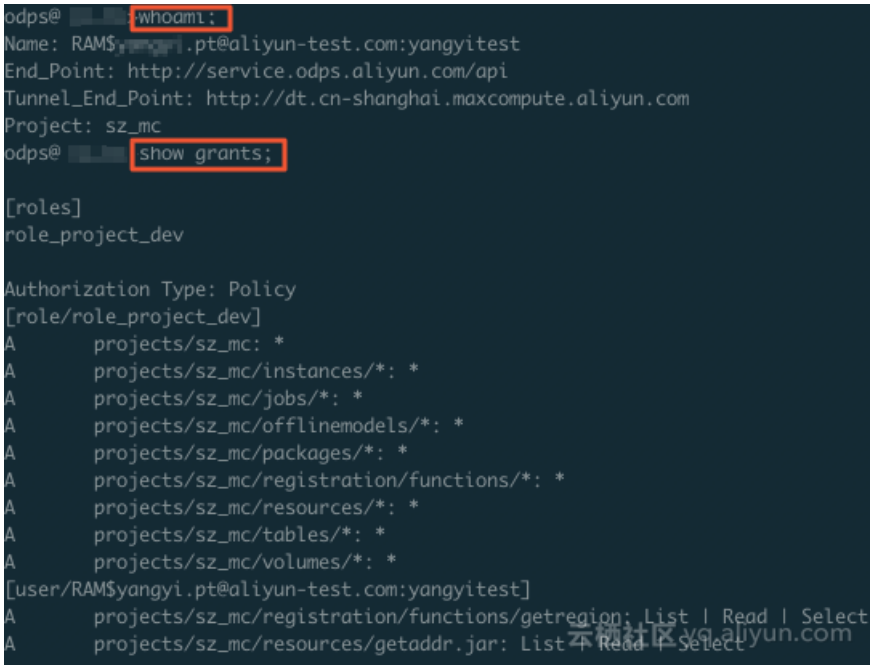
iii. 在客户端执行如下命令查看Project Policy。

```
get policy;
```

返回结果如下。



iv. 通过 `whoami;` 和 `show grants;` 进行验证。



v. 运行SQL任务，查看是否仅指定的RAM子账号能够查看指定的UDF和依赖的包。

- 指定的RAM子账号查看指定的UDF。

- 查看依赖包。

3.2. RAM用户仅从特定IP登录DataWorks

本文为您介绍如何实现RAM用户仅从特定的本地IP登录DataWorks。

前提条件

您需要首先创建RAM用户并进行授权，详情请参见准备RAM用户。权限AliyunDataWorksFullAccess为系统默认权限，无法修改，您需要额外创建一个自定义授权策略。

创建自定义策略

1. 云账号登录RAM控制台。
2. 在左侧导航栏，单击**权限管理 > 权限策略管理**。

3. 单击**创建权限策略**。
4. 在**新建自定义权限策略**页面，输入**策略名称**为dataworksIPlimit1，选中**配置模式**为**脚本配置**，配置您的自定义权限。

   自定义权限的完整内容如下所示，"acs:SourceIP"后填写的IP，便是您允许访问DataWorks的IP（支持设置多个IP）。自定义权限配置项说明请参见Policy基本元素。

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "dataworks:*"
            ],
            "Resource": [
                "acs:dataworks:*:*:*"
            ],
            "Condition": {
                "NotIpAddress": {
                    "acs:SourceIp": [
                        "10.0.0.0",
                        "192.168.0.0"
                    ]
                }
            }
        }
    ]
}
```

5. 单击**确定**。

授权自定义策略给RAM用户

1. 在左侧导航栏的**人员管理**菜单下，单击**用户**。
2. 在**用户登录名称/显示名称**列表下，单击目标RAM用户名称。
3. 在**权限管理**页签下，单击**添加权限**，被授权主体会自动填入。
4. 单击**自定义策略**页签。
5. 在左侧**权限策略名称**列表下，单击需要授予RAM用户的权限策略。

   > **说明** 在右侧区域框，选择某条策略并单击×，可撤销该策略。

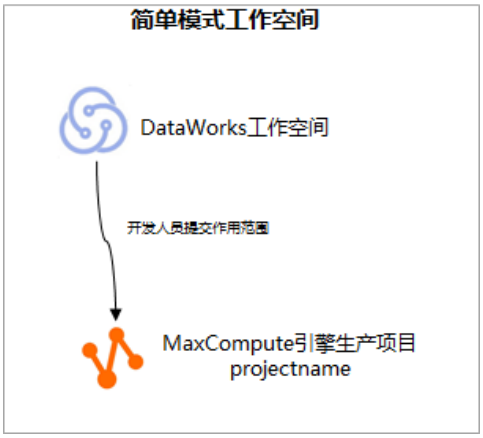
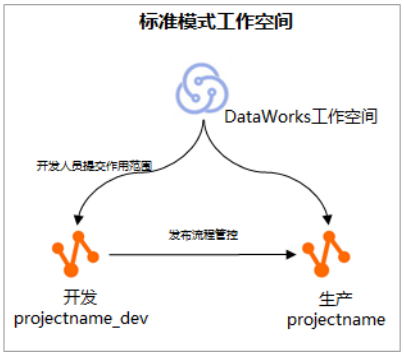
6. 单击**确定**。
7. 单击**完成**。

3.3. 权限管理与规范化数据开发

DataWorks工作空间类型分为“简单模式”、“标准模式”两种，两种类型在权限管理上有细分区别。本实践将基于DataWorks标准模式空间完成从“数据建模”到“数据生产”的基本流程，帮助您快速掌握规范化的数据体系建设流程，提升在数据开发过程中的规范性、安全性、稳定性。

背景信息

DataWorks采取RBAC权限模型供用户管理DataWorks所有页面可见功能以及API的使用权限，同时这套权限体系与MaxCompute的RBAC角色体系存在天然的映射关系，详情可参见角色及成员管理：空间级。不同工作空间类型的权限管理特征与优缺点不一致，以下表格为您对比介绍两种空间类型的权限细分特点。

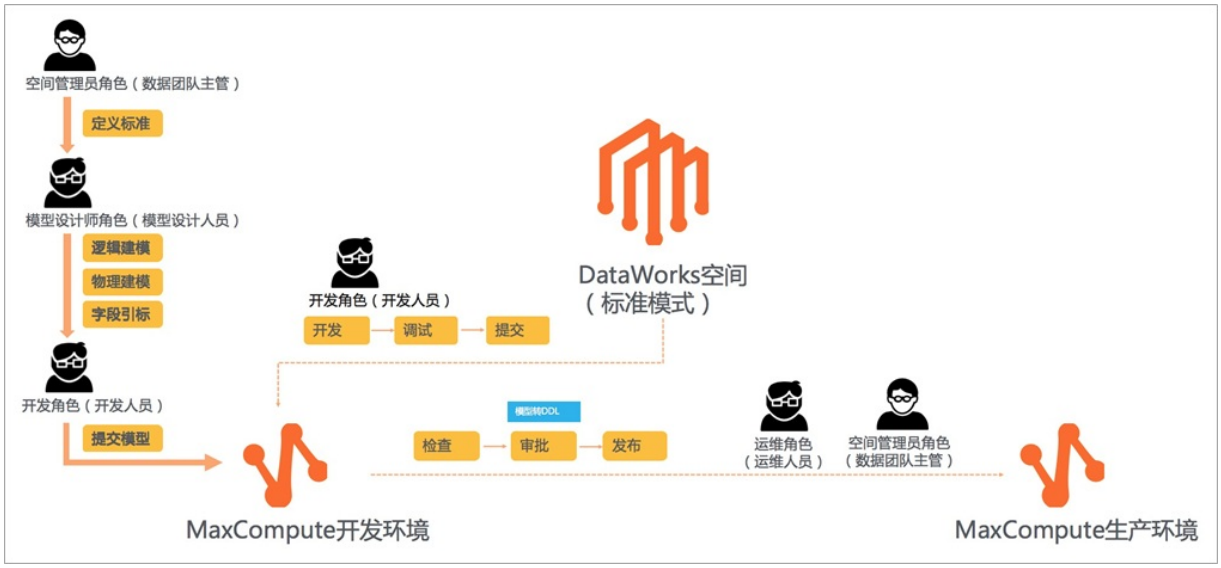
细分特点	简单模式	标准模式
描述	在简单模式工作空间下，一个DataWorks空间下层对应一个MaxCompute项目（或一个EMR集群、Hologres数据库等），该环境即视为生产（PROD）环境。  	在标准模式工作空间下，一个DataWorks空间下层对应两个MaxCompute项目（或两个EMR集群、Hologres数据库等），一个视为开发（DEV）环境，一个视为生产（PROD）环境。 
权限概述	在简单模式空间下，DataWorks的“开发”角色因为与所绑定MaxCompute项目的“Role_Project_Dev” Role进行了映射，因此DataWorks开发角色天然能够读取MaxCompute项目内的所有数据。	在标准模式空间下，DataWorks的“开发”角色因为与所绑定MaxCompute项目（dev环境）的“Role_Project_Dev” Role进行了映射，因此： • DataWorks开发角色天然能够读取MaxCompute项目（dev环境）内的所有数据。 • 由于没有和MaxCompute项目（PROD环境）的role映射，因此默认情况下DataWorks开发角色无MaxCompute（PROD环境）的数据权限。
优点	简单、方便、易用。 仅需要授权数据开发人员“DataWorks开发角色”即可完成所有数据仓库开发工作。	安全、规范。 • 具备安全、规范的代码发布管控流程（包含代码评审、代码DIFF查看等功能），保障生产环境稳定性，避免不必要的因代码逻辑引起的脏数据蔓延或任务报错等非预期情况。 • 数据访问得到有效管控，数据安全得以保障。

细分特点	简单模式	标准模式
缺点	存在不稳定、不安全的风险。 • 开发角色可以不经过任何人审批，随时新增、修改代码并提交至调度系统，给生产环境带来不稳定因素。 • 面向MaxCompute计算引擎时，开发角色默认拥有当前MaxCompute项目所有表的读写权限，可随意对表进行增加、删除和修改等操作，存在数据安全风险。	流程相对复杂，一般情况下无法一人完成所有数据开发、生产流程。

> **说明** 关于简单模式与标准模式差异详情可参考文档：简单模式和标准模式的区别

标准模式对使用流程的影响

如图，标准模式“生产、开发隔离”的模式将影响数据模型设计、数据处理逻辑代码发布等流程。



实践操作流程

以下以一个具体的实践为您演示，标准模式下规范化的数据开发流程。

1. Step1：开通产品与创建空间
2. Step2：角色管理
3. Step3：权限管理
4. Step4：数据建模
5. Step5：数据开发与生产

Step1：开通产品与创建空间

1. 开通DataWorks与MaxCompute。

   进入阿里云组合产品页面，选择**计算引擎组合购买 > DataWorks+MaxCompute**。

   DataWorks与MaxCompute的组合为阿里巴巴数据中台建设的黄金搭档，可为您快速落地企业大数据开发与治理平台。如需开通其他类型组合，可进入DataWorks组合售卖自行选购。

2. 开通DataWorks数据建模（DDM）。

   进入数据建模

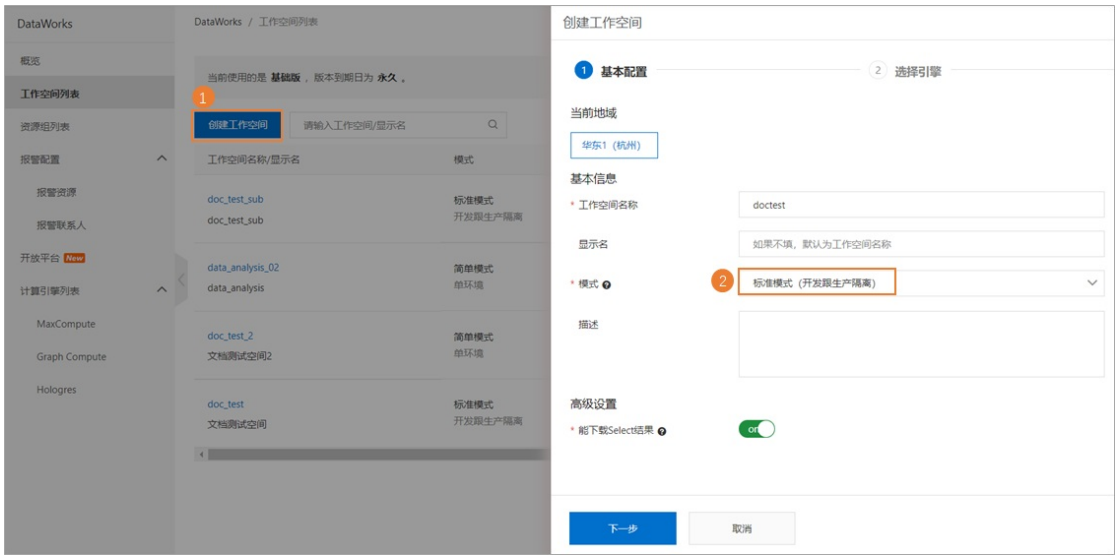
   阿里云DataWorks联合建模工具DDM（Datablau Data Modeler）为您提供一体化的数据建模解决方案，我们将数据模型设计管控、引标落标等能力融入DataWorks规范化开发流程，助力用户实现数据资产价值化输出，在数据全生命周期上夯实数据基础，为企业数据价值化提供有力支撑。

3. 创建DataWorks工作空间。

   > **说明** 阿里云主账号可直接操作创建工作空间，如需使用RAM用户创建工作空间，则需对RAM用户授权后再操作，详情可参见准备RAM用户。

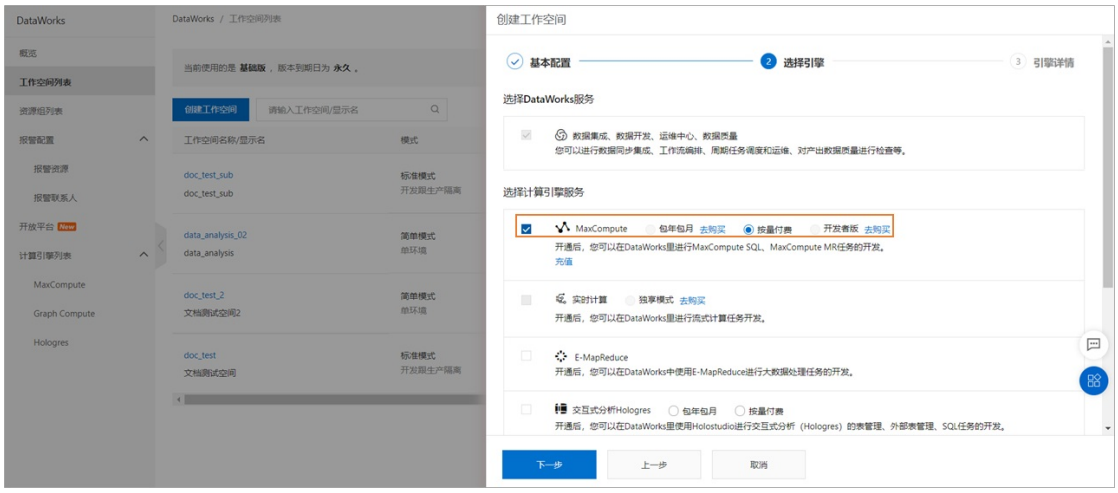
   i. 配置工作空间基本信息。

   进入DataWorks控制台，在**工作空间列表**页面中单击**创建工作空间**，选择工作空间的模式为**标准模式（开发跟生产隔离）**，其他参数根据界面提示配置，完成后单击**下一步**。



   ii. 为工作空间绑定MaxCompute引擎。

   勾选MaxCompute引擎，并根据实际情况选择计费方式，完成后单击**下一步**。选择好计费方式后，后续您在当前工作空间执行MaxCompute任务时，会采用此处配置的计费方式进行计费收费。

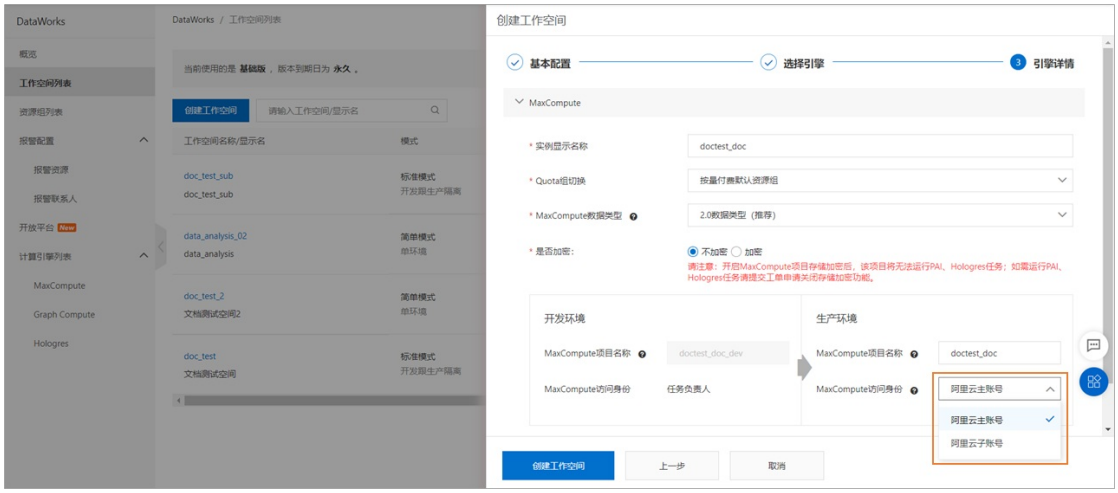


iii. 配置引擎详情。

根据界面提示配置引擎的显示名称等详情。

其中生产环境的MaxCompute访问身份即调度访问身份，是开发任务发布到生产环境进行周期性调度运行时所使用的身份，通常情况下为保证调度任务顺利进行，比起开发者自己的身份来，调度访问身份往往拥有较大数据范围读写权限。

生产环境MaxCompute访问身份默认情况下为阿里云主账，如您存在不同部门使用不同空间并要求调度访问身份权限隔离，则可以选择独立RAM用户作为该空间专属调度身份。



Step2：角色管理

在DataWorks工作空间下，您可以根据不同的业务场景为不同的RAM用户设置不同的工作空间权限，DataWorks目前有**预设角色**，同时也支持您**自定义角色**。您在将RAM用户添加到DataWorks工作空间时，可以选择添加为当前工作空间的**预设角色**或者**自定义角色**。

> **说明** DataWorks权限体系与MaxCompute权限体系是相互独立的，某个用户拥有DataWorks使用权限不代表一定拥有MaxCompute使用权限，但工作空间的两种模式权限的情况例外，如背景信息所示。

1. 角色规划。
   - **预设角色**

     DataWorks提供预设角色供用户进行快速授权，具体权限请参见附录：预设角色权限列表（空间级），除此之外，预设角色权限控制大体可以参考以下说明。

     - 空间级别-空间管理员角色：可以做当前空间所有操作。
     - 空间级别-开发角色：可以做任务开发的工作，如DataStudio任务开发、数据服务开发、DQC配置等。
     - 空间级别-运维角色：负责配置资源与任务部署，如配置数据源、发布任务。
     - 空间级别-部署角色：仅负责发布任务。
     - 空间级别-模型设计师角色：仅能操作建模工作，无法定义数据标准。

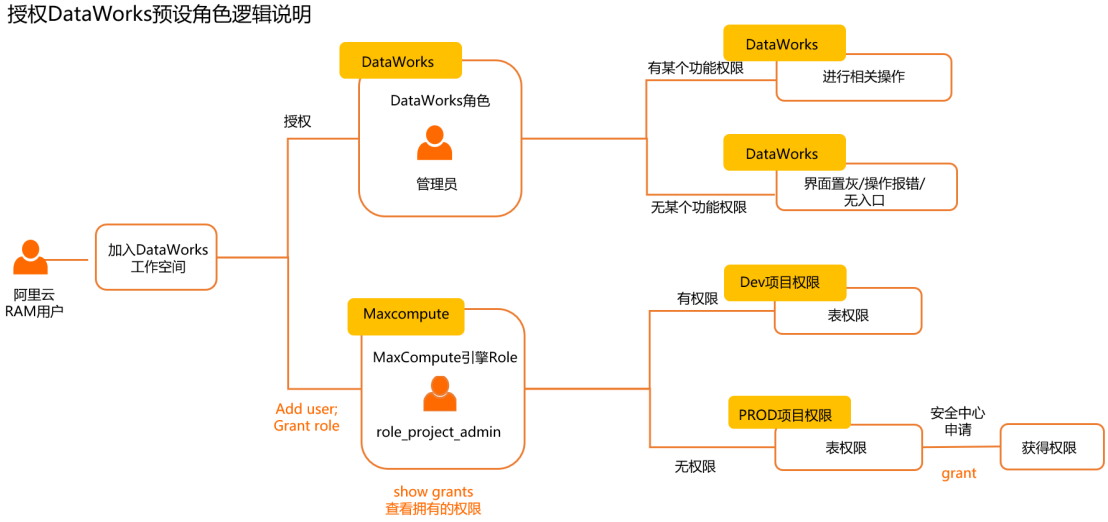
     同时，为方便用户进行数据开发工作，这些DataWorks预设角色与所绑定的MaxCompute项目Role存在映射关系，详情请参见背景信息。

     以下以一个案例为您示意。

- DataWorks上给一个RAM用户授予开发角色，他可以在DataWorks上开发代码并且提交，但是他不能将代码直接发布到生产，发布生产操作需要有运维权限（项目所有者、管理员、运维角色拥有此权限）。
- 在MaxCompute引擎层面，DataWorks上为RAM用户授予开发角色这个操作，在MaxCompute引擎层面就是为这个RAM用户授予了一个role_project_dev这个角色，这个角色会被赋予一些当前MaxCompute项目的表和项目的权限。

> **说明** 查看MaxCompute Role所拥有权限请参考用户规划与管理。

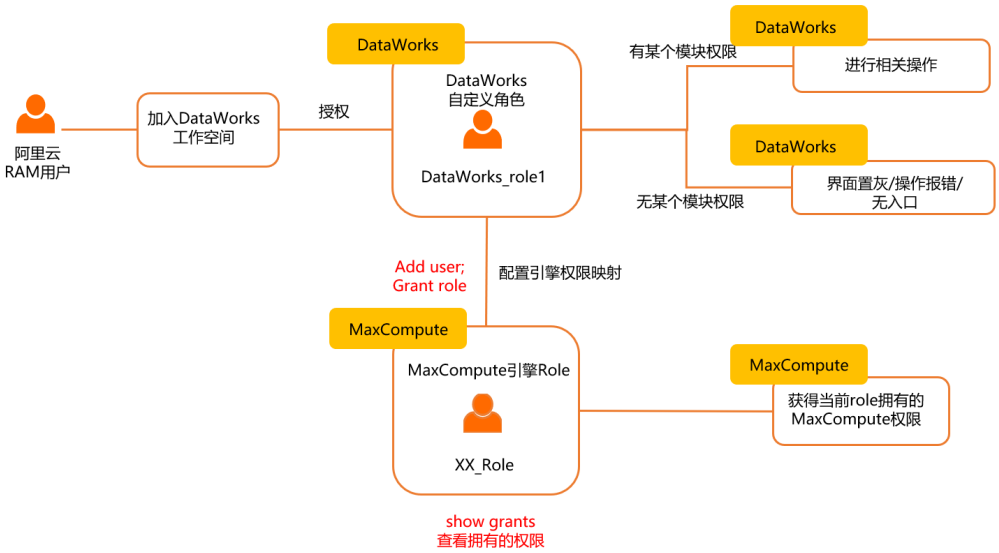
授权DataWorks预设角色逻辑说明



- **DataWorks自定义角色**

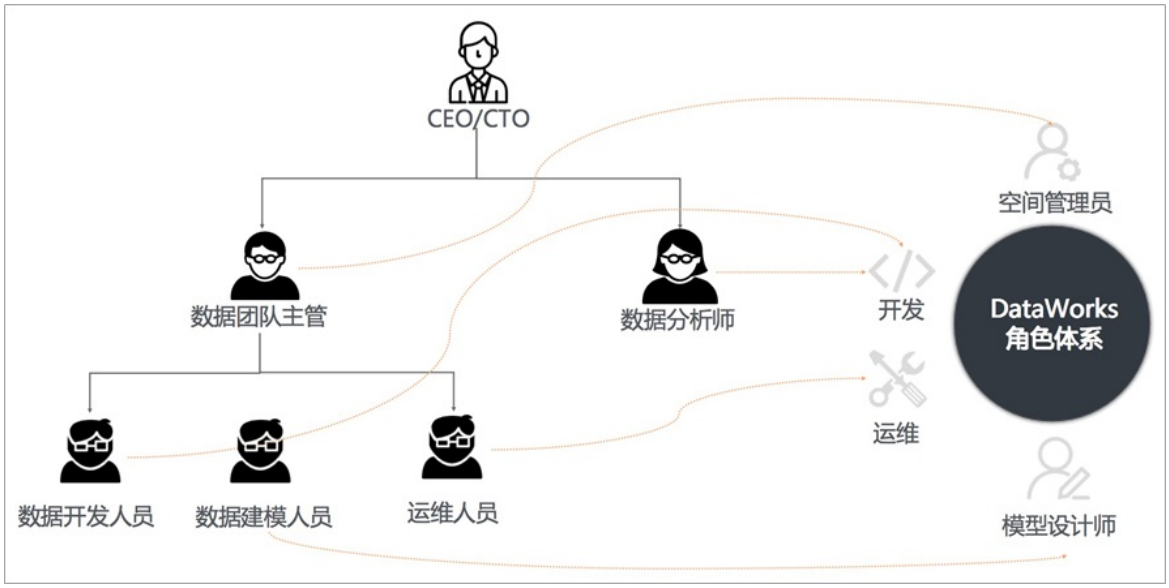
  您可以通过自定义角色控制是否让某个RAM用户只能访问部分模块，同时您也可以自行控制配置引擎权限映射操作给该自定义角色默认的底层MaxCompute引擎相关的权限。

  授权DataWorks自定义角色逻辑说明



2. 角色分配。

   在本案例中，您需要准备至少5个RAM用户，并对其赋予如下几种角色：



授权步骤请参考用户授权与管理，其中：

- 数据团队主管被赋予“空间管理员”权限。
- 数据开发人员被赋予“开发角色”权限。
- 数据建模人员被赋予“模型设计师角色”权限。
- 运维人员被赋予“运维角色”权限。
- 数据分析师人员被赋予“开发角色”权限。

Step3：权限管理

在上述“角色管理”中已对角色相关内容进行了介绍，虽然部分默认配置涉及数据权限管理（如背景信息），但DataWorks仍提供了更为专业的概述，帮助您快速构建平台的数据内容、个人隐私等相关的安全能力，实现更加精细化、场景化的数据权限及高危风险行为管控，满足企业面向高风险场景的各类安全要求（例如，审计），无需您额外配置即可直接使用该功能。

1. 准备工作。

   请参考如下步骤打开相关配置，以实现更加精细化的权限管控能力。

   进入DataWorks控制台，在**工作空间列表**页面中单击**创建工作空间**，选择工作空间的模式为**标准模式（开发跟生产隔离）**，其他参数根据界面提示配置，完成后单击**下一步**。

i. 启动列级别访问控制。

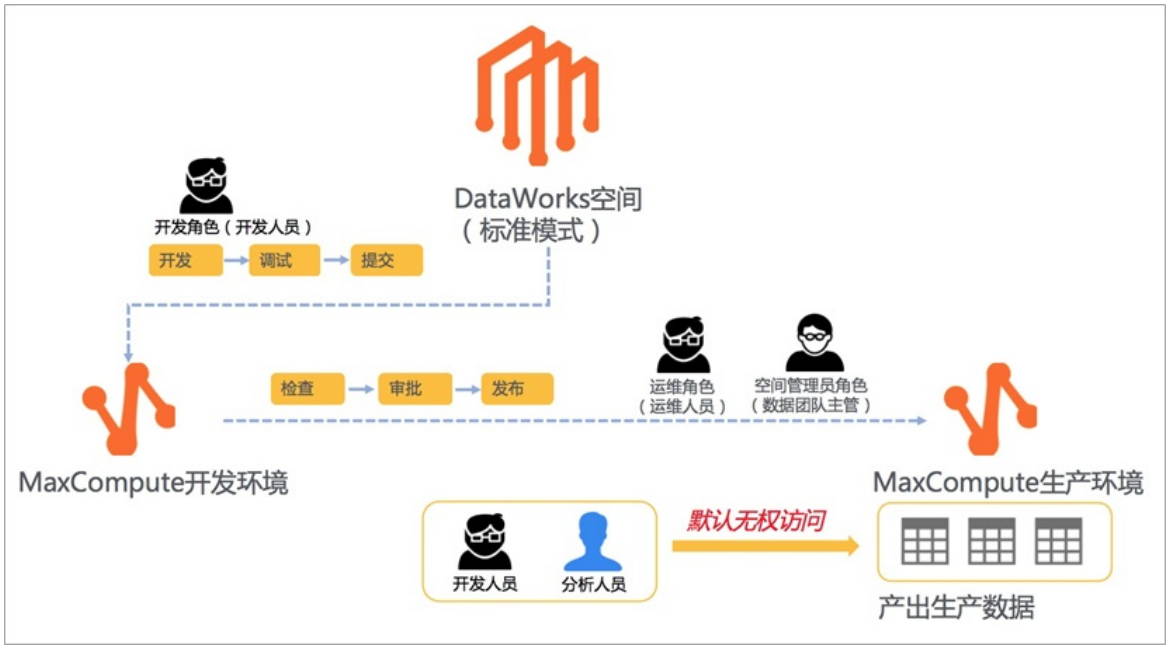
进入DataWorks控制台，在**工作空间列表**页面中单击**工作空间配置**后单击**更多设置**，在**MaxCompute高级配置**的**基本设置**中，选择MaxCompute项目，然后单击**启动列级别访问控制**后的开启按钮。

ii. 进入安全中心平台安全诊断，打开数据下载控制。

操作详情请参见进入平台安全诊断。打开后即可实现odpscmd tunnel download，数据下载权限需申请后才能执行。

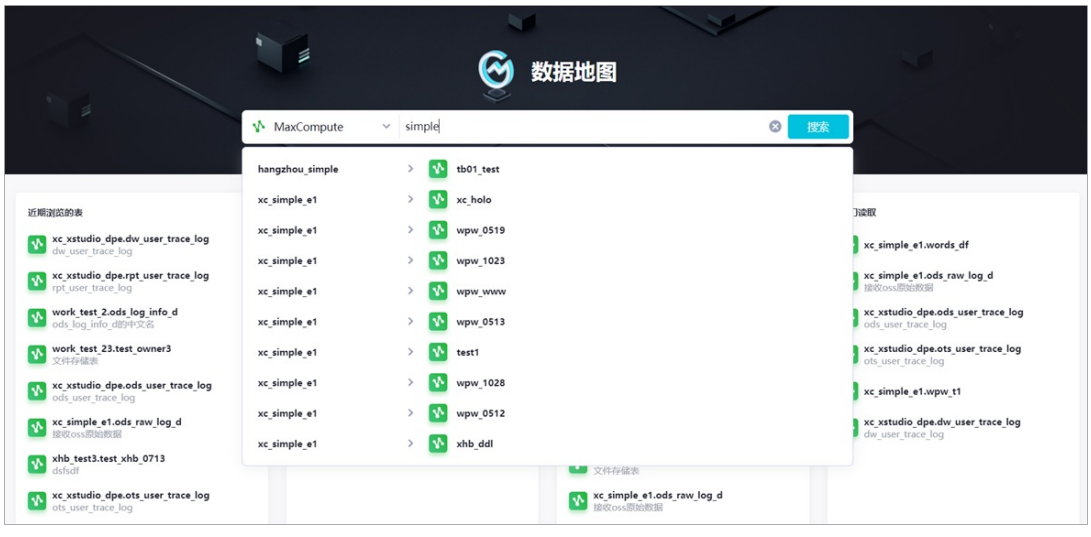
2. 标准模式下的数据权限管理。

在标准模式下，当数据产出至生产环境后，默认情况下任何人均无数据读写权限。



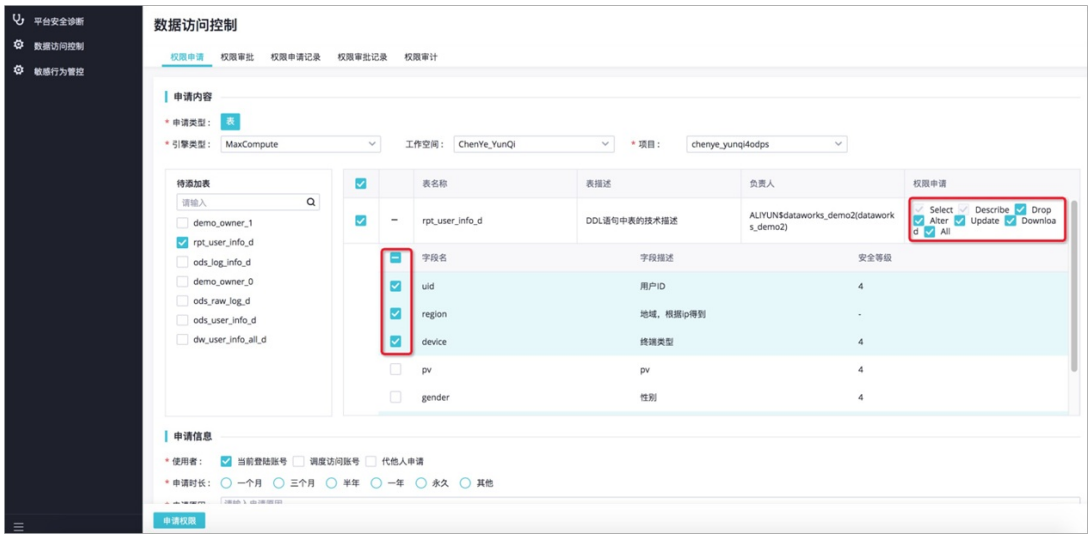
此时，如开发人员或分析师需读取生产环境数据进行数据分析或用于生产，则可以发起相关权限申请流程。

- 默认数据权限申请流程。
  a. 登录数据地图定位至某张表。



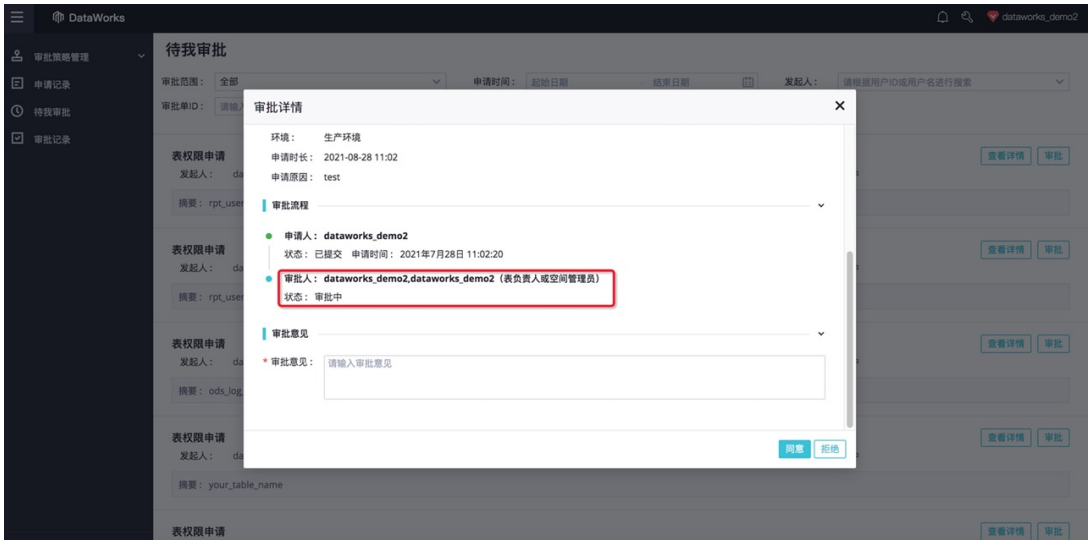
  b. 点击权限申请。

c. 选择所需申请的权限点以及字段并发起申请。



d. 执行审批。

表责任人（Owner）或空间管理员角色进入DataWorks审批中心-待我审批，即可完成表权限审批。



- 自定义数据权限申请流程。

  在上述步骤中，表权限申请流程仅需由“表责任人（Owner）”或“空间管理员”执行审批即可，但对于权限管控相对严格的企业来说，远远无法满足其需求，仍需更加复杂的审批流程来支持权限申请与审批。

  DataWorks支持管理员通过定义MaxCompute项目维度、数据保护伞分级分类维度的审批策略来进行数据权限申请审批，满足企业在不同场景下、针对不同类别数据的审批流程定义，实现更加安全的授权流程。

Step4：数据建模

数据建模的流程包括：创建数据标准、创建数据模型、修改数据模型、保存模型至模型库、提交模型至开发环境计算引擎。操作详情请参见概述。

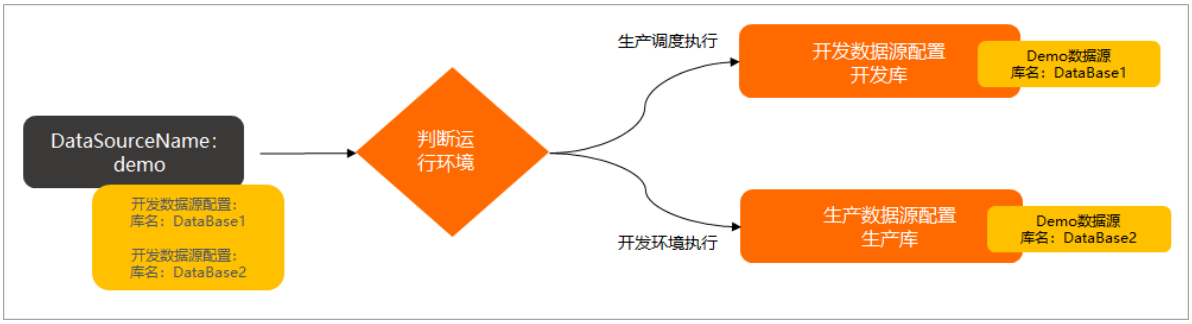
Step5：数据开发与生产

进行数据开发与生产前，您需要了解几个重要的概念。

- 生产开发数据源

  DataWorks支持基于标准模式工作空间对应的两个环境这一特性，来分别为这两个环境配置不同的数据库访问地址。即您可以在数据源配置界面分别为开发环境（DataStudio）测试运行时和生产调度时指定不同的数据库访问地址。

  同一个名称的数据源存在开发环境和生产环境两套配置，您可以通过数据源隔离使其在不同环境隔离使用。DataWorks将通过判断任务执行环境来自动访问对应环境下该同名数据源对应的配置信息。详情请参考数据源开发和生产环境隔离。



- 调度参数

  调度参数是DataWorks在调度场景下支持自动根据业务时间替换为具体值的参数，节点中使用调度参数后，在调度场景下，可以实现将对应业务时间的业务数据动态写入对应的时间分区中。详情请参见调度参数概述。

- 依赖关系

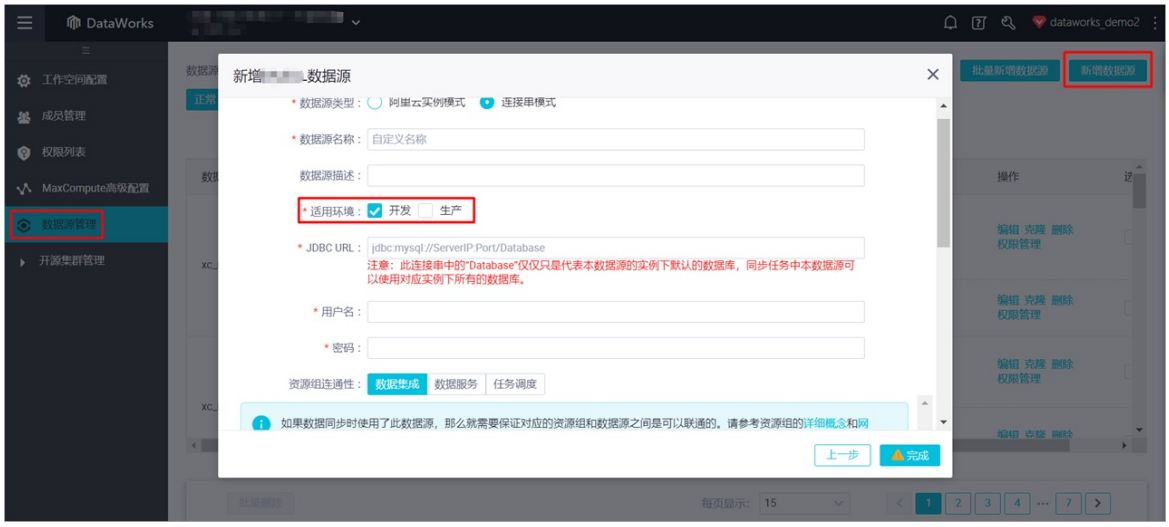
  调度依赖就是调度场景下节点间的上下游依赖关系，在DataWorks调度场景中，上游节点运行成功，下游任务节点才会开始运行。

  根据表血缘来配置节点调度依赖后，可以保障调度任务在运行时能取到正确的数据，避免下游节点取数据时，上游表数据还未正常产出，导致下游节点取数出现问题。

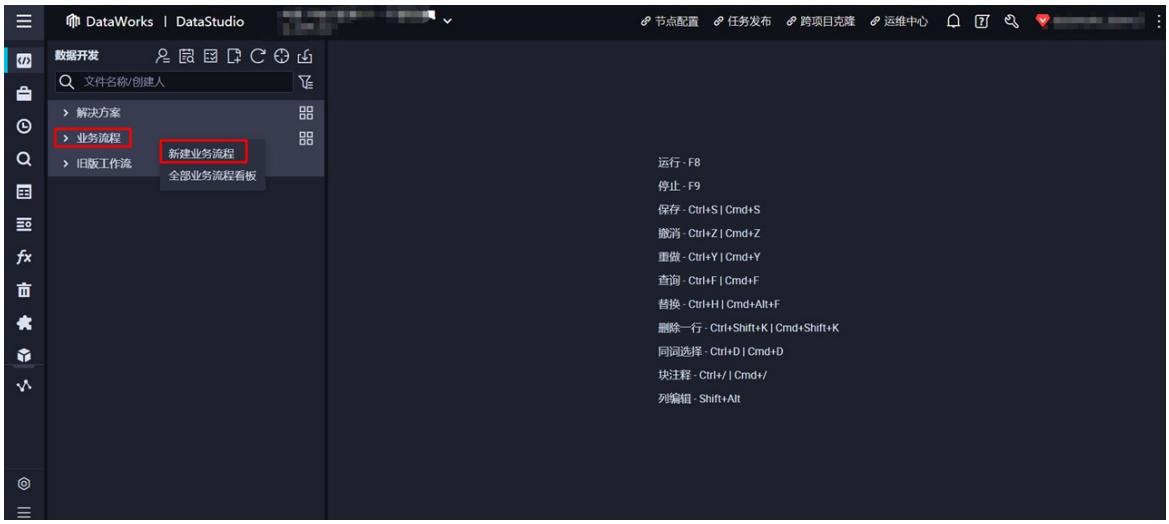
  在DataWorks依赖配置中，上游节点的输出作为下游节点的输入，形成节点依赖关系。平台支持通过自动解析快速设置节点依赖，关于调度依赖详情可参考文档同周期调度依赖逻辑说明。

了解相关概念后，您可进行数据开发与生产的操作步骤。

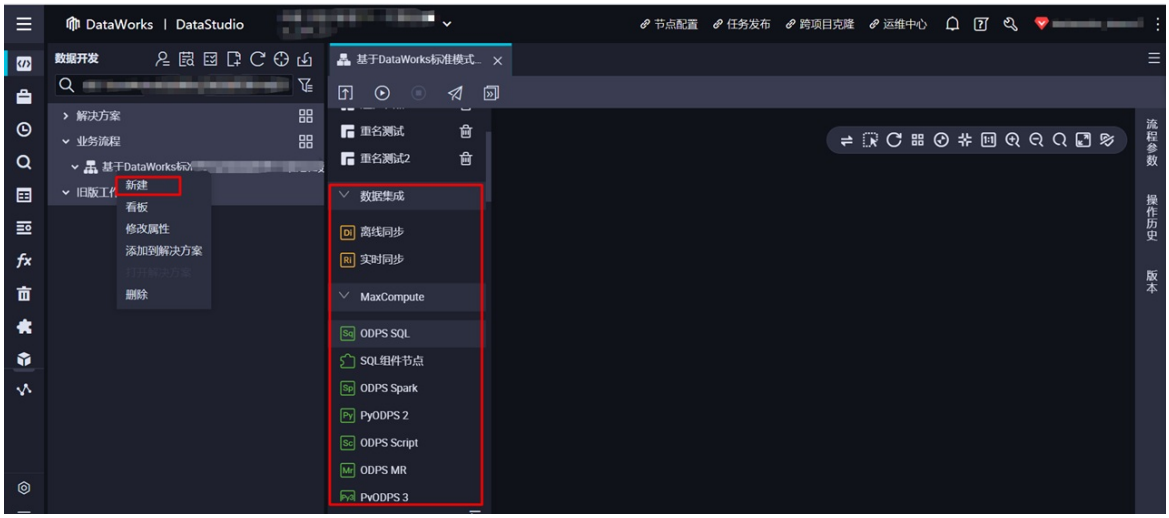
1. 以管理员权限创建生产开发数据源。



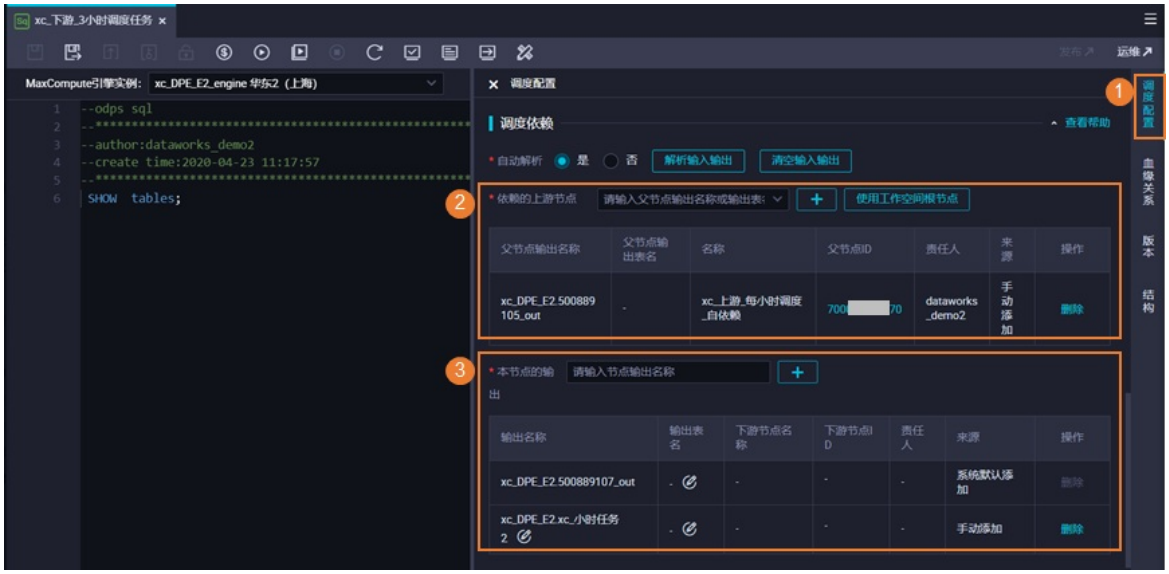
2. 开发人员创建业务流程。



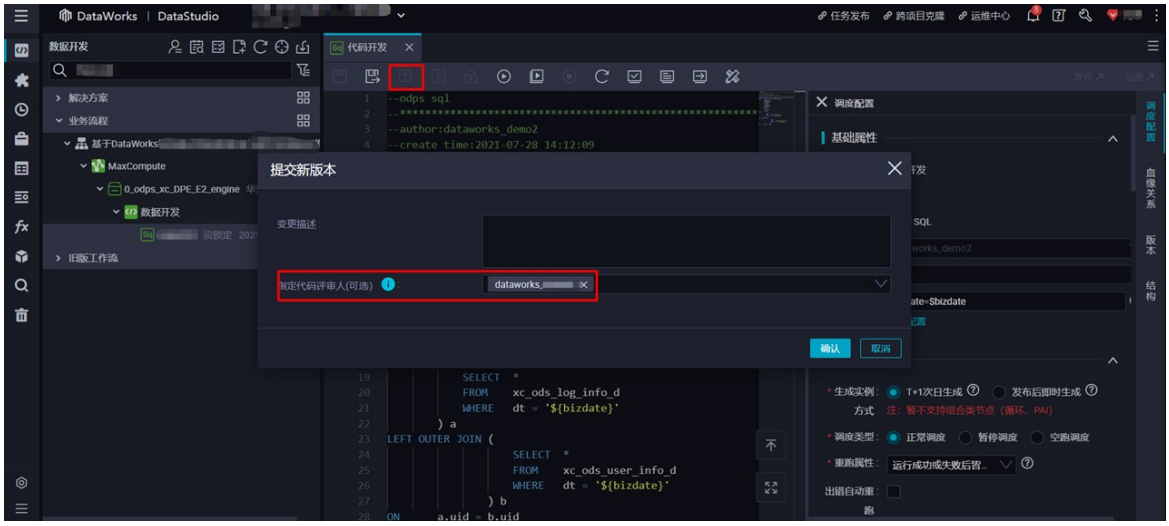
3. 开发人员在业务流程内新建节点。



4. 开发人员配置节点任务并配置调度属性、依赖关系。

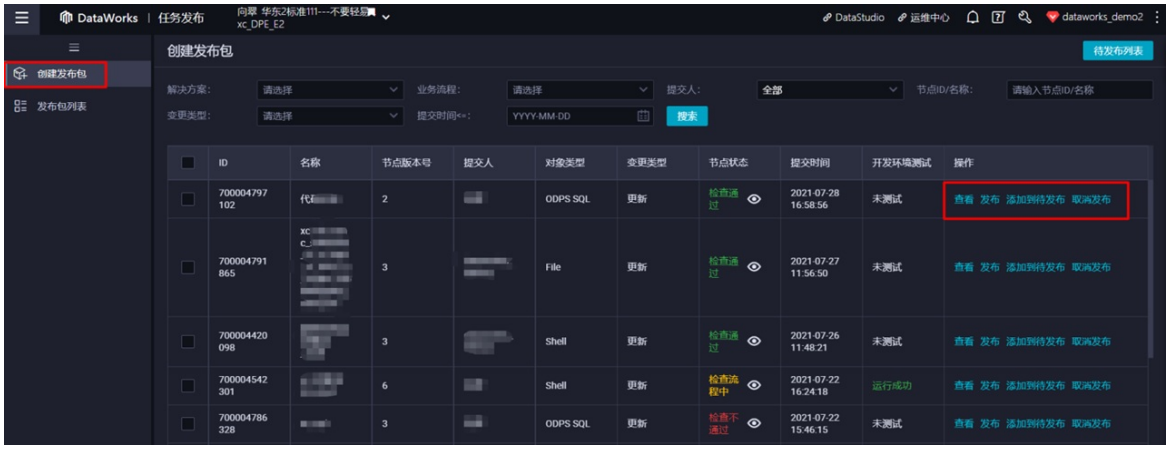


5. 开发人员提交任务并发起代码审核。



6. 代码审批人员进行代码审核决定是否通过审批进入待发布状态。

7. 有运维权限的同学可将节点发布生产或者根据节点变更内容来决定是否取消发布。



8. 运维同学或开发人员进入发布包界面查看发布状态。

4.数据分析

4.1. 电商网站智能推荐

电商网站智能推荐基于阿里巴巴的大数据和人工智能技术，结合在电商行业的多年积累，为开发者提供个性化推荐服务，提升商品的购买率和转化率。

概述

本实践以电商网站为例，通过日志服务采集日志，将RDS作为后端数据服务、MaxCompute作为数据仓库，并通过DataWorks进行数据同步和处理，使用智能推荐产品搭建电商网站智能推荐系统。

电商行业需要向用户推荐的物品包括物流信息、售卖信息等商品属性，可以引导用户直接交易，以提升购买率和转化率，适合首页推荐、信息流等相关场景。

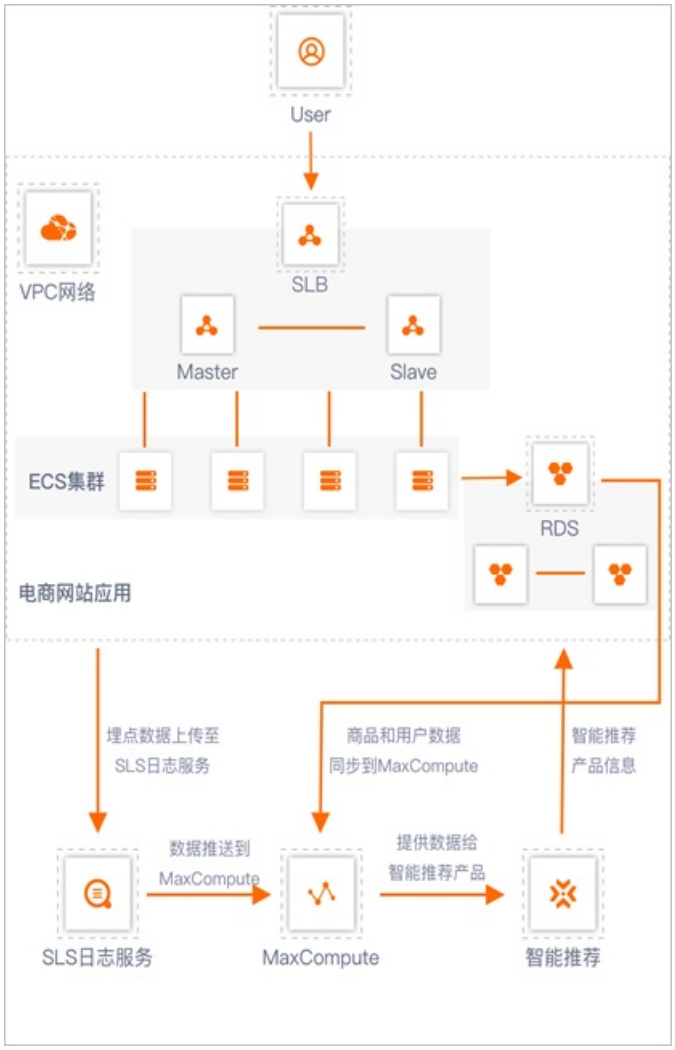
应用场景

- 以瀑布流的形式展示猜您喜欢、个性化内容。
- 在商品页面为您提供产品的相关推荐。
- 在网站首页为您提供焦点图推荐和热门推荐。
- 可以扩展至新闻资讯、视频直播和社交领域。

方案介绍

实现电商网站智能推荐方案的流程如下：

1. 日志服务对行为数据进行埋点。
2. 上传行为日志并推送至MaxCompute。
3. 上传商品和用户数据同步至MaxCompute。
4. 将数据推送至智能推荐，通过AI算法输出推荐结果。



方案优势

- 来自于阿里巴巴集团内部研发多行业和多场景的成熟方案，适用于不同用户的业务需求。
- 您无需考虑流处理、算法、运维、监控等问题，DataWorks平台一站式解决。
- 用户级数据隔离，敏感信息加密，保障信息安全。
- 算法技术、多模态融合、高效冷启动方案、实时调整策略和模型训练，无需人工干预。
- 多产品之间无缝对接，数据小时级别同步。

方案详情

方案的详情请参见电商网站智能推荐。

4.2. 互联网、电商行业离线大数据分析

通过阿里云MaxCompute、云数据库RDS MySQL、DataWorks等产品，可以实现互联网、电商网站的离线数据分析，且支持通过DataV大屏展示分析后的业务指标数据。

概述

电商网站的销售数据通过大数据进行分析后，可以在大屏幕展示销售指标、客户指标、销售排名和订单地区分布等业务指标数据。DataV大屏支持可视化动态展示销售数据，触控大屏支持您自助查询数据，极大地提高数据的可读性。

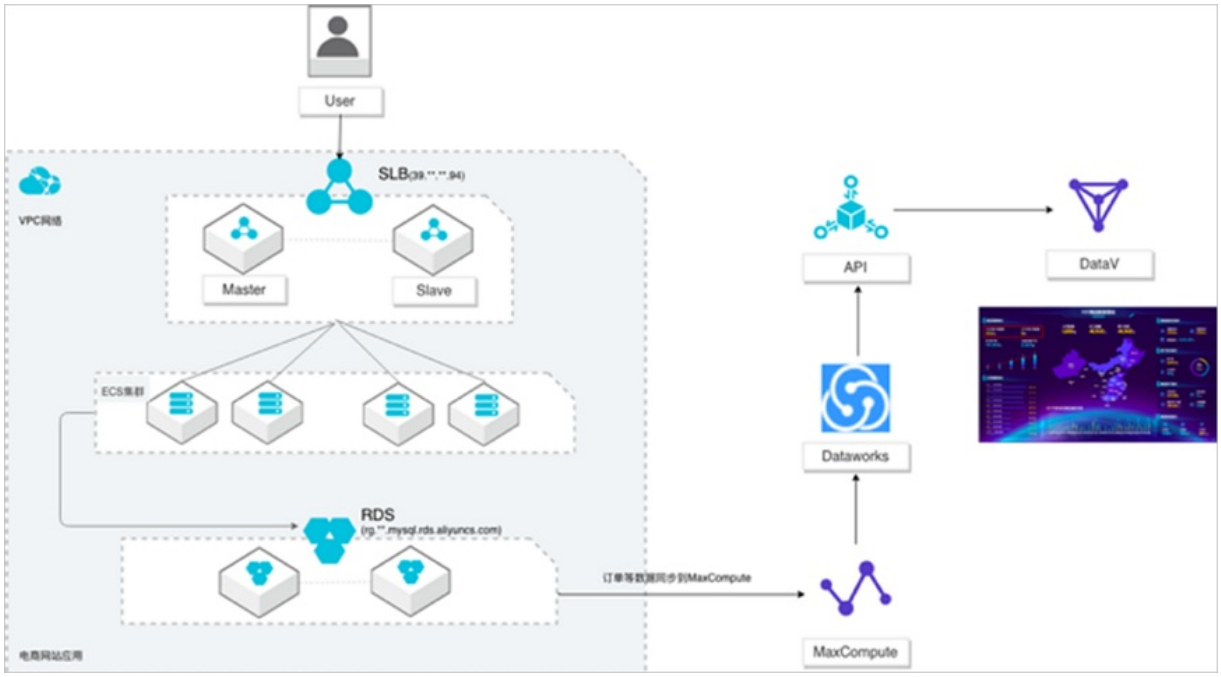
应用场景

- 电商网站数据看板。
- 全国、全球业务的态势分析。
- 互联网、金融行业的风险数据监控。

方案介绍

实现互联网、电商行业离线大数据分析的流程如下：

1. 同步用户订单等数据至MaxCompute。
2. 通过DataWorks对原始数据进行处理，并形成开放API。
3. 以API的形式通过DataV在大屏上展示结果数据。



方案优势

- 大规模存储：超大规模存储且自动扩容，最大可以支持EB级别的数据。
- 高性能：性能更加高效、稳定。
- 低成本：与自建数据库进行分析相比，成本更低。
- 安全：原生的多租户系统，以工作空间进行隔离，所有计算任务在安全沙箱中运行。
- 可视化编辑：在图形化的编辑页面，通过拖拽即可完成专业级的大数据可视化。

方案详情

方案的详情请参见互联网、电商行业离线大数据分析和大屏展示。

4.3. 基于MaxCompute进行大数据BI分析

您可以通过阿里云MaxCompute、云数据库RDS MySQL、DataWorks等产品进行数据分析，且可以通过Quick BI进行可视化展示。

概述

本实践以电商行业为例，通过MaxCompute、DataWorks对业务数据和日志数据进行ETL处理，并同步至分析型数据库MySQL（AnalyticDB for MySQL）进行实时分析后，再通过Quick BI进行可视化展示。

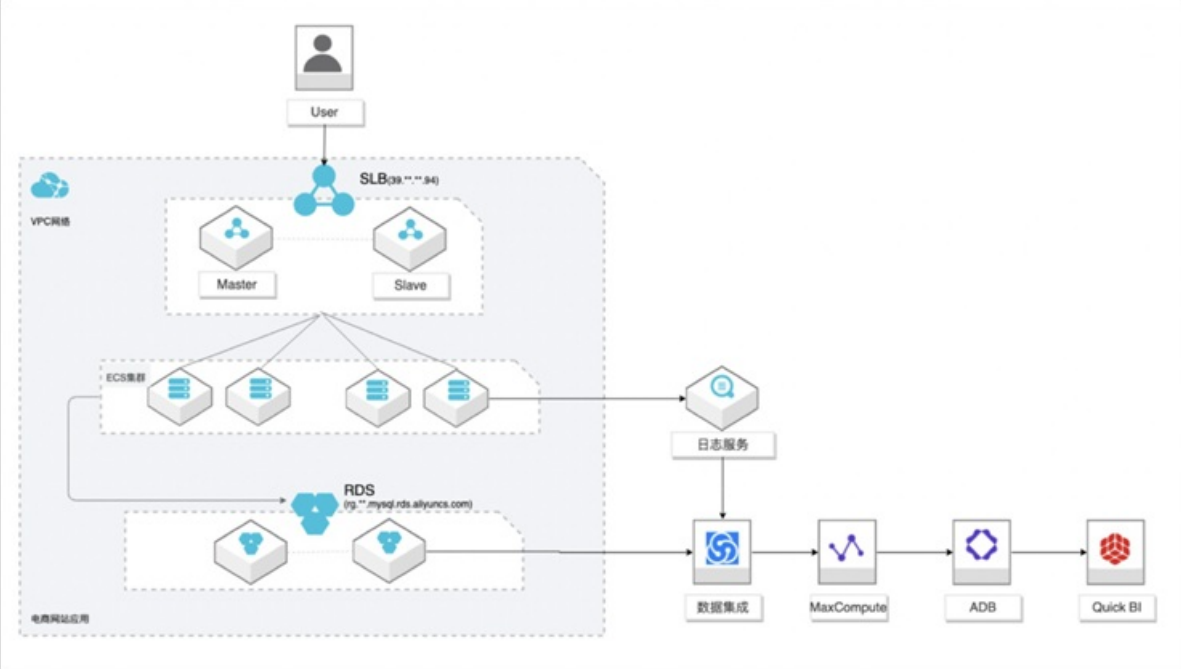
应用场景

- 互联网行业、电商、游戏行业等网站、App、小程序应用内的BI分析场景。
- 各类网站的BI分析场景。

方案介绍

基于MaxCompute进行大数据BI分析的流程如下：

1. 通过数据集成同步业务数据和日志数据至MaxCompute。
2. 通过MaxCompute、DataWorks对数据进行ETL处理。
3. 同步分析后的结果数据至AnalyticDB for MySQL。
4. 通过Quick BI可视化建立用户画像。



方案优势

- 以AnalyticDB for MySQL配合Quick BI快速、实时分析数据的核心能力为切入点，引导用户同步业务数据、日志数据至阿里云的日志服务和分析性数据库。
- 融合阿里云的日志服务的生态，增强用户体验。例如，无缝对接Blink、Elasticsearch、AnalyticDB for MySQL、E-MapReduce和DataV等产品。
- 通过MaxCompute、AnalyticDB for MySQL强大的数据加工和分析能力，降低大数据平台建设的门槛，轻松解决了海量数据的计算问题。同时有效降低企业成本，并保障数据安全。
- 与第三方开源生态无缝对接，在不侵入用户应用的情况下，传输日志至日志服务，降低使用门槛。

方案详情

方案的详情请参见基于MaxCompute的大数据BI分析。