ALIBABA CLOUD

阿里云

DataWorks
数据开发（DataStudio）

文档版本：20220712

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.数据开发概述

DataWorks的DataStudio（数据开发）模块为您提供了界面化、智能高效的大数据数据开发与测试体验，本文将基于开发组件（节点）、支持开发的任务类型、开发过程中的资源管控与使用说明、开发过程中的成员权限控制（资源与功能）来说明数据开发的功能使用。

> 说明
> - 本文以生产环境与开发环境隔离的标准模式工作空间为例，为您介绍数据开发的使用说明。
> - 所有生产环境调度节点的代码变更都需要在数据开发界面修改完成后走发布流程进行发布。
> - 如果您工作空间下无可用的引擎或目录树上对应引擎不可见，请在工作空间配置界面确认是否已经开通并绑定对应引擎服务。业务流程下仅展示当前工作空间下已经绑定的引擎服务，引擎绑定详情可参考文档：配置工作空间。
> - 如果您无法操作部分功能，或者没有新建入口，请在工作空间配置的成员管理处确认是否有开发权限（即操作账号是否为阿里云主账号、是否被授予开发角色或空间管理员），或查看当前开通的DataWorks版本是否为要求的版本。

开发组织结构

您可以基于包括**工作空间 > 解决方案 > 业务流程**三级结构，对业务进行划分，您可以基于公司部门、公司业务或数仓层次进行规划分组。

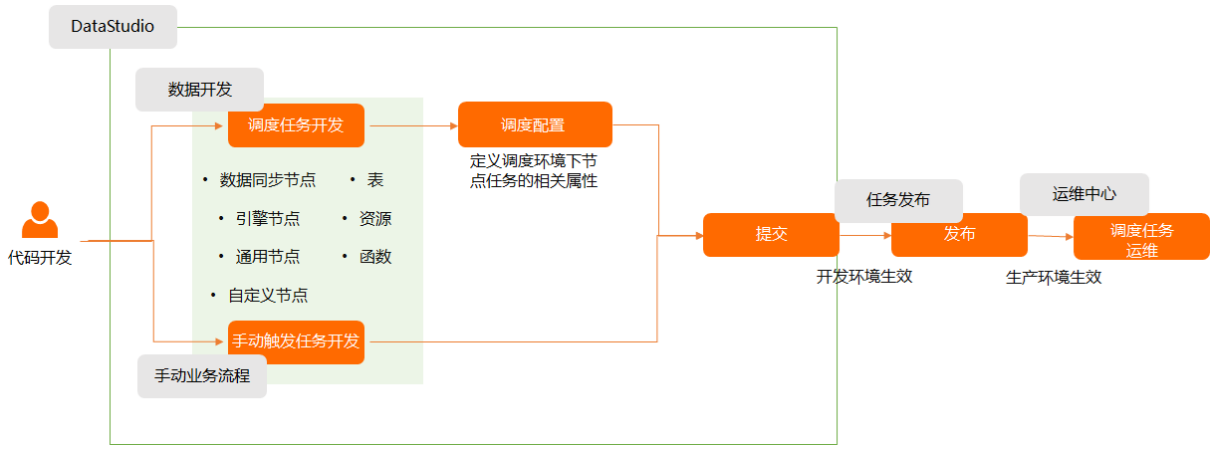
结构层级	特征	定位
工作空间	不同的工作空间可以有不同的管理员、不同的内部成员，各工作空间拥有完全独立的成员角色设定以及引擎实例的各项参数开关。关于工作空间的规划请参见规划工作空间。	DataWorks支持的最大业务划分粒度，权限组织的基本单位，用来控制您的开发、运维等权限。工作空间内成员的所有代码均可以协同开发管理。
解决方案	您可以将一类业务流程划分为一个解决方案进行统筹管理，同时一个业务流程也可以被多个解决方案复用，您只需要开发自己的解决方案。其他人可以在其它解决方案或业务流程中，直接编辑您引用的业务流程，构成协同开发。	业务整合。
业务流程	业务的抽象实体，让您能够以业务的视角来组织数据代码开发。工作空间之间的业务流程、任务节点独立开发，互不影响。 业务流程两种形态，目录树与面板，让您从业务视角组织代码，资源类别更明确，业务逻辑更清晰。 • 目录树结构提供基于任务类型的代码组织方式。 • 业务流程面板提供流程化的业务逻辑展现方式。	具体的代码开发、资源组织单位。

数据开发基于业务流程下对应的节点进行开发操作，您可以在业务流程面板下新建一个或多个业务流程，每个业务流程按照引擎类型进行分组，每个引擎分组下再对数据开发类型节点、表、资源、函数进行一步分组，即一类业务使用的组件（节点、表、资源、函数）统筹在一个业务流程中，业务流程下仅展示当前业务流程中使用的组件。

- 在DataWorks上，具体的数据开发工作是基于业务流程开展的，所以您需要先新建业务流程，再进行后续的开发工作。
- 所有生产环境调度节点的代码变更都需要在数据开发界面修改完成后走发布流程进行发布。

简单逻辑说明

DataWorks的数据开发基于业务流程进行数据开发，支持手动触发任务与调度任务进行开发，您可以选择引擎节点、控制类节点、自定义节点进行数据清洗操作。调度任务需要配置调度相关参数，并提交节点进入待发布界面，在任务发布界面进行发布操作。节点发布完成后，任务将进入生产环境，之后将根据您的调度配置周期性调度。



DataStudio主要功能

- 开发节点类型

  全面的引擎能力封装，让您无需接触复杂的引擎命令行。并提供自定义节点插件化机制，支持您扩展计算任务类型，自主接入自定义计算服务，同时，您可以结合DataWorks其他节点进行复杂数据处理。

- 基于数据集成节点进行数据同步。
- 基于引擎节点数据开发。
- 引擎节点结合通用节点进行复杂流程处理。
- 通过节点配置自定义节点进行数据开发。

开发节点类型的选择请参见下文的选择数据开发节点章节。

- 任务开发

  支持调度任务（周期任务）开发与手动触发式任务开发，且开发过程中通过界面化、智能化提高开发效率：

  - 业务流程混合编排：可视化拖拽式多引擎任务混合编排，详情请参考下文的选择开发任务类型章节的**新建调度任务**。
  - 智能SQL编辑器：AI加持的SQL编辑器，智能提示，SQL算子结构可视化展示，详情请参考下文的选择开发任务类型章节的**节点代码查看与版本管理**。

  新建周期任务、新建手动任务的介绍详情请参见选择开发任务类型。

- 表、资源、函数的可视化管理与使用

  表、资源、函数的管理与使用的介绍请参见下文的表、资源、函数的管理与使用。

- 成员权限管控与开发行为管控

  DataWorks数据开发的权限管控主要包括：

  - 资源权限控制
  - 界面功能权限控制
  - 操作流程管控

  详细介绍请参见下文的成员权限管控与开发行为管控。

- 代码版本管理与操作审计

  操作审计主要包括：

  - 获取开发人员界面相关操作审计日志。
  - 重要数据通过事前设置来获得事后溯源能力。
  - MaxCompute表权限审计。
  - 表数据、节点删除的恢复。
  - 节点的版本对比与回滚。

  详细介绍请参见下文的操作审计。

选择数据开发节点

DataWorks将引擎能力进行封装，您可以基于引擎节点进行数据开发，无需接触复杂的引擎命令行，同时您也可以结合平台提供的通用类型节点进行复杂逻辑处理，此外，DataWorks也为您提供自定义节点插件化机制，支持您扩展计算任务类型，自主接入自定义计算服务，通过自定义节点开发，来实现自定义处理代码逻辑。

> **说明** 产品能力在仍在不断丰富中。

- 数据集成节点：您可基于数据集成节点进行数据同步

数据集成节点	使用介绍
离线节点	用于离线（批量）数据同步场景。 ○ 支持复杂场景下多种异构数据源，基于数据传输框架，通过抽象化的数据抽取插件（Reader）、数据写入插件（Writer）间的离线（批量）数据同步。 ○ 支持40+关系型数据库、非结构化存储、大数据存储、消息队列之间的数据同步。 离线同步支持的数据源详情请参见支持的数据源与读写插件。
实时同步节点	用于实时同步同步场景。 实时同步包括实时读取、转换和写入三种基础插件，各插件之间通过内部定义的中间数据格式进行交互。 实时同步支持的数据源详情请参见实时同步支持的数据源。
数据同步解决方案	DataWorks为您提供多种数据源之间进行不同数据同步场景的同步解决方案，包括实时数据同步、离线全量同步、离线增量同步等同步场景，助力企业数据更高效、更便捷的一键上云。 同步任务配置化方案具有如下优势： ○ 全量数据初始化。 ○ 增量数据实时写入。 ○ 增量数据和全量数据定时自动合并写入新的全量表分区。 同步解决方案详情可参考文档概述。

- 引擎节点：您可基于引擎节点进行数据开发

  在具体业务流程下，您可以选择在某一引擎下的数据开发分组下新建对应引擎类型节点，来将相应的引擎代码下发到对应的数据清洗引擎上执行。

DataWorks集成的引擎	DataWorks对引擎能力的封装
MaxCompute	○ 创建ODPS SQL节点 ○ 创建ODPS Spark节点 ○ 创建PyODPS 2节点 ○ 创建PyODPS 3节点 ○ 创建ODPS Script节点 ○ 创建ODPS MR节点
E-MapReduce	○ 创建EMR Presto节点 ○ 创建EMR Hive节点 ○ 创建并使用EMR MR节点 ○ 创建EMR Spark SQL节点 ○ 创建并使用EMR Spark节点 ○ 创建并使用EMR Shell节点 ○ 创建EMR Impala节点 ○ 创建并使用EMR Spark Streaming节点

DataWorks集成的引擎	DataWorks对引擎能力的封装
AnalyticDB For PostgreSQL	AnalyticDB for PostgreSQL节点
AnalyticDB For MySQL	AnalyticDB for MySQL节点
Hologres	Hologres SQL节点
数据库	MySQL节点
ClickHouse	ClickHouse SQL节点
算法	机器学习（PAI）节点

- 通用节点：引擎节点可结合通用节点进行复杂逻辑处理

  在具体业务流程下，您可以在通用节点分组下新建对应的节点，结合引擎节点实现复杂逻辑处理。

业务场景	节点类型	使用说明
业务管理	虚拟节点	虚拟节点属于控制类型节点，它是不产生任何数据的空跑节点，通常作为业务流程统筹节点的根节点，方便您管理节点及业务流程。
事件触发	HTTP触发器节点	如果您希望其他调度系统的任务完成后触发DataWorks上的任务运行，您可以使用此功能。
	OSS对象检查节点	通过监控OSS对象产生来触发下游节点执行。
	FTP Check节点	通过监控FTP文件产生来触发下游节点执行。
参数赋值	赋值节点	用于参数传递，通过自带的output输出将赋值节点最后一条查询或输出结果通过节点上下文功能传递到下游，实现参数跨节点传递。
控制类	配置for-each节点	用于遍历赋值节点传递的结果集。
	配置do-while节点	用于循环执行部分节点逻辑，同时您也可以结合赋值节点来循环输出赋值节点传递的结果。
	分支节点	用于对上游结果进行判断决定不同结果走不同的分支逻辑，您可以结合赋值节点一块使用。
	归并节点	用于对上游节点的运行状态进行归并，用于解决分支节点下游节点的依赖挂载和运行触发问题。
参数传递	参数节点	用于上游节点间参数汇总与分发向下传递。
	Shell节点	Shell节点支持标准Shell语法，但不支持交互性语法。

业务场景	节点类型	使用说明
代码复用	创建SQL组件节点	SQL组件是一种带有多个输入参数和输出参数的SQL代码模板。使用SQL代码处理数据表时，通过过滤、连接和聚合源数据表，获取结果表。 **说明** 目前仅支持MaxCompute语法。

- 自定义节点：通过节点配置自定义节点进行数据开发

  DataWorks提供自定义节点插件化机制，支持您扩展计算任务类型，自主接入自定义计算服务。您可以本地开发好插件代码，通过节点配置界面将该插件添加至DataWorks环境内，添加完成后，当前在数据开发时，可以在具体业务流程的自定义分组中选择该自定义节点进行数据开发。

  自定义节点的使用流程如下。

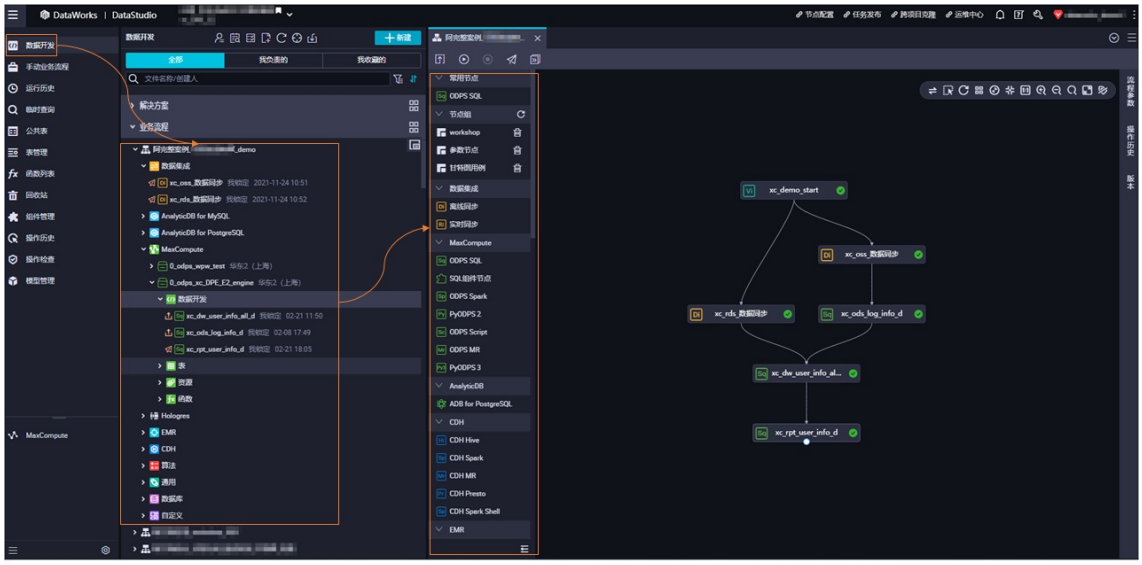
操作流程	步骤描述
step1：开发自定义插件包	DataWorks自定义节点中运行任务时，需要调用自定义插件，因此在使用自定义节点前您需要创建好自定义插件包，并上传发布至DataWorks，便于使用自定义节点运行任务时使用。
step2：新增节点插件	在DataWorks环境中部署该插件。
step3：新增自定义节点	新增自定义节点，配置自定义节点与自定义插件关系，编辑自定义节点在DataWorks界面的交互方式，基本信息、编辑器。

选择开发任务类型

DataWorks上支持周期调度任务的开发，同时也支持手动触发式任务的开发。您可以在左侧的目录树上右键来新建任务，也可以在双击业务流程后通过鼠标拖拽来新建任务。

- **新建调度任务**
  i. 新建周期调度节点

在数据开发分组下新建节点时，您可以在业务流程DAG图通过可视化拖拽组件的方式编排业务流程，即通过拖拽的方式设置业务流程内的节点依赖关系。此外，跨业务流程、跨工作空间的节点依赖关系可以通过自动解析功能来快速设置。



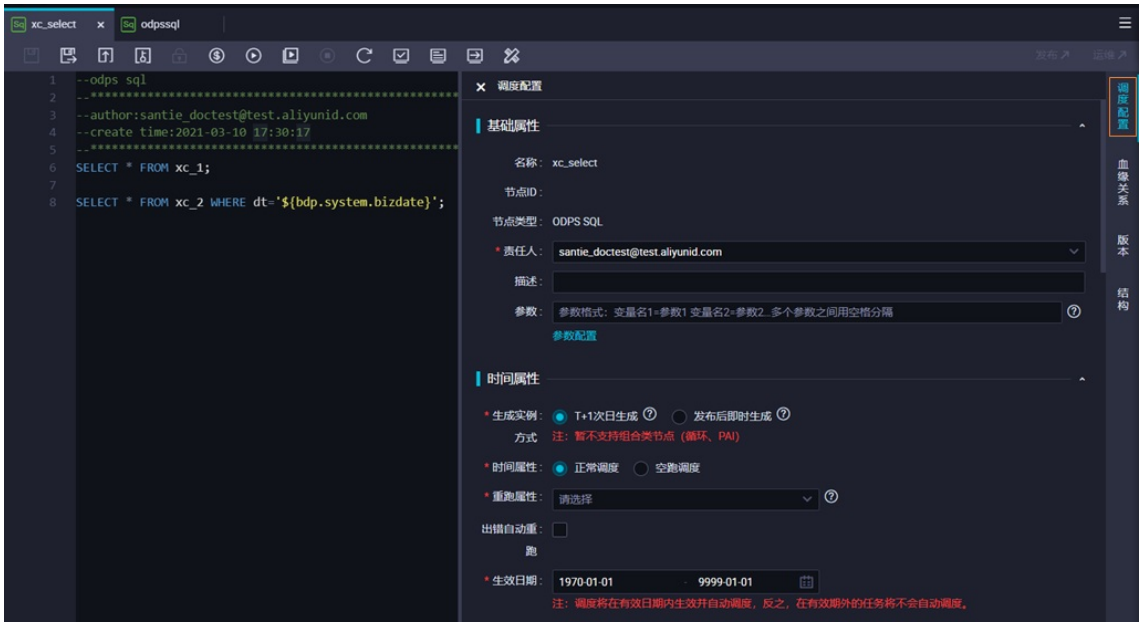
双击业务流程名称，即可进入业务流程的编辑面板。在面板中，您可以通过拖拽方式创建所需节点。

此外，您还可以通过以下功能，提高数据开发效率。

附加功能	使用说明
节点组	用于快速复制业务流程，您可以将多个节点组合成节点组，并在其他业务流程中快速引用该节点组。
流程参数	当整个业务流程需要对同一个变量进行统一的赋值或替换参数值时，请选择使用流程参数功能。
操作历史	用于查看业务流程面板的操作记录。
版本	每提交一次业务流程都将生成一个业务流程版本，您可以在此界面查看业务流程的每次提交的版本，支持版本对比。
代码搜索	用于通过关键字搜索节点中的代码片段，并展示包含该代码片段的所有节点及片段的详细内容。当目标表数据产生变更，您需要查找操作源（即导致目标表数据变更的任务）时，可以使用该功能。

ii. 节点调度配置

定义节点在调度运行时候的相关属性，DataWorks提供小时、分钟、日、月、年等多种形式调度，与日千万级大规模周期性任务调度服务，支持节点间参数传递，详情请参见调度配置。



其中调度配置的核心配置项如下。

> **说明** 支持通过批量操作入口批量修改相关属性，详情参考：批量操作

- 配置基础属性

核心配置参数	配置要点
参数	用于给代码中的变量赋值。您可以在参数处填写DataWorks调度参数实现变量动态赋值，调度参数会根据任务调度时间替换为具体的值，详情请参见调度参数概述。

- 时间属性配置说明

核心配置参数	配置要点
实例生成方式	节点配置完成后，什么时候会自动调度，即什么时候在周期实例面板生成实例自动调度。 ■ DataWorks支持T+1次日生成：任务发布生产环境后第二天生成周期实例并且自动调度。 ■ 发布后即时生成：发布后及时生成有时间限制，详情请参见实例生成方式：发布后即时生成实例。

核心配置参数	配置要点
调度类型	设置调度场景下节点是否真实执行，及非真实执行场景下对下游节点的影响控制。 ■ 正常调度 ■ 影响说明：正常执行任务（真实跑数据），当前节点正常执行后，也会触发下游节点正常调度执行。 ■ 使用场景：通常任务默认选中该项。 ■ 暂停调度 ■ 影响说明：冻结状态的周期任务，并且生成的实例也是正常状态的，不可执行任务，并且阻塞下游节点执行。 ■ 使用场景：当某一类业务流程在一定时间内不需要执行时，可选择此功能冻结业务流程根节点。 ■ 空跑调度 ■ 影响说明：任务是空跑状态（不会真实跑数据），即一调度到该任务便直接返回成功（执行时长为0），不会真正执行任务（执行日志为空），不会阻塞依赖了当前节点的下游节点执行（下游节点正常执行），且不会占用资源。 ■ 使用场景：当某一个节点在一定时间内不需要执行，并且需要不阻塞他的下游节点执行时，可选择此功能对当前节点设置空跑。
重跑属性	从数据幂等性考虑任务是否可以进行重跑。 ■ 运行成功或失败后均可重跑。 ■ 运行成功后不可重跑，运行失败后可以重跑。 ■ 运行成功或失败后皆不可重跑。
出错自动重跑	设置调度场景下节点的出错重跑次数与重跑间隔。
生效日期	当前节点在指定时间段内自动重跑，指定时间段外不自动调度（不生成周期实例）。
调度周期与定时时间	支持**分钟**、**小时**、**日**、**周**、**月**和**年**调度。 **说明** 非调度时间内实例空跑。
超时时间	任务运行时长超过指定时间，任务将自动终止运行，失败退出。

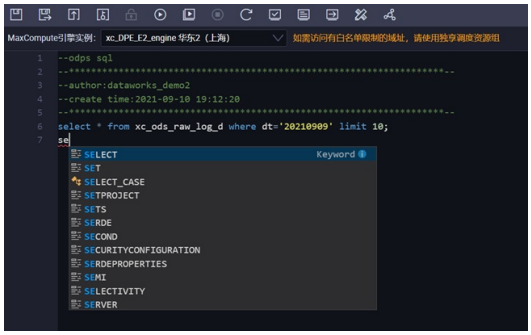
- 配置资源属性：用于指定任务调度时使用的调度资源组。

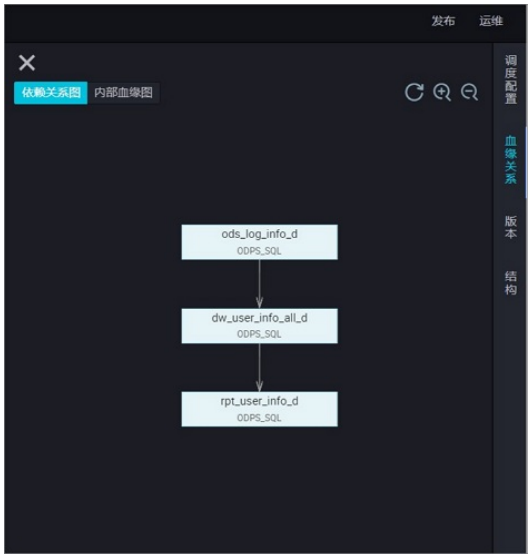
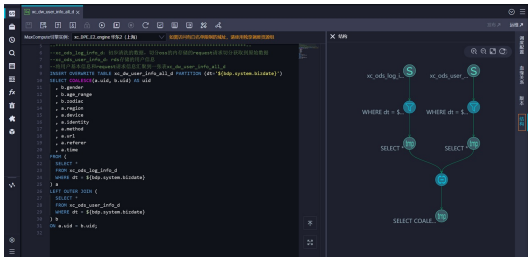
- 配置同周期调度依赖

核心配置参数	配置要点
依赖同周期	当前节点运行需要由哪些节点触发，此处依赖的是指定节点同一周期的依赖，即依赖某些节点今天的自动调度实例，从业务维度说便是当前节点依赖上游节点今天产出的表数据。
依赖跨周期	当前节点运行需要由哪些节点触发，此处依赖的是指定节点上一周期实例依赖。（即依赖某些节点昨天的自动调度实例，从业务维度说便是当前节点依赖上游节点昨天产出的表数据。

- 配置节点上下文：与赋值节点同步使用，通过节点上下文可将赋值节点输出的结果集传递到下游节点，实现参数传递。

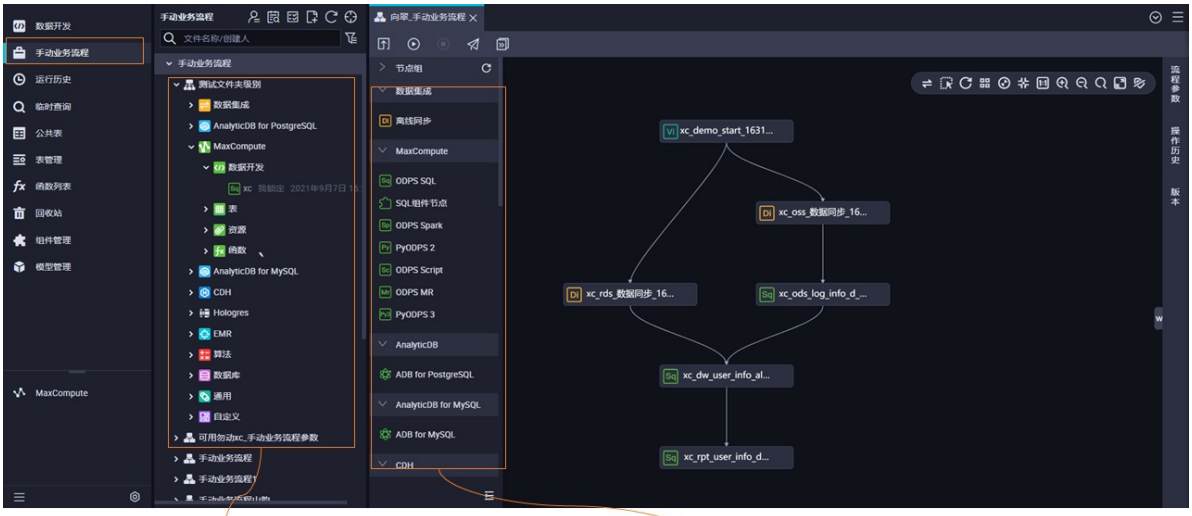
iii. 节点代码查看与版本管理

功能	功能描述	示意图
代码编辑	AI加持的SQL编辑器，提供智能语法提示。	

功能	功能描述	示意图
血缘关系	支持查看节点依赖关系图与节点内部代码的血缘关系图。	
版本	节点每次提交都将生成新的版本，您可以基于版本进行版本代码对比与版本回退操作。	
结构	代码结构通过SQL算子可视化展示。	

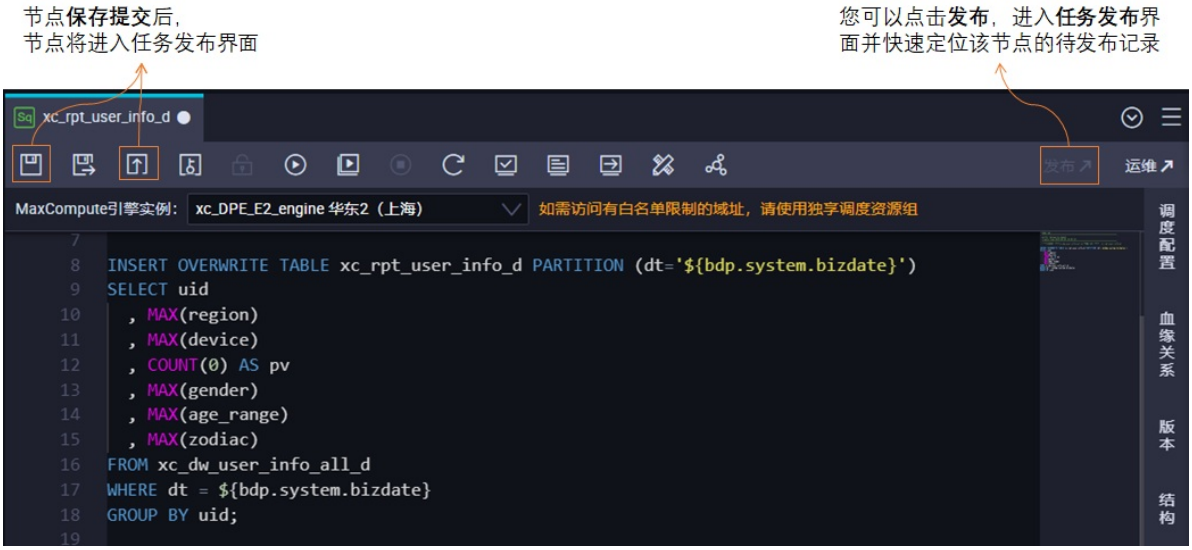
- **新建手动任务**

在手动业务流程模块具体的手动业务流程，数据开发分组下新建节点，您可以通过业务流程DAG图通过可视化拖拽组件的方式编排业务流程并拉线设置节点依赖关系。



- **提交节点**

  节点点击提交后，该条节点的操作记录将进入到任务发布界面，您可以在任务发布界面管控是否发布该条记录，只有将操作发布后，生产调度才会生效。



- **发布节点**

创建发布包界面展示工作空间下所有待发布的操作记录，包括新增、更新、下线记录，将对应操作发布生产，生产环境调度任务才会生效。

> **说明** 简单模式工作空间可以通过跨项目克隆将工作空间下的代码发布至另一个工作空间。

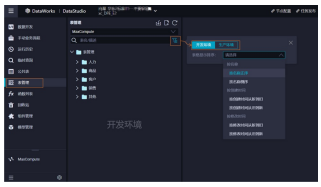
表、资源、函数的管理与使用

DataWorks将引擎下的表、资源、函数进行封装，您可以通过可视化方式创建表、资源，及注册函数等操作。

- **表管理**

  支持可视化操作表、上传本地表数据与表数据导出。

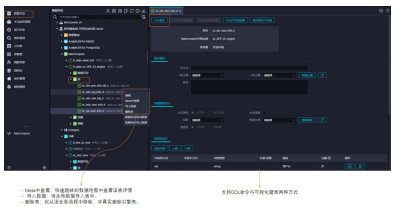
  DataWorks的表管理与使用的操作界面入口有多种，您可根据操作的需要进入对应的操作界面入口进行操作。

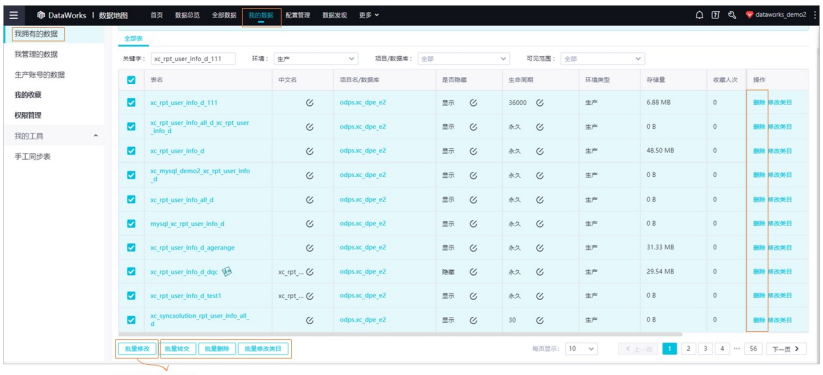
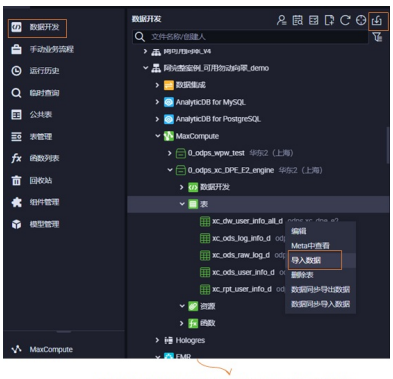
入口分类	主要操作入口1：数据开发页面	主要入口2：表管理页面	入口3：公共表页面
入口示意			

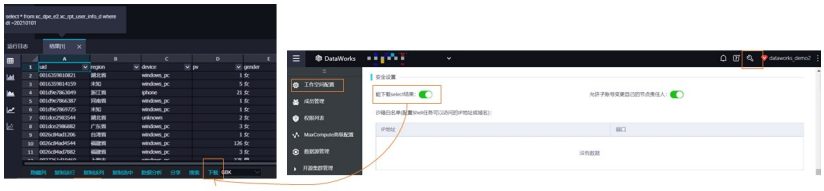
入口分类	主要操作入口1：数据开发页面	主要入口2：表管理页面	入口3：公共表页面
操作说明	在数据开发页面基于业务流程进行表管理（基于业务进行表管理）。 ○ 导入表至业务流程中：通过导入表功能将业务流程内使用到的表加入业务流程方便快速基于业务进行表管理。 ○ 从业务流程中移除表：如果业务流程内已用不到某张表，您可以右键表，进行删除表操作。 **说明** 此处的删除操作只是将该表从当前业务流程中移除，并非将表从引擎中真实的删除。	○ 管理表：可查看工作空间下所有的开发表及生产表。 ○ 表管理：可对表主题进行设置。	无

说明 您可以通过数据地图模块查看表的基本元数据信息、血缘信息和影响等，详情可参考数据地图文档：数据地图概述。

不同表操作在不同入口的操作注意事项如下。

操作分类	细分操作	主要操作入口1：数据开发页面	主要入口2：表管理页面
表管理	基本表操作： ○ 新建表 ○ 删除开发表 ○ 重命名表 ○ 修改表注释 ○ 添加字段 ○ ……	表操作行为基本与引擎行为一致。 	表操作行为基本与引擎行为一致。 

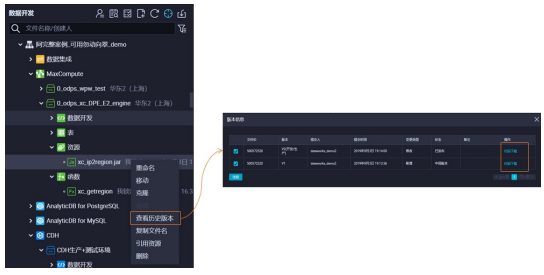
<table>
<tr><th>操作分类</th><th>细分操作</th><th>主要操作入口1：数据开发页面</th><th>主要入口2：表管理页面</th></tr>
<tr><td></td><td>○ 删除生产表
○ 批量转交表责任人
○ 批量修改表生命周期
○ 批量修改表中文名</td><td colspan="2">生产表不支持在数据开发、表管理页面中直接操作，需要在数据地图中操作，详情请参见数据地图概述。
 
- 批量修改生命周期
- 批量修改表搜索隐藏设置</td></tr>
<tr><td rowspan="3">表数据导入</td><td>上传本地数据至表中</td><td> 
• 基于数据开发业务流程表容器下具体的表上传表数据。
• 提供快速入口，通过搜索表找到指定表上传表数据。</td><td> 
• 基于表管理上传表数据。
• 提供快速入口，通过搜索表找到指定表上传表数据。</td></tr>
<tr><td>其他数据库数据同步至表中</td><td> 
通过离线同步节点、实时同步节点将其他数据库数据同步至表中。</td><td>不支持</td></tr>
<tr><td></td><td></td><td></td></tr>
</table>

<table>
<tr><th>操作分类</th><th>细分操作</th><th>主要操作入口1：数据开发页面</th><th>主要入口2：表管理页面</th></tr>
<tr><td rowspan="2">表数据导出</td><td>数据导出至本地</td><td colspan="2"> 

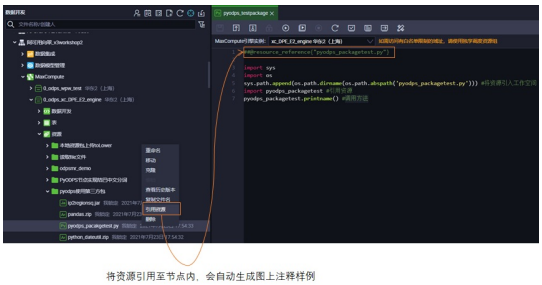
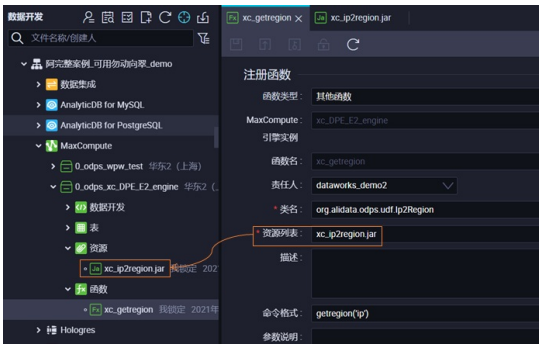
工作空间此项配置开启后，查询结果页才会有下载入口。
说明 管理员权限可控制此开关。</td></tr>
<tr><td>数据导出至其他数据库</td><td> 

通过离线同步节点、实时同步节点将数据同步至其他数据库。</td><td>不支持</td></tr>
</table>

- **资源的管理与使用**
  - 资源管理

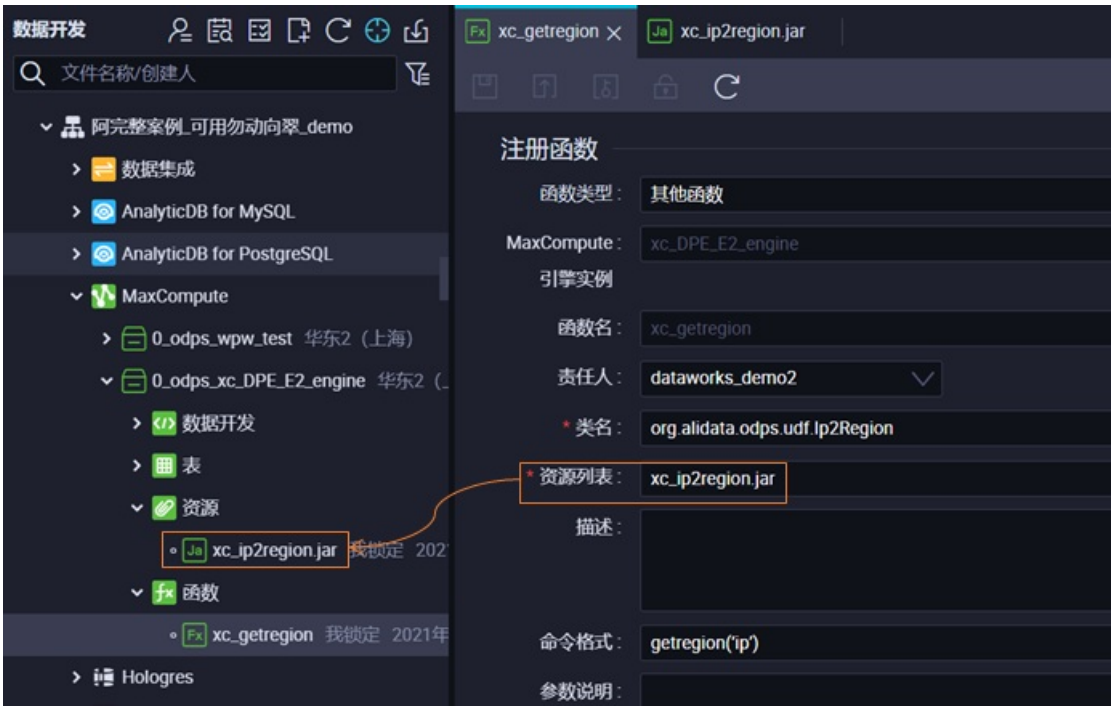
<table>
<tr><th>可视化上传</th><th>资源版本管理</th></tr>
<tr><td> 
</td><td> 
</td></tr>
</table>

- 资源使用

节点内使用	使用资源注册函数
 将资源引用至节点内，会自动生成图上注释样例	

- **函数的管理与使用**

  您可以在数据开发页面注册函数。



成员权限管控与开发行为管控

以下将DataWorks数据开发的权限分为两部分：引擎操作权限、DataWorks功能权限。

- 引擎权限管控：

  指将引擎绑定至DataWorks的某一工作空间后，您在DataWorks上进行数据开发过程中，对引擎进行操作所需权限的管控。

> **注意** 此权限要求为引擎的操作权限要求，对引擎的操作权限的控制有两部分：
> - 在绑定引擎至DataWorks时，会给预设角色部分引擎的操作（表、函数、资源）权限。
> - 没有预设角色或预设角色不包含的引擎操作权限，需进入引擎的授权页面进行授权，当前不支持直接在DataWorks的界面授权这类权限。

- DataWorks界面权限管控：

  指非引擎操作的其他DataWorks数据开发时，对DataWorks的界面功能的权限控制。

开发行为管控：指DataWorks提供的操作权限控制能力，您可以在敏感行为发生时做到第一时间阻断，支持人工干预或自定义事件检查逻辑，流程管控可分为阻断操作流程和不阻断操作流程仅通知。

- **引擎权限管控：MaxCompute**

  当前工作空间使用的是MaxCompute引擎时，DataWorks标准模式下：

  - DataWorks预设角色与MaxCompute引擎**开发项目**的Role存在权限映射关系，预设角色默认拥有MaxCompute开发项目映射的role所有的引擎层面的权限。
  - DataWorks预设角色与MaxCompute引擎**生产项目**的Role没无权限映射关系，预设角色无法直接操作生产引擎（资源）。

  综上，DataWorks的RAM用户被添加为管理员角色或开发角色的成员后，会拥有开发环境（MaxCompute引擎开发项目）所有权限，但默认没有生产环境（MaxCompute生产项目）的操作权限，如果需要在生产环境访问生产表，需要在数据地图中单独申请生产环境表权限。

  在开发代码编译调测时，您主要在**数据开发**页面进行操作，在任务代码编译调测完成后，可发布至生产环境，后续在**运维中心**页面执行生产环境的任务。

操作页面	访问开发环境表	访问生产环境表
数据开发页面	○ 示例代码： `select col1 from tablename` ○ 操作结果说明：用个人账号访问开发环境下的表，即用个人账号访问 `projectname_dev.tablename` 。	○ 示例代码： `select col1 from projectname.tablename` ○ 操作结果说明：用个人账号访问生产环境下的表，即用个人账号访问 `projectname.tablename` 。
运维中心页面	不支持	○ 示例代码： `select col1 from tablename` ○ 操作结果说明：用调度引擎指定账号访问生产环境下的表，。即调度引擎指定账号访问 `projectname.tablename` 。

- **引擎权限管控：E-MapReduce**

  当前工作空间使用的是E-MapReduce引擎时，DataWorks预设角色与引擎无直接权限映射关系，绑定EMR引擎时，您可以选择**快捷模式**或**安全模式**，两种模式下的绑定配置和配置过程中的权限操作不一致，详情请参见准备工作：绑定EMR引擎。

- **快捷模式**

操作页面	访问开发环境表	访问生产环境表
数据开发页面&运维中心页面	统一使用Hadoop账号执行。	

- **安全模式**

操作页面	访问开发环境表	访问生产环境表	原理
数据开发页面	使用在绑定引擎时配置的**开发环境**指定账号访问所有引擎资源。	不支持	通过为DataWorks工作空间下的成员配置LDAP权限映射，实现控制每个子账号在DataWoks操作时的EMR底层权限控制的目的。 在DataWorks上，使用阿里云主账号或RAM用户下发代码的同时，EMR集群内会匹配对应的同名用户来运行任务。管理者可以使用EMR集群内的Ranger组件对每个用户进行权限管控，最终实现不同阿里云主账号、任务责任人或RAM用户在DataWorks上运行EMR任务时，拥有对应不同数据权限。
运维中心页面	不支持	使用在绑定引擎时配置的**生产环境**指定账号访问所有引擎资源。	

- **引擎权限管控：其他引擎**

  当前工作空间使用的其他引擎时，预设角色与引擎无直接关系。您在数据开发界面执行任务是否有权限与您在引擎配置中的配置账号有关。

- **DataWorks界面权限管控**
  - 模块维度权限管控

    DataWorks支持自定义DataWorks角色，来控制某个角色是否有某个模块的读写权限，详情可参考文档角色及成员管理：空间级。

  - 模块细节功能管控

    DataWorks上如果有功能置灰或者没有功能入口，请确认是否有相应的权限。DataWorks上预设角色拥有的权限不同，具体可参考文档附录：预设角色权限列表（空间级）。

  - **操作流程管控**

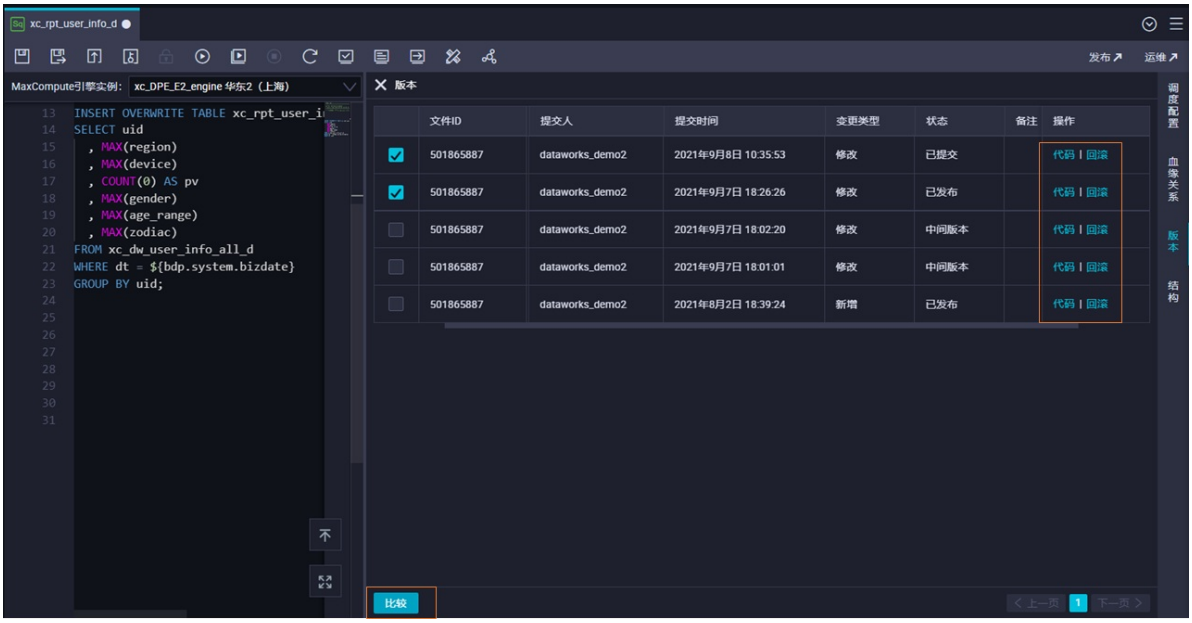
    DataWorks提供操作权限控制能力，您可以在敏感行为发生时做到第一时间阻断，支持人工干预或自定义事件检查逻辑，流程管控可分为阻断操作流程和不阻断操作流程仅通知。

    DataWorks提供代码评审功能，评审功能分为强制代码评审（阻断操作流程）和消息发送（不阻断操作流程仅通知）的功能，同时支持您针对节点重要性（节点所在基线）来对指定优先级的基线开启代码强制评审。您可以开放数据通过开放相关接口或功能，让您可以第一时间知道的核心变更消息并对此做出相关措施，旨在便于用户及时监控变更并作出响应。您通过开放消息功能订阅DataWorks项目中表变更、任务变更等消息，并实现个性化自动化响应。

操作审计

- 恢复已删除节点和表数据
  - 节点恢复：数据开发回收站可以还原近期删除的节点，但需注意，节点还原后节点ID是新生成的，与原ID不一致。
  - MaxCompute表数据恢复：DataWorks提供数据备份与恢复功能，系统会自动备份数据的历史版本（例如被删除或修改前的数据）并保留一定时间，详情请参见备份与恢复。
- 节点版本对比与回滚

  您可以在数据开发界面找到该节点，在节点配置右侧版本处进行版本对比回滚操作，详情请参见版本。



- 获取界面操作审计日志，如界面下载数据的操作

  DataWorks已对接操作审计（ActionTrail）中，您可以在ActionTrail中查看及检索阿里云账号最近90天的DataWorks行为事件日志。后续可以通过ActionTrail将事件日志投递至日志服务LogStore或指定的OSS Bucket中，实现对事件的监控和告警，满足及时审计、问题回溯分析等需求。详情请参考通过操作审计查询行为事件日志。
- 数据脱敏与泄露数据溯源

  如果您文件比较重要，为防止文件泄露，您可以通过数据保护伞功能的脱敏配置，对重要数据进行脱敏规则设置，并可依据数据水印功能对泄露的数据进行溯源。详情请参考文档数据脱敏管理。
- MaxCompute表权限的权限审计

  您可以进入安全中心，在数据访问控制的**权限审计**处，查看拥有表权限的人员列表、权限详情以及权限有效期，并支持回收表权限。

2.添加用户及定制化展示

DataStudio的功能模块及概念较多，您在实际使用时可能无法快速上手，为帮助您降低DataStudio的使用门槛，DataStudio会根据您的角色权限展示预设的功能模块，并支持按照您的需要定制化展示所需模块。本文为您介绍如何添加用户角色权限及定制化展示DataStudio功能模块。

背景信息

新用户使用DataStudio前，您需要为该用户添加相应的角色权限，不同角色权限可执行的操作及DataStudio支持的预设模块不同。同时，您可以根据自己的需要调整DataStudio左侧导航栏展示的模块。

- 添加用户角色权限，详情请参见准备工作：添加用户角色权限。
- 不同角色权限DataStudio的预设模块，详情请参见界面认识：预设模块说明。
- 定制化展示DataStudio的模块，详情请参见改变布局：定制化展示模块。

准备工作：添加用户角色权限

DataWorks提供了**项目所有者**、**空间管理员**、**数据分析师**、**开发**、**运维**、**部署**、**访客**、**安全管理员**、**模型设计师**等角色，您需要提前为用户授权相关角色，才能执行该角色支持的特定操作。具体说明如下：

- 不同角色支持的操作权限不同，详情请参见附录：预设角色权限列表（空间级）。
- 不同角色的用户进入DataStudio后，DataStudio界面左侧导航栏展示的预设模块不同，详情请参见界面认识：预设模块说明。
- 授予用户不同角色权限，详情请参见角色及成员管理：空间级。

界面认识：预设模块说明

不同角色的新用户进入DataStudio时，DataStudio左侧导航栏展示的预设模块如下表。

角色	模块展示
• **空间管理员** • **项目所有者** • **安全管理员** • **模型设计师** • **开发** • **运维**	• 数据开发（周期调度） • 数据开发（手动触发） • 表管理 • 临时查询 • 运行历史 • 回收站 • 发布任务 • 运维中心 **说明** **安全管理员**、**模型设计师**、**运维**角色在DataStudio界面只有只读权限。不同角色在DataStudio界面的操作权限不同，详情请参见附录：预设角色权限列表（空间级）。

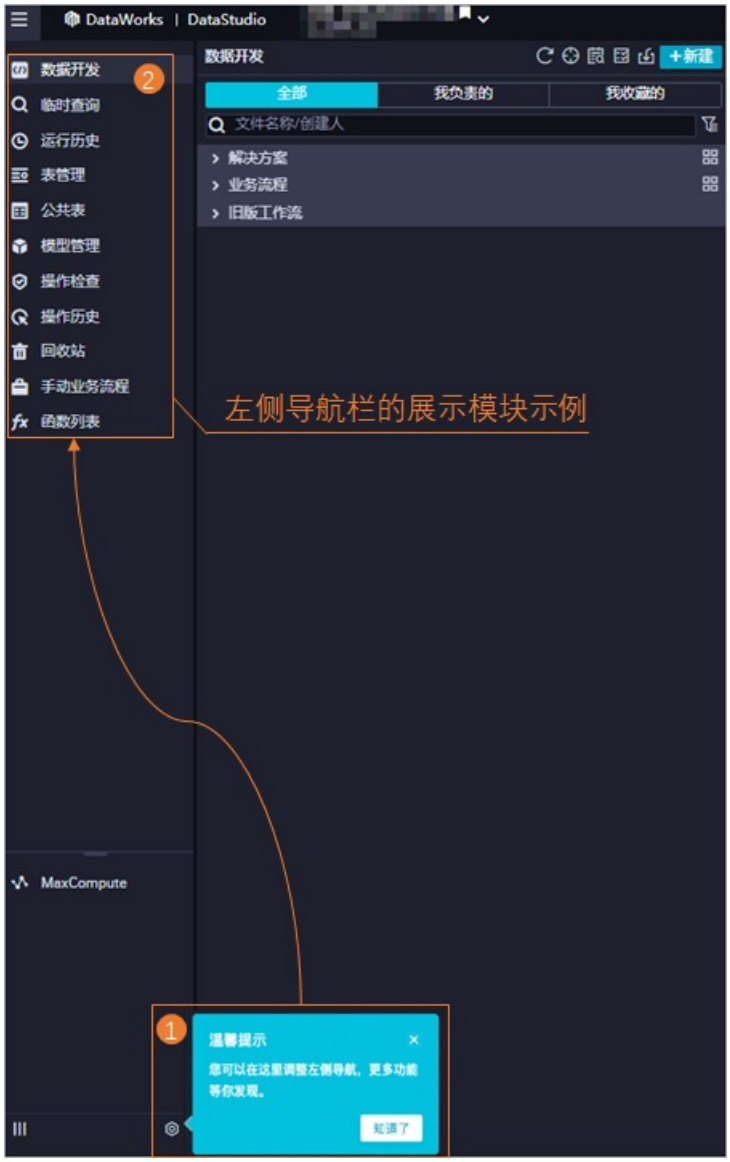
角色	模块展示
数据分析师	• 临时查询 • 公共表 • 运行历史 • 回收站
访客	• 数据开发（周期调度） • 数据开发（手动触发） • 临时查询

改变布局：定制化展示模块

您可以根据自己的需要调整DataStudio左侧导航栏的模块，操作步骤如下。

1. 登录DataWorks控制台，选择目标区域后，在左侧导航栏单击**工作空间列表**。
2. 单击目标工作空间后的**数据开发**，默认进入该工作空间的DataStudio功能模块。
3. 调整DataStudio左侧导航栏模块（即区域2展示的功能）。

   新用户可根据区域1的提示，在DataStudio界面左侧导航栏底部单击图标，进入**设置** > **个人设置**页面，即可在**模块管理**区域选择需要在DataStudio左侧导航栏展示的模块。详情请参见个人设置。


DataWorks | DataStudio
数据开发
+新建
全部
我负责的
我收藏的
文件名称/创建人
解决方案
业务流程
旧版工作流
数据开发
临时查询
运行历史
表管理
公共表
模型管理
操作检查
操作历史
回收站
手动业务流程
函数列表
左侧导航栏的展示模块示例
MaxCompute
温馨提示
您可以在这里调整左侧导航，更多功能等你发现。
知道了

3.数据开发功能索引

本文为您介绍DataWorks数据开发（DataStudio）界面各按钮的功能，方便您了解数据开发模块的整体布局，快速了解各组件、模块的使用并获取相关文档。

进入数据开发

1. 登录DataWorks控制台，选择目标区域后，在左侧导航栏单击**工作空间列表**。
2. 单击目标工作空间后的**数据开发**，即可进入该工作空间的数据开发模块。

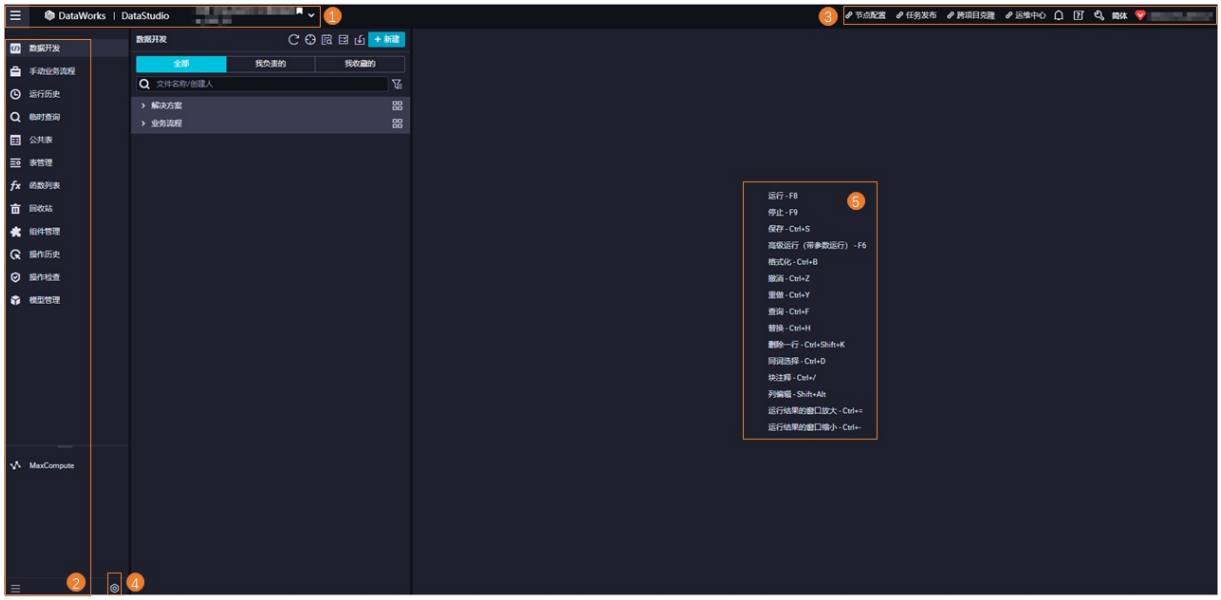
进入数据开发后，您可以创建业务流程及不同类型的节点进行相关开发操作，详情请参见创建业务流程及创建节点。

不同开发操作的界面功能存在差异，您可以根据本文快速了解对应操作的界面功能。

- DataStudio界面总览，详情请参见DataStudio界面总览。
- 数据开发（业务流程）界面功能，详情请参见数据开发（业务流程）界面功能。
- 数据开发（业务流程）的快捷菜单，详情请参见数据开发（业务流程）快捷菜单。
- 数据开发（节点）界面功能，详情请参见数据开发（节点）界面功能。
- 数据开发（节点）的快捷菜单，详情请参见数据开发（节点）快捷菜单。

DataStudio界面总览

DataStudio界面总体介绍如下图所示。



区域	描述

区域	描述
1	• 切换工作空间。 该区域为您展示了当前登录数据开发模块的工作空间。单击图标即可切换至当前地域的其他工作空间。 • 进入DataWorks其他模块。 单击图标即可选择进入数据集成、运维中心等其他模块。 ○ **数据集成**：数据集成。 ○ **数据建模**：数仓规划、数据标准、数据指标、维度建模、DATABLAU。 ○ **数据开发与运维**：DataStudio（数据开发）、运维中心（工作流）、任务发布、代码评审、Holo Studio。 ○ **数据治理**：数据地图、数据质量、安全中心、数据保护伞、数据治理中心。 ○ **数据分析**：数据分析。 ○ **数据服务**：数据服务。 ○ **机器学习**：机器学习PAI。 ○ **其他**：**发布中心**审批中心、Function Studio、资源优化、**DataWorks（首页）**、迁移助手、全局成员管理。 • 返回DataWorks控制台。 单击图标，在当前页面左下角单击返回控制台图标，即可返回DataWorks控制台。

区域	描述
2	在该区域单击图标，即可展示对应功能按钮的名称。 • 数据开发：用于周期调度任务的开发，支持基于各类引擎创建不同节点进行数据开发，该模块开发的任务可发布生产进行运维。 说明 您需要绑定相应类型的引擎后，才可以基于该引擎进行数据开发。 • 手动业务流程：用于手动触发式任务的开发，该模块开发的任务可发布生产进行运维。 • 运行历史：用于查看在DataStudio界面测试运行的历史记录，当前支持保留3天的历史记录。 • 临时查询：用于进行单次简单的测试查询，无法发布生产运维。 • 公共表：用于查看当前登录的阿里云账号下所有的生产表。 • 表管理：用于使用可视化方式对目标表执行相关操作。支持的表操作与表对应的引擎可执行的操作一致。 • 函数列表：MaxCompute系统自带函数的相关介绍。 • 回收站：用于管理在**数据开发**与**手动业务流程**中删除的节点、资源及函数。 • 组件管理：组件是一种带有多个输入参数和输出参数的SQL代码过程模板，SQL代码过程的处理通常会引入一到多个源数据表，通过过滤、连接和聚合等操作，加工出新业务需要的目标表。 • **操作历史**：可以通过操作类型、操作人、操作时间进行筛选，查看当前工作空间中的历史操作记录。 • **操作检查**：可以通过操作类型、检查状态进行筛选，查看相应操作的详细信息。 • **模型管理**：使用DATABLAU模块后，用于对DATABLAU中的数据模型进行管理。 • **MaxCompute**：单击MaxCompute即可显示下列子模块。 ◦ MaxCompute资源：用于管理MaxCompute引擎现有的资源。您可以通过该功能查看资源的操作记录。同时，支持将不在DataWorks中上传的资源通过此功能加载至DataWorks的数据开发进行管理。 ◦ MaxCompute函数：用于管理MaxCompute引擎现有的函数。您可以通过该功能查看函数的操作记录。同时，支持将不在DataWorks上注册的函数通过此功能加载至DataWorks的数据开发进行管理。 说明 如果您当前的界面左侧导航栏模块展示不全，则可单击区域4的图标，在**个人设置**界面添加相应模块，详情请参见个人设置。

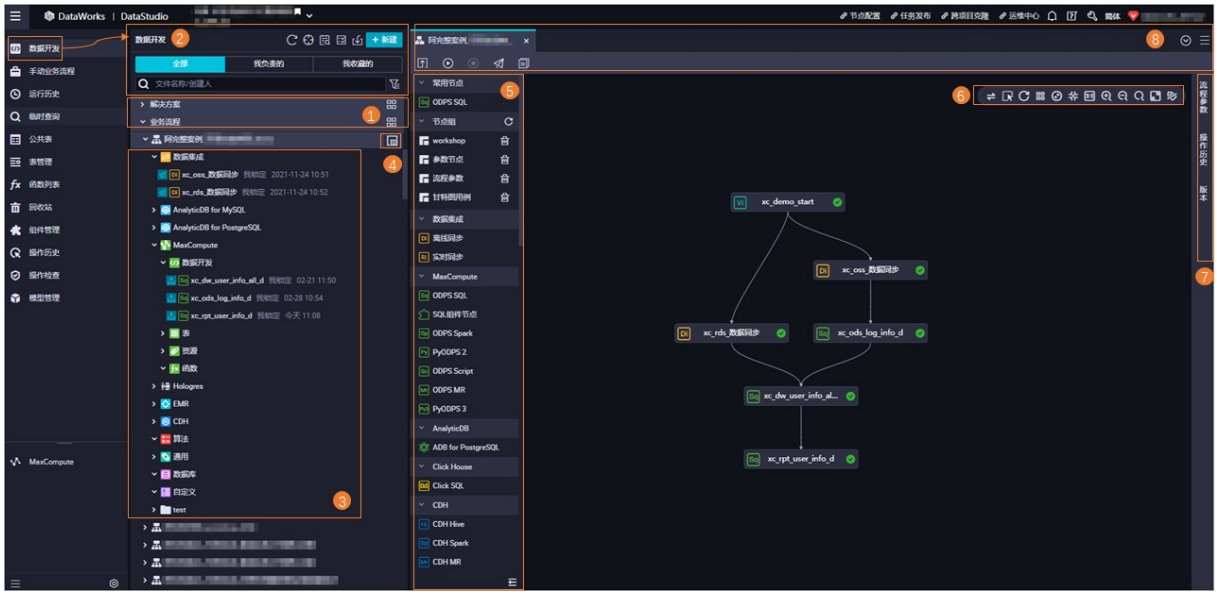
区域	描述
3	DataStudio中进入其他模块的快捷入口： • 节点配置：用于管理自定义节点及节点插件，进行个性化数据开发。同时，能够满足多样的数据质量定制化需求。配置完成后，您可以在数据开发界面选择该类型节点来编写SQL语句，SQL语句运行时，DataWorks会通过您后台定义的插件逻辑进行解析并执行。新增自定义节点前您需要先开发自定义插件的处理逻辑。 • 任务发布：用于将数据开发界面开发完成的节点发布至生产环境，您可以在发布流程中执行相关管控操作。 • 跨项目克隆：您可以利用跨项目克隆功能实现计算、同步等类型的任务在工作空间之间的克隆迁移。 • 运维中心：用于快速跳转至运维中心对任务进行运维操作。运维中心分开发运维中心和生产运维中心，生产运维中心生承担生产调度任务的整体运维管控。 DataWorks各模块的通用功能： 说明 本文以DataStudio界面为例，为您讲解如下通用功能，其他模块对应界面，该类功能相同。 • **消息中心（）**：用于发送产品侧功能变更的消息通知，方便您及时获取产品最新信息。 • **互动学习（）**：用于提供相应的产品功能说明，当您有相关需要时，可使用该功能快速查看帮助内容。 • **工作空间管理（）**：用于快速进入工作空间配置界面，您可以在该界面查看工作空间配置的基本信息、调度信息、白名单详情及引擎绑定情况。详情请参见配置工作空间。 • **语言切换**：单击当前显示的语言，即可进行语言（中英文）的切换。 • **账号信息**：单击当前登录账号即可查看该账号的个人信息、工作台任务概况。

4	系统配置，包括如下内容：
	• 个人设置：用于配置管理模块、编辑器功能、界面风格等。
	• 代码模板：用于管理语句代码的模板，使其呈现所需样式。
	• 调度设置：用于配置启用调度设置页面的调度周期功能，启用后周期任务才能自动调度运行。
	• **安全设置与其他**：
	  ◦ 安全设置：用于控制在工作空间中查询页面内容时，是否对敏感信息进行脱敏。
	  ◦ 其他配置：用于开启代码强制评审，配置代码审核人员，把控开发任务的代码质量；也可以批量删除所有不再使用的DataBlau DDM数据模型。

5	数据开发编辑器常用的快捷键。更多快捷键，详情请参见编辑器快捷键列表。

数据开发（业务流程）界面功能

进入DataStudio后，默认进入数据开发模块，您需要在该模块先创建业务流程，组织后续业务开发。创建业务流程详情请参见创建业务流程。业务流程的功能界面如下图所示。



区域	功能描述
1	• 解决方案：用于将一类业务流程组合为一个解决方案，业务流程可以被多个解决方案复用。解决方案支持使用列表及图形化的方式呈现。 • 业务流程：用于实际业务开发，业务流程为业务的抽象实体，帮助您使用业务视角来组织数据代码开发。 单击图标，即可呈现当前工作空间下的所有解决方案或业务流程。

区域	功能描述

2	• **刷新（）**：用于手动刷新目录树，当您对业务流程或解决方案进行变更操作，可手动刷新对应目录树。
	• **定位（）**：用于快速定位当前打开的文件。
	• 代码搜索（）：用于通过关键字搜索代码片段，快速定位数据开发、手动业务流程、临时查询、回收站中包含该代码片段的所有节点及相关代码片段的详细内容。当目标表数据产生变更，您需要查找操作源（即导致目标表数据变更的任务）时，可以使用该功能。
	• 批量操作（）：用于快速对表、资源、函数进行批量修改（包括修改责任人、引擎实例、调度资源组、调度重跑属性、调度类型、调度周期、调度超时时间等操作）。
	• 导入（）：用于快速将本地数据上传至目标表中。目前仅支持上传数据至MaxCompute表中。
	• **快捷新建**（  ）：用于快速创建业务流程，以及各类型的节点、表、资源、函数等。
	• **解决方案及业务流程目录树展示**：
	  ○ **全部**：目录树基于解决方案及业务流程展示当前工作空间下所有已创建的文件（节点、资源、函数等）。
	  ○ **我负责的**：目录树基于解决方案及业务流程展示当前登录账号为负责人的文件（节点、资源、函数等）。
	  ○ **我收藏的**：目录树基于解决方案及业务流程展示当前登录账号收藏的文件（节点、资源、函数等）。
	• **文件查找**：
	  ○ **精确查找**：您可以输入**文件名称**或**创建人**，单击  图标查找指定节点。
	  ○ **按节点类型查找**：您可以单击  图标，通过筛选节点类型，查找指定类型的所有节点。指定节点类型后，则目录树将仅展示当前工作空间中该类型的节点。
	说明 您还可以根据业务需求选择是否需要**隐藏引擎实例**及**隐藏节点类型文件夹**，隐藏后，目录树将不会呈现相应内容。
	■ **隐藏引擎实例**及**隐藏节点类型文件夹**仅适用于新版业务流程。
	■ 通常，目标引擎下仅包含一个引擎实例时，建议您将其隐藏。
	■ 如果您不需要使用数据开发、表、资源、函数等节点类型文件夹时，则可以将其隐藏。
	说明 如果您当前的工作空间为新建的工作空间，请先创建业务流程，并在业务流程内新建节点进行数据开发。创建业务流程详情请参见管理业务流程。

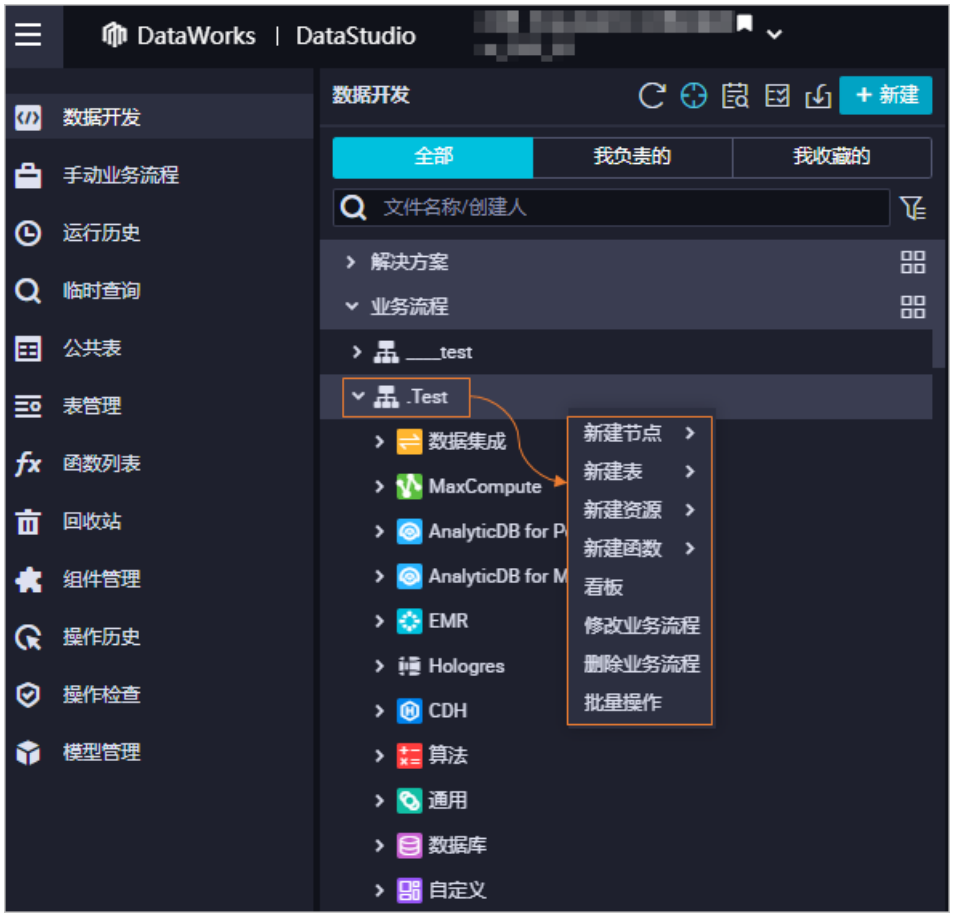
区域	功能描述
3	使用目录树的方式对各业务流程中的节点、表、资源、函数进行管理： • **业务流程**：业务开发的单位，用于进行具体的业务开发工作。 • **节点**：代码开发的最小单位，支持对应引擎、算法、数据集成、数据库、通用节点及自定义节点进行代码开发。 • **表**：使用可视化方式操作表。 • **资源**：使用可视化方式上传资源。 **说明** 当前仅支持MaxCompute、E-MapReduce、CDH引擎使用可视化方式上传资源。 • **函数**：使用可视化方式注册函数。 **说明** 当前仅支持MaxCompute、E-MapReduce、CDH引擎使用可视化方式注册函数。 您可以通过节点名称前的图标查看该节点的状态： • 图标：表示节点未提交。单击该图标即可快速提交节点。 • 图标：表示节点未发布。单击该图标即可进入发布中心发布节点。 同时，节点名称后为您展示了最近一次编辑该节点的时间。 双击业务流程名称，即可进入业务流程编辑页面（区域5~8），您可以在该页面进行数据开发。
4	资源组编排（）：用于在数据开发阶段，批量修改指定业务流程下目标节点使用的调度资源组。当您的工作空间中有多个调度资源组时，可以根据实际业务需求，使用该功能快速为目标节点重新分配资源组，促进资源的合理使用。修改后您需要通过批量发布将该操作发布至生产运维中心，使生产环境节点使用的调度资源组修改操作生效。
5	• **常用节点**：为您展示当前工作空间中常用的类型节点，方便您快速筛选创建目标类型节点。 • 节点组：用于跨业务流程引用一批节点，您可以将业务流程内复用率较高的节点组合为一个节点组，以便在其他业务流程中快速复用该节点组（即快速克隆这批节点）。 • 快速创建节点：您可以将数据集成、MaxCompute、E-MapReduce等目录下的节点直接拖拽至右侧业务流程编辑面板，创建对应类型的节点。

区域	功能描述
6	业务流程可视化操作面板详情1： • **切换布局（）**：您可以切换当前业务流程编辑面板的布局为**纵向**、**横向**或**网格**。 • **框选（）**用于将选中的节点组合为节点组，批量执行节点相关的操作。 • **刷新（）**刷新当前业务流程。当您对业务流程执行变更操作时，可手动刷新，获取最新界面。 • **格式化（）**：将业务流程中各节点的位置格式化为水平对齐。 • **适配窗口（）**：根据当前界面的窗口大小，自动适配业务流程的布局。 • **居中（）**居中当前业务流程的各节点。 • **1：1（）**用于将当前业务流程的各节点与编辑面板按照1：1比例布局。 • **放大（）**放大当前业务流程的各节点。 • **缩小（）**：缩小当前业务流程的各节点。 • **查找（）**输入关键字，搜索包含关键字的节点。 **说明** 查找方式为模糊匹配，即输入关键字后，DataWorks会展示出当前业务流程中包含关键字的所有节点。 • **全屏（）**全屏显示当前业务流程。 • **隐藏引擎信息（）**用于显示或隐藏各节点的引擎信息。
7	业务流程可视化操作面板详情2： • 流程参数：用于批量替换代码中的统一变量，目前仅支持**ODPS SQL**类型节点使用流程参数。 • 操作历史：用于查看业务流程面板中节点的相关操作记录。 • 版本：业务流程每次提交都会生成一个新的版本，您可以在此处查看业务流程的各个版本记录及版本详情。

区域	功能描述
8	业务流程可视化操作面板详情3： • **提交（）**：批量提交业务流程内更新的节点至任务发布界面。 • **运行（）**：运行当前业务流程下所有节点。 • **停止运行（）**：运行中的业务流程可以选择批量终止运行业务流程中的节点。 • 发布（）：快速在任务发布界面定位该业务流程下待发布的节点。节点的发布操作。 • 前往运维（）：快速进入生产运维中心，查看节点的运维详情。 • **搜索**：当前如果打开的页签较多，您可以单击图标，使用下拉列表查看所有页签。 • **关闭页签**：单击图标，关闭指定页签。

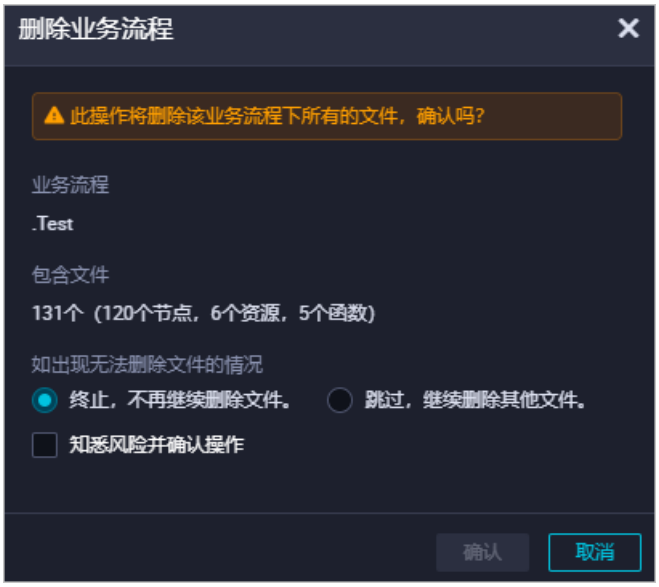
数据开发（业务流程）快捷菜单

将鼠标悬停至目标业务流程，单击鼠标右键，即可显示业务流程的快捷菜单，相关功能如下图所示。



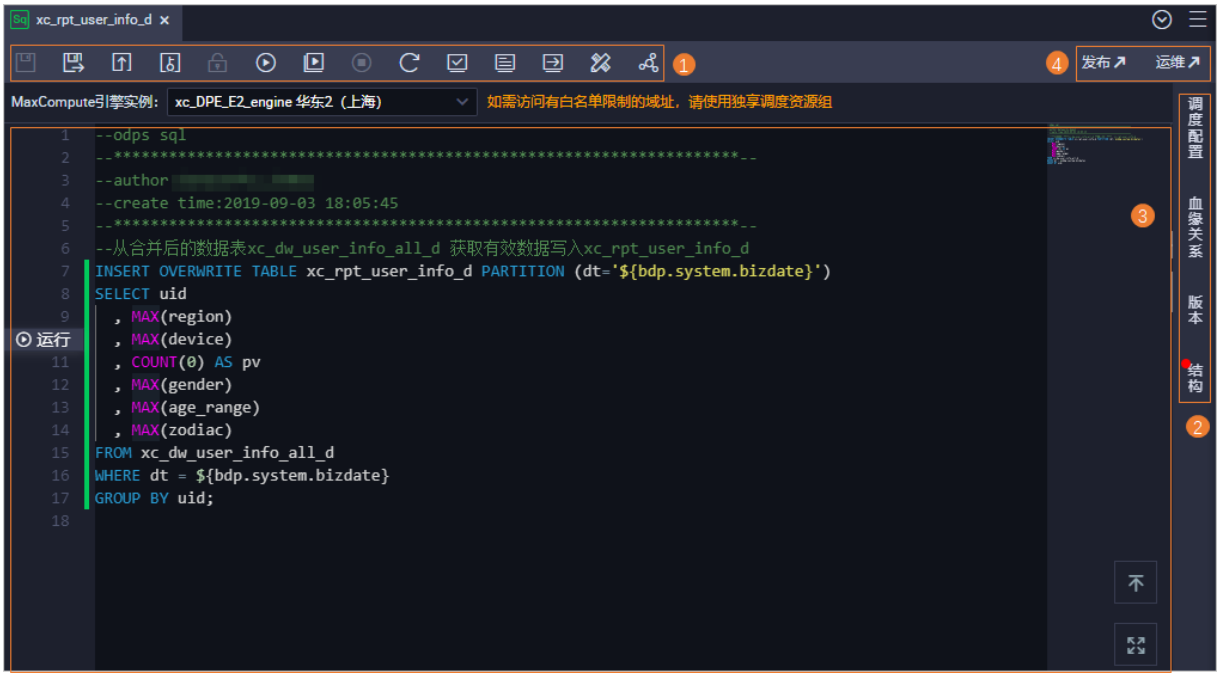
功能	描述

功能	描述
新建节点	用于快速新建各类型节点。 新建节点时，系统会显示最近一次使用的节点类型，单击该类型，即可自动输入最近一次使用的**引擎实例**及**节点类型**。如果您需要创建近期使用过的类型节点，则可使用该方式快速创建。 
新建表	用于快速新建各类型表。
新建资源	用于快速新建各类引擎资源。 **说明** 当前仅支持新建MaxCompute、CDH、EMR引擎资源。
新建函数	用于快速新建各类引擎函数。 **说明** 当前仅支持新建MaxCompute、CDH、EMR引擎函数。
看板	用于进入目标业务流程的编辑面板。
修改业务流程	用于修改业务流程的名称、负责人、描述等信息。

功能	描述
删除业务流程	删除当前业务流程。 **说明** 此操作将批量删除该业务流程下所有文件，请您谨慎操作。 删除业务流程时，针对无法删除的文件，您可以选择如下处理策略： • **终止，不再继续删除文件**：默认选择该选项。执行删除操作时，当遇到无法删除的文件，删除操作会被中断。已删除的文件将成功删除。 • **跳过，继续删除其他文件**：执行删除操作时，当遇到无法删除的文件，系统会跳过该文件，继续删除其他文件。 
批量操作	用于批量修改业务流程中节点、资源、函数的责任人、引擎实例、调度属性等，并支持批量提交并发布，将变更操作发布至生产环境生效。

数据开发（节点）界面功能

业务流程创建完成后，您可以根据开发需求创建不同类型的数据开发节点，详情请参见创建开发节点。不同类型的节点功能类似，本文以ODPS SQL节点为例，为您介绍数据开发节点编辑界面的功能。



区域	功能描述

区域	功能描述
1	节点开发相关功能按钮： • **保存（）**：保存当前节点的代码及相关配置。 • **另存为临时查询文件（）**：将当前代码另存为一个临时文件，您可以进入**临时查询**页面查看。详情请参见创建临时查询。 • **提交（）**：提交当前节点。 • **提交并允许他人编辑该文件（）**：提交当前节点，并允许其他用户编辑该节点的代码。 • **偷锁编辑（）**：用于非节点责任人编辑节点。 • **运行（）**：运行当前节点的代码。运行SQL代码时，您只需要给SQL语句中的变量赋一次值，即使节点的代码发生变更，也会保留初始的赋值。 **说明** 如果您创建的节点没有选择调度资源组，则运行任务时，系统会先提示您选择可用的调度资源组。 • **高级运行（带参数运行）（）**：使用配置的参数运行当前节点代码。**运行**代码时每次都需要手动给SQL语句中的变量进行赋值，**运行**的初始赋值会传递给**高级运行**，高级运行的自定义参数赋值后，会刷新当前运行的自定义参数。 **说明** 如果您创建的节点没有选择调度资源组，则运行任务时，系统会先提示您选择可用的调度资源组。 • **停止运行（）**：停止正在运行的节点。 • **重新加载（）**：刷新节点页面，返回至上次保存的页面。 • **在开发环境执行冒烟测试（）**：在开发环境测试当前节点的代码。开发环境冒烟测试可以模拟调度参数在生产调度中参数的替换情况，选择业务日期后，根据您填写的调度参数替换该业务日期下的值。您可以通过该功能测试调度参数的替换情况。 **说明** 开发环境冒烟测试每次变更调度属性后，其中的参数配置需要重新保存并提交，然后选择开发环境冒烟测试，否则替换的调度属性仍会是原来的值。 • **查看开发环境的冒烟测试日志（）**：查看运行在开发环境的节点运行日志。 • **前往开发环境的调度系统（）**：跳转至开发环境的运维中心进行相关运维操作。详情请参见查看周期实例。 • **格式化（）**：对当前节点代码排序，常用于单行代码过长的情况。 • **分享（）**：分享当前节点给其他用户。

区域	功能描述
2	**调度配置**： • 基础属性：查看调度节点的名称、ID、类型，并配置责任人、描述等基本信息。 • 参数：任务调度时需要使用的参数，可使用调度参数实现参数的动态取值。 • 时间属性：用于定义节点发布生产调度系统后在调度环境下的相关属性。您可以通过调度配置的时间属性，配置节点生成周期实例的方式，实例调度周期与执行时间，是否支持重跑，任务执行超过多长时间自动退出等。 • 资源属性：配置节点调度时需要使用的资源组。 • **调度依赖**：用于配置上下游节点的依赖关系，详情请参见配置同周期调度依赖、配置上一周期调度依赖。 • 节点上下文：用于上下游节点参数传递，多用于使用赋值功能通过节点上下文参数，将上游节点的查询结果传递至下游节点。 血缘关系：展示当前节点和其它节点的依赖关系和内部血缘关系。 版本节点每次提交、发布都将生成新的版本。您可以在版本面板查看节点历史版本、提交人、提交时间、变更类型、状态、备注等信息。版本的状态说明如下： • **已提交**：节点已提交至开发环境，在任务发布界面处于待发布状态。 • **已发布**：节点已经发布至生产环境，您可以在生产运维中心周期任务查看。详情请参见查看并管理周期任务。 • **中间版本**：节点提交一次后未发布，如果再提交一次，则上一次提交的版本将成为中间版本。 • **发布已取消**：节点提交后在任务发布界面将该条待发布记录取消发布，该版本的状态则会变为发布已取消。 结构：代码结构通过SQL算子进行可视化展示。
3	**SQL编辑器**：您可以根据业务需求在编辑器中编写任务的SQL语句。 • 单击图标，即可跳转至SQL语句的首行位置。 • 单击图标，即可全屏展示SQL编辑器。 • 单击运行图标，快速运行目标代码片段，测试代码片段编写是否正确。详情请参见调试代码片段：快捷运行 **说明** 鼠标单击代码行，才会显示该图标。
4	**发布运维操作**： • 发布：进入任务发布页面，您可以在该页面查看节点的发布详情，或进行节点发布后的生产运维操作。 • 运维：进入生产运维中心，执行节点相关的运维操作。

数据开发（节点）快捷菜单

将鼠标悬停至目标开发节点，单击鼠标右键，即可显示节点的快捷菜单，相关功能如下图所示。

功能	描述
重命名	修改目标节点的名称。
收藏	收藏目标节点后，单击数据开发目录树右上方的**我收藏的**，即可展示已收藏的节点。对于已收藏的节点，后续无需收藏时，则可在节点的快捷菜单单击**取消收藏**。
移动	移动目标节点至其他业务流程目录。
克隆	用于复制出一个具有相同节点类型、责任人及资源属性的节点。原节点和克隆节点根据不同名称进行区分。
查看历史版本	用于在版本面板查看节点历史版本、提交人、提交时间、变更类型、状态、备注等信息。
在运维中心中定位	进入**运维中心**查看节点的运行信息。如果节点分别提交至开发环境及生产环境，则您可以选择**在运维中心（生产）中定位**或**在运维中心（生产）中定位**
发起代码评审	提交当前节点的代码至评审人进行评审。开发人员提交的节点必须通过评审人对代码的审核才可以发布。

功能	描述
删除	删除该节点及其上下游依赖节点对该节点的依赖。已发布至生产环境的节点被删除后，您需要进入**任务发布**界面执行发布操作，发布后该节点才会在生产环境下线，详情请参见下线任务。

4.业务流程与解决方案

4.1. 创建解决方案

数据开发模式全面升级，包括工作空间>解决方案>业务流程三级结构，抛弃陈旧的目录组织方式。

背景信息

DataWorks对数据开发模式进行全面升级，按照业务种类组织相关的不同类型的节点，让您能够更好地以业务为单元、连接多个业务流程进行开发。

DataWorks通过**工作空间 > 解决方案 > 业务流程**三级结构，全新定义开发流程，提升开发体验：

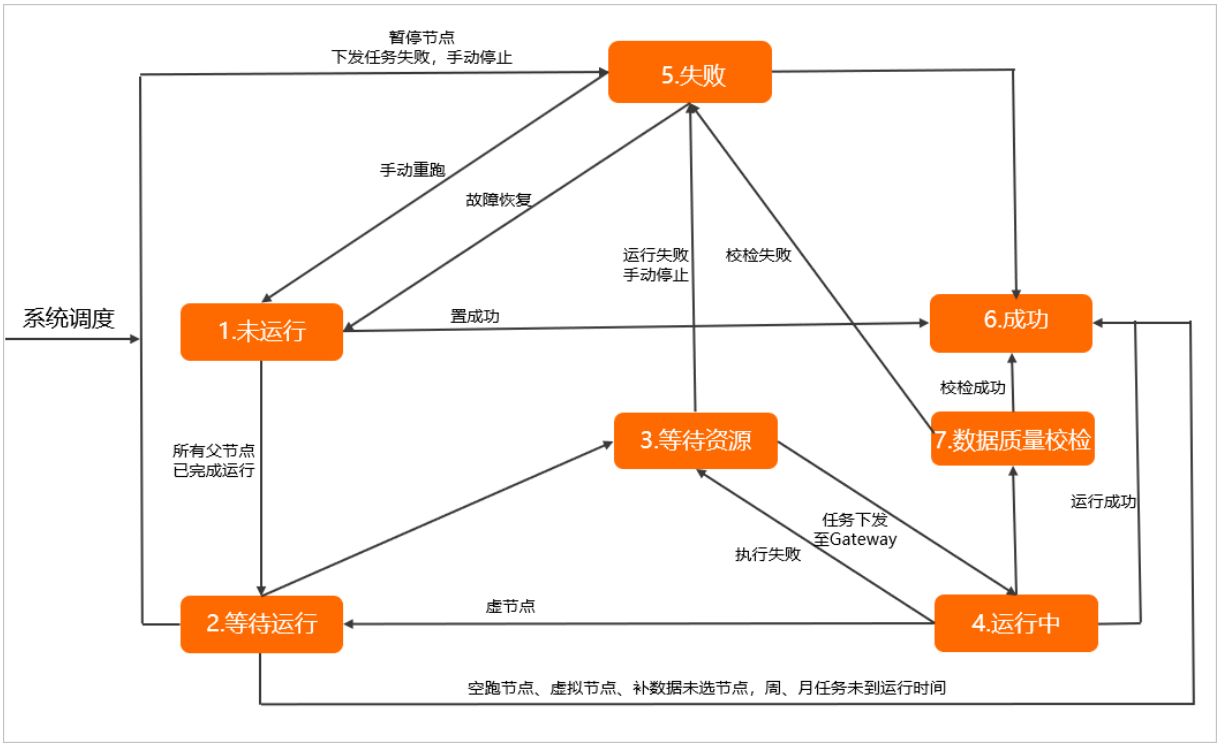
- 工作空间：权限组织的基本单位，用来控制您的开发、运维等权限。工作空间内成员的所有代码均可以协同开发管理。
- 解决方案：您可以自定义组合业务流程为一个解决方案。优势如下所示：
  - 包括多个业务流程。
  - 解决方案之间可以复用相同的业务流程。
  - 自定义组合而成的解决方案，可以让您进行沉浸式开发。
- 业务流程：业务的抽象实体，让您能够以业务的视角来组织数据代码开发。业务流程可以被多个解决方案复用。

  业务流程的优势如下：
  - 帮助您从业务视角组织代码，更清晰，并且提供基于任务类型的代码组织方式。每个节点类型下均支持创建多级子目录，右键单击相应的节点类型，选择**新建文件夹**即可（建议不超过4级）。

- 让您可以从业务视角查看整体的业务流程，并进行优化。
- 提供业务流程看板，开发更高效。
- 让您可以按照业务流程组织进行发布和运维。

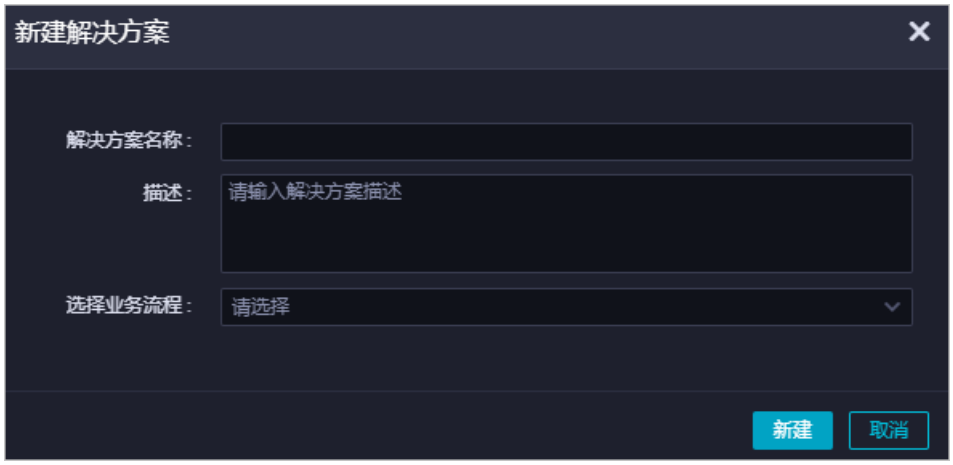
业务流程可以被多个解决方案复用，您只需要开发自己的解决方案。其他人可以在其它解决方案或业务流程中，直接编辑您引用的业务流程，构成协同开发。

任务状态机模型是针对数据任务节点在整个运行生命周期的状态定义，共有6种状态，您可以通过下图了解状态之间的转换逻辑。



操作步骤

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. 单击**+新建 > 新建解决方案**，在弹出的对话框中，配置各项参数。



参数	描述
解决方案名称	解决方案名称的长度不能超过128个字符。
描述	对解决方案的描述不能超过256个字符。
选择业务流程	您可以选择多个业务流程，组成一个解决方案。

3. 新建成功后，单击**解决方案**展开列表，进行以下操作：
   - 双击解决方案的名称，查看该解决方案下所有的业务流程。单击相应的业务流程名称，即可查看和编辑该业务流程，详情请参见管理业务流程。
   - 右键单击解决方案的名称，选择**解决方案看板**，查看选择的业务流程的节点，以及修改解决方案。
4. 发布和运维解决方案。

   鼠标悬停至解决方案名称上，会显示和图标：

   - 单击图标，进入**任务发布**页面。您可以查看当前解决方案下**待发布**状态的节点。

     发布任务的详情请参见发布任务。

     > **说明**
     > - 发布解决方案是指合并多个业务流程为一个解决方案，进行整体发布。
     > - 如果多个解决方案复用了同一个业务流程，解决方案下没有需要发布的节点时，则不会重新发布。
     > - 解决方案无法指定业务流程的运行顺序。您需要在节点中配置定时执行时间的规则，解决方案的执行顺序以实际实例中的定时时间为准。

   - 单击图标，进入**运维中心 > 周期任务运维 > 周期实例**页面，默认展示当前解决方案下所有的节点

的周期实例。

4.2. 管理业务流程

业务流程能够根据业务类别组织不同类型的节点，以业务为单元开发代码。本文为您介绍如何创建、设计、提交和查看业务流程，以及批量修改或删除业务流程中的节点。

背景信息

一个工作空间可以支持多种类型的计算引擎，也可以包含多个业务流程。一个业务流程是多种类型对象的集合，对象的类型包括数据集成、数据开发、表、资源、函数和算法等。

每种对象类型对应一个独立的文件夹，在每个对象类型文件夹下，支持继续创建子文件夹。为了便于管理，建议子文件夹的层数不要超过4层。 如果超过4层，可能说明您规划的业务流程结构过于复杂，建议将该业务流程拆分成两个或多个业务流程，并将相关的业务流程收纳到一个解决方案中进行管理，提升工作效率。

开发组织结构

您可以基于包括**工作空间 > 解决方案 > 业务流程**三级结构，对业务进行划分，您可以基于公司部门、公司业务或数仓层次进行规划分组。

结构层级	特征	定位
工作空间	不同的工作空间可以有不同的管理员、不同的内部成员，各工作空间拥有完全独立的成员角色设定以及引擎实例的各项参数开关。关于工作空间的规划请参见规划工作空间。	DataWorks支持的最大业务划分粒度，权限组织的基本单位，用来控制您的开发、运维等权限。工作空间内成员的所有代码均可以协同开发管理。
解决方案	您可以将一类业务流程划分为一个解决方案进行统筹管理，同时一个业务流程也可以被多个解决方案复用，您只需要开发自己的解决方案。其他人可以在其它解决方案或业务流程中，直接编辑您引用的业务流程，构成协同开发。	业务整合。
业务流程	业务的抽象实体，让您能够以业务的视角来组织数据代码开发。工作空间之间的业务流程、任务节点独立开发，互不影响。 业务流程两种形态，目录树与面板，让您从业务视角组织代码，资源类别更明确，业务逻辑更清晰。 • 目录树结构提供基于任务类型的代码组织方式。 • 业务流程面板提供流程化的业务逻辑展现方式。	具体的代码开发、资源组织单位。

数据开发基于业务流程下对应的节点进行开发操作，您可以在业务流程面板下新建一个或多个业务流程，每个业务流程按照引擎类型进行分组，每个引擎分组下再对数据开发类型节点、表、资源、函数进行一步分组，即一类业务使用的组件（节点、表、资源、函数）统筹在一个业务流程中，业务流程下仅展示当前业务流程中使用的组件。

- 在DataWorks上，具体的数据开发工作是基于业务流程开展的，所以您需要先新建业务流程，再进行后续的开发工作。
- 所有生产环境调度节点的代码变更都需要在数据开发界面修改完成后走发布流程进行发布。

> **说明**
>
> - 如果您工作空间下无可用的引擎或目录树上对应引擎不可见，请在工作空间配置界面确认是否已经开通并绑定对应引擎服务。业务流程下仅展示当前工作空间下已经绑定的引擎服务，引擎绑定详情可参考文档：配置工作空间。
> - 如果您无法操作部分功能，或者没有新建入口，请在工作空间配置的成员管理处确认是否有开发权限（即操作账号是否为阿里云主账号、是否被授予开发角色或空间管理员），或查看当前开通的DataWorks版本是否为要求的版本。
> - 如果目录文件超过4级，可能说明您规划的业务流程结构过于复杂，建议将该业务流程拆分成两个或多个业务流程，并将相关的业务流程收纳到一个解决方案中进行管理，提升工作效率。

创建业务流程

数据开发基于业务流程下对应的开发组件进行具体开发操作，所以您创建节点前需要先新建业务流程。

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. 鼠标悬停至**+新建**图标，单击**业务流程**。

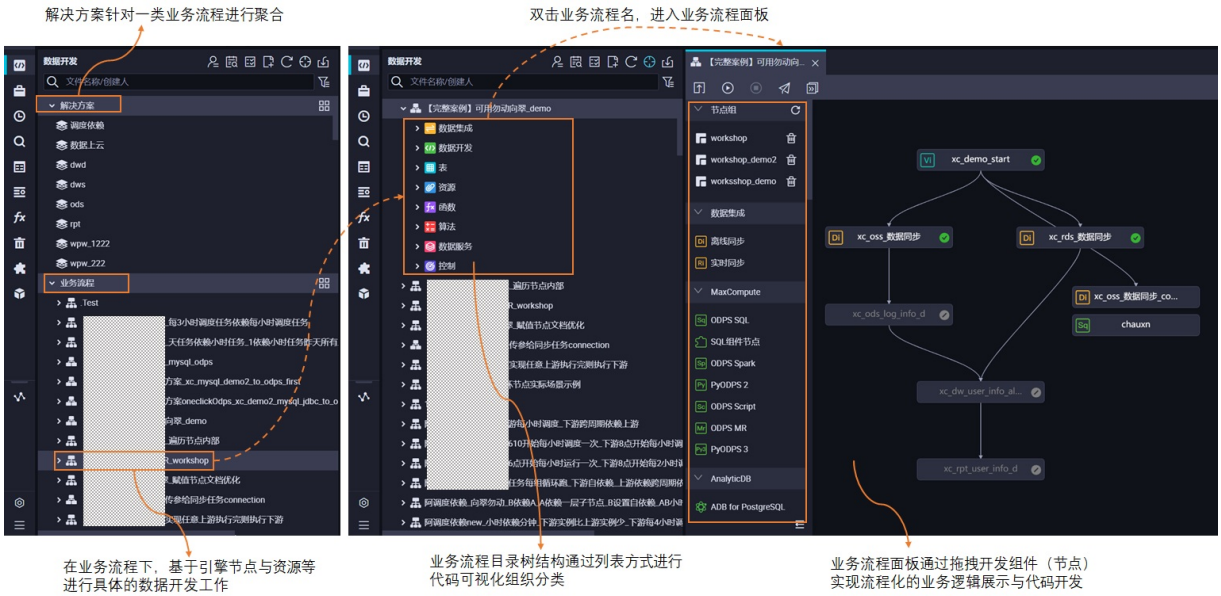
3. 在**新建业务流程**对话框中，输入**业务名称**和**描述**。

> 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。

4. 单击**新建**。

设计业务流程

代码开发都基于业务流程进行操作，您可以在具体的业务流程目录树选择具体引擎下的开发组件，新建节点进行开发（列表模式），同时，您也可以选择双击具体的业务流程名，进入业务流程面板进行可视化拖拽式多引擎任务混合编排（基于DAG图可视化拖拽方式）。



进行业务流程设计时：

- 建议单个业务流程下节点总数不要超过100个。

> 说明　业务流程中节点总数超过1000个时，该业务流程的DAG图将无法打开。

- DAG图模式下，您可以通过拖拽依赖线的方式设置节点调度依赖。当然，您也可以进入节点的调度配置界面，来手动编辑节点依赖关系。详情请参见同周期调度依赖逻辑说明。

- 列表模式下新建的节点，其业务流程可根据代码血缘关系来设置节点调度依赖，详情请参见同周期调度依赖逻辑说明。

开发业务逻辑

DataWorks将引擎能力进行封装，您可以基于引擎节点进行数据开发，无需接触复杂的引擎命令行，同时您也可以结合平台提供的通用类型节点进行复杂逻辑处理。

在业务流程内，您可以基于同步和计算节点等组件进行具体的业务流程开发。

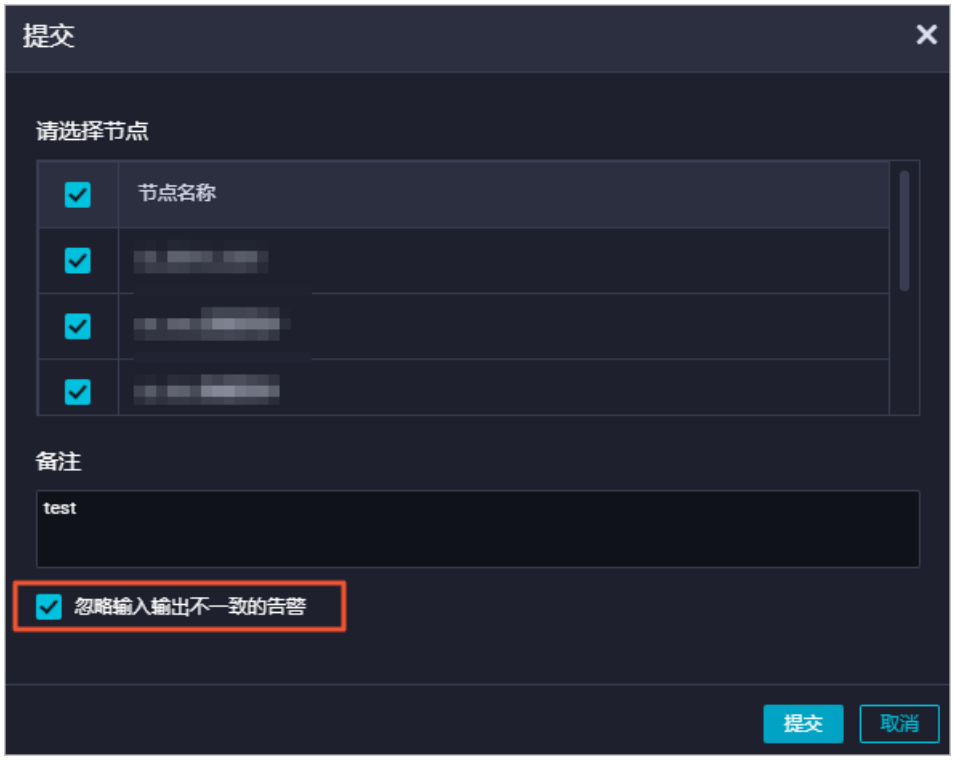
- 您可以在数据集成下，通过离线同步和实时同步组件节点，来将您其他数据库的数据同步到另一个库。
- 您可以基于业务流程该引擎分组下的数据开发来进行具体的数据清洗工作，如果代码开发过程中需要用到资源或函数，DataWorks也支持您通过可视化的方式来新增资源、注册函数。

> 说明
> - 目前DataWorks对引擎能力的封装、产品层面对开发能力的支持情况请参见选择数据开发节点。
> - 节点的调度依赖，调度属性相关配置请参见调度配置。

提交业务流程

标准模式工作空间下，数据开发界面仅作为节点任务的开发与测试页面，如果您需要将代码发布到生产环境，您可以批量提交该业务流程下的节点，并进入任务发布界面批量发布该业务流程下的节点项。

1. 业务流程设计并完成测试后，单击工具栏中的图标。
2. 在**提交**对话框中，选中需要提交的节点，输入**备注**信息，并根据业务需求选择是否**忽略输入输出不一致的告警**。如果您的输入输出内容和代码血缘分析不匹配时，当不勾选**忽略输入输出不一致的告警**，会产生相应的告警提示，详情请参见提交节点时提示：输入输出和代码血缘分析不匹配。



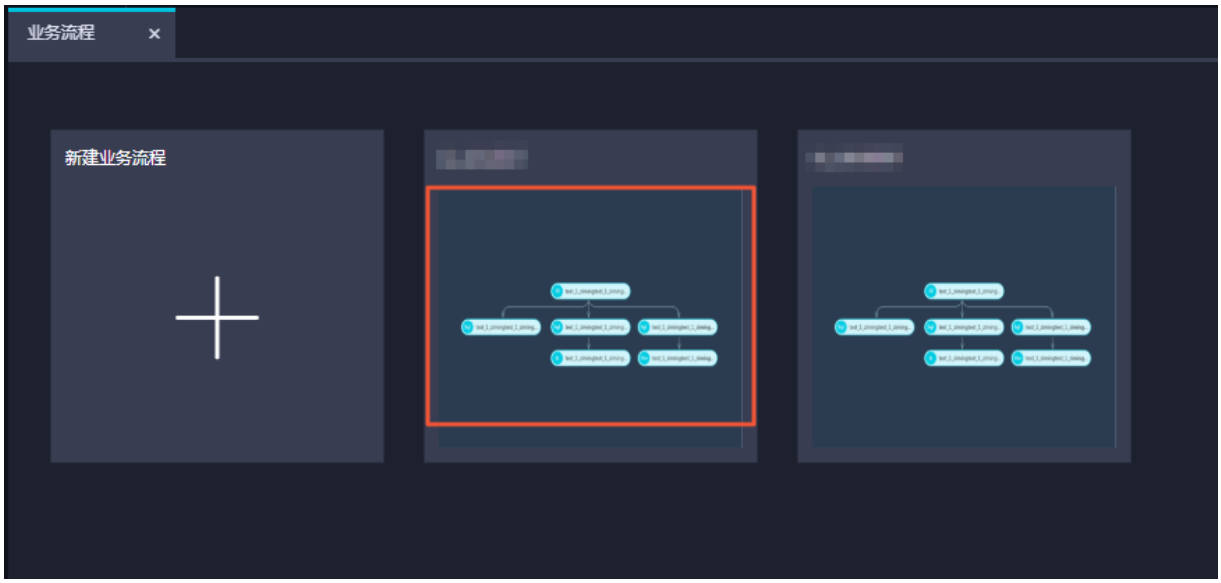
3. 单击**提交**。

> **说明** 如果您的节点已经提交过，在不改变节点内容的情况下，无法再次选择节点。此时输入**备注**后单击**提交**即可，节点属性等改动会被正常提交。

查看所有的业务流程

在**数据开发**页面，右键单击**业务流程**，选择**全部业务流程看板**，查看该工作空间下所有的业务流程。

单击某个看板，即可进入相应的业务流程看板。



基于解决方案管理业务流程

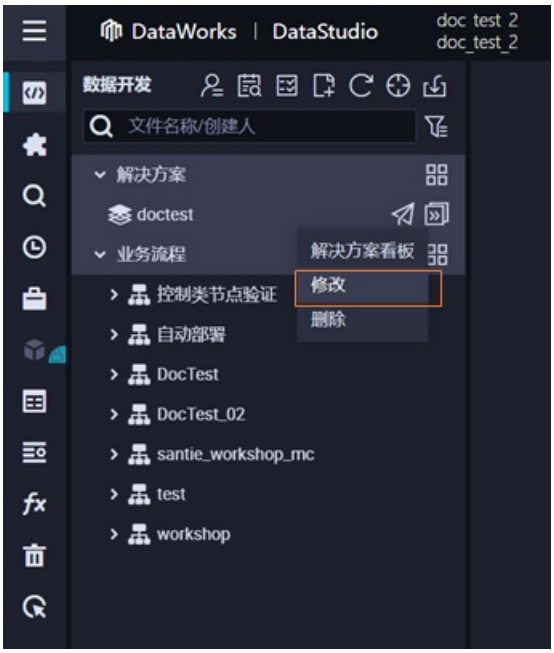
您可以自定义组合业务流程为一个解决方案，解决方案支持：

- 包含多个业务流程。
- 解决方案之间可以复用相同的业务流程。
- 自定义组合而成的解决方案，可以让您进行沉浸式开发。

通过解决方案管理业务流程时，您可以：

- 将单个业务流程纳入到具体的解决方案中。

- 通过解决方案修改面板批量添加业务流程。

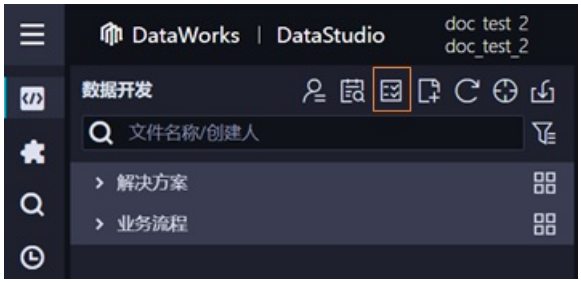


批量修改或删除业务流程的节点

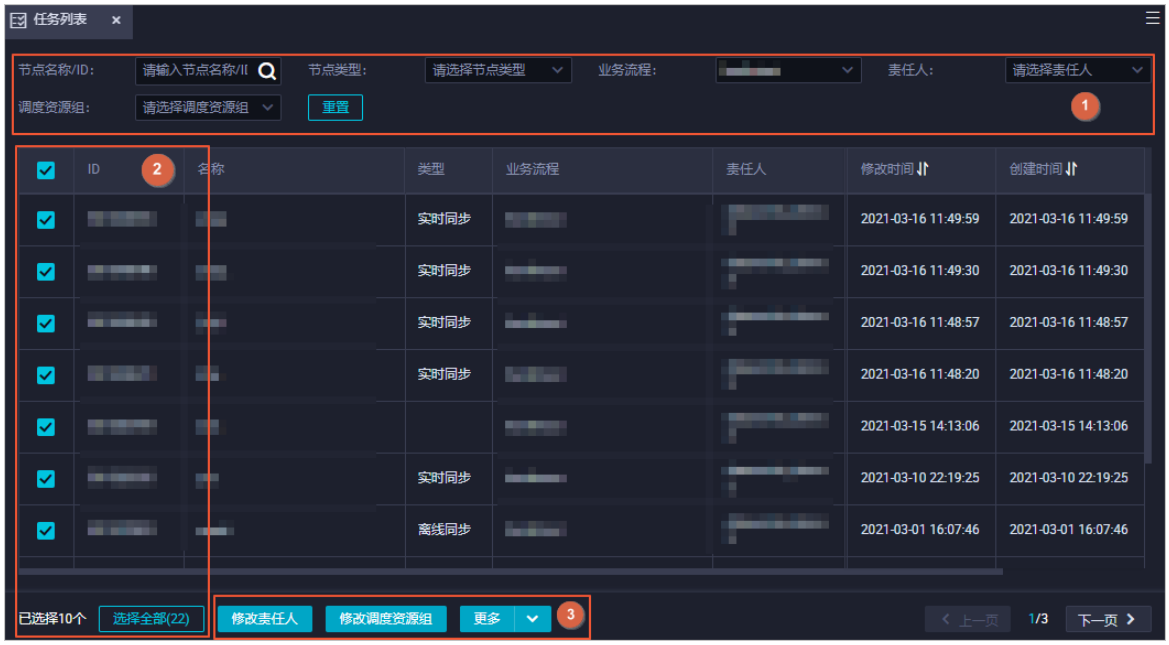
如果您需要批量修改或删除项目空间中某一类别的节点（例如，修改所有离线同步节点），则可以在**业务流程**的**任务列表**页面，使用**节点类型**、**业务流程**、**调度资源组**等条件进行筛选，批量处理目标节点。

> **说明** 当前仅支持批量修改目标节点的责任人及调度资源组。

1. 在**数据开发**页面，单击**业务流程**后的图标，进入**任务列表**页面。



2. 修改或删除目标节点。



i. 您可以根据**节点名称/ID**、**节点类型**、**业务流程**等条件，筛选相应类型的节点。

ii. 选中需要处理的部分或全部节点。

iii. 修改或删除目标节点。

- 修改目标节点：当前仅支持批量修改目标节点的责任人及调度资源组。单击**修改责任人**或**修改调度资源组**进行修改。

  当修改对话框中，**强制修改**参数配置为**是**时，您可以修改所有选中的节点，当该参数配置为**否**时，您只能修改自己锁定的节点，而不能修改他人锁定的节点。

- 删除目标节点：单击**更多 > 删除**，删除选中的节点。

  当**删除节点**对话框中，**强制删除**参数配置为**是**时，您可以删除所有选中的节点，当该参数配置为**否**时，您只能删除自己锁定的节点，而不能删除他人锁定的节点。

快速复制业务流程

您可以通过节点组的功能快速将某一个业务流程组成一个节点组，然后在新业务流程中引用该节点组，详情可参见节点组。

快速导入导出多个业务流程至其他DataWorks工作空间或其他开源引擎

如果您需要快速、批量导出DataWorks工作空间的多个业务流程，并将业务流程导入至其他DataWorks工作空间或其他开源引擎，则可以使用DataWorks的迁移助手功能，详情请参见概述。

4.3. 管理手动业务流程

手动业务流程中创建的所有节点都需要手动触发，无法通过调度执行。因此，手动业务流程中的节点无需配置父节点依赖与本节点的输出。

创建手动业务流程

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. 在左侧导航栏，单击**手动业务流程**。

   单击左下方的☰图标，即可展开或折叠左侧导航栏。
3. 右键单击**手动业务流程**，选择**新建业务流程**。
4. 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。
5. 单击**新建**。

界面功能点介绍

手动业务流程界面的功能说明如下表所示。

序号	功能	描述
1	**提交**	提交当前手动业务流程中的所有节点。
2	**运行**	运行当前手动业务流程下的所有节点，因为手动任务不存在依赖，所以会同时运行。
3	**停止运行**	停止正在运行的节点。
4	**发布**	跳转至**任务发布**页面，可以将当前所有只提交未发布的节点，选择部分或全部发布至生产环境。
5	**前往运维**	快速进入生产运维中心，查看节点的运维详情。
6	**切换布局**	您可以切换当前业务流程编辑面板的布局为**纵向**、**横向**或**网格**。

序号	功能	描述
7	**框选**	您可以框选需要的节点组成节点组。
8	**刷新**	刷新当前手动业务流程界面。
9	**格式化**	格式化当前手动业务流程界面。
10	**适配窗口**	根据当前界面的窗口大小，自动适配业务流程的布局。
11	**居中**	居中当前业务流程的各节点。
12	**1：1**	用于将当前业务流程的各节点与编辑面板按照1：1比例布局。
13	**放大**	放大界面。
14	**缩小**	缩小界面。
15	**查询**	查询当前手动业务流程下的某个节点。
16	**全屏**	全屏展示当前手动业务流程的节点。
17	**隐藏引擎信息**	用于显示或隐藏各节点的引擎信息。
18	**流程参数**	设置参数，流程参数优先级高于节点参数的优先级。如果参数key与参数对应，会优先执行手动业务流程设置的参数。
19	**操作历史**	对当前手动业务流程下所有节点的操作历史。
20	**版本**	当前手动业务流程下所有节点的提交发布记录。

手动业务流程的组成

> **说明** 建议单个手动业务流程下的节点总数不要超过100个。

手动业务流程由以下各模块的节点组成，您根据上文的操作新建手动业务流程后，再创建相应的节点。详情请参见业务流程：

- **数据集成**

  双击相应手动业务流程下的**数据集成**，即可查看所有的数据集成任务。

  右键单击**数据集成**，单击**新建 > 离线同步**即可新建离线同步节点，详情请参见离线同步。

- **MaxCompute**

  > **注意** 您在**工作空间配置**页面添加MaxCompute计算引擎实例后，当前页面才会显示MaxCompute目录。详情请参见配置工作空间。

  MaxCompute计算引擎包括ODPS SQL、SQL组件节点、ODPS Spark、PyODPS 2、ODPS Script、ODPS MR和PyODPS 2等数据开发节点，并可以查看和新建表、资源及函数：

- **数据开发**

  打开相应手动业务流程下的**MaxCompute**，右键单击**数据开发**，即可创建相关的数据开发节点。详情请参见创建ODPS SQL节点、创建SQL组件节点、创建ODPS Spark节点、创建PyODPS 2节点、创建ODPS Script节点、创建ODPS MR节点和创建PyODPS 3节点。

- **表**

  打开相应手动业务流程下的**MaxCompute**，右键单击**表**，即可进行新建。您也可以在此查看当前MaxCompute计算引擎下所有创建的表。详情请参见创建MaxCompute表。

- **资源**

  打开相应手动业务流程下的**MaxCompute**，右键单击**资源**，即可进行新建。您也可以在此查看当前MaxCompute计算引擎下所有创建的资源，详情请参见创建MaxCompute资源。

- **函数**

  打开相应手动业务流程下的**MaxCompute**，右键单击**函数**，即可进行新建。您也可以在此查看当前MaxCompute计算引擎下所有创建的函数，详情请参见注册MaxCompute函数。

- **EMR**

  > **注意** 您在**工作空间配置**页面添加E-MapReduce计算引擎实例后，当前页面才会显示EMR目录。详情请参见配置工作空间。

  EMR计算引擎包括EMR Hive、EMR MR、EMR Spark SQL、EMR Spark、EMR Shell、EMR Presto和EMR Impala等数据开发节点，并可以查看和新建EMR资源：

  - **数据开发**

    打开相应手动业务流程下的**EMR**，右键单击**数据开发**，即可创建相关的数据开发节点。详情请参见创建EMR Hive节点、创建并使用EMR MR节点、创建EMR Spark SQL节点、创建EMR Spark节点、创建EMR Presto节点和创建EMR Impala节点。

  - **资源**

    打开相应手动业务流程下的**EMR**，右键单击**资源**，即可进行新建。您也可以在此查看当前EMR计算引擎下所有创建的资源，详情请参见创建和使用EMR资源。

  - **函数**

    打开相应手动业务流程下的**EMR**，右键单击**函数**，即可进行新建。您也可以在此查看当前EMR计算引擎下所有创建的资源，详情请参见创建和使用EMR资源。

- **算法**

  打开相应的手动业务流程，右键单击**算法**，即可进行新建。您也可以查看当前手动业务流程下所有创建的机器学习节点。详情请参见创建机器学习（PAI）节点。

- **通用**

  打开相应的手动业务流程，右键单击**通用**，即可创建相关节点。详情请参见Shell节点和虚拟节点。

- **自定义**

  打开相应的手动业务流程，右键单击**自定义**，即可创建相关节点。详情请参见创建Data Lake Analytics节点和创建AnalyticDB for MySQL节点。

5.创建及管理表

5.1. 创建表

5.1.1. 创建MaxCompute表

本文为您介绍如何创建MaxCompute表。

前提条件

您在**工作空间配置**页面添加MaxCompute引擎后，当前页面才会显示MaxCompute目录。详情请参见配置工作空间。

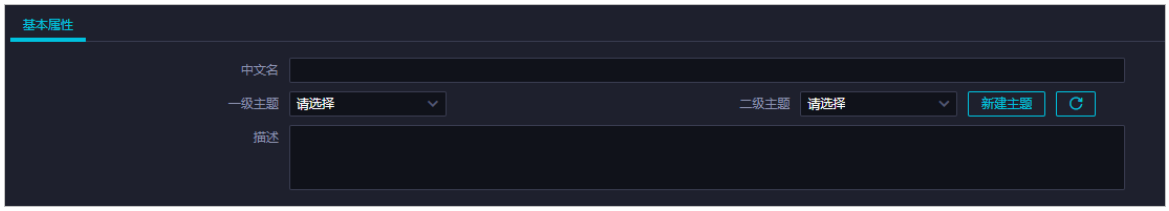
操作步骤

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**数据开发**。
2. 鼠标悬停至图标，单击**新建表 > MaxCompute > 表**。

   您也可以打开相应的业务流程，右键单击**MaxCompute**，选择**新建表**。
3. 在**新建表**对话框中，选择**路径**，输入**名称**，单击**新建**，进入表的编辑页面。

   > 注意
   > - 表名不能超过100个字符，且必须以字母开头，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**。

4. 在**基本属性**区域，配置各项参数。



名称	描述
中文名	MaxCompute表的中文名称。
一级主题	新建表所处的一级目标文件夹名称。 **说明** 一级、二级主题仅仅是DataWorks上文件夹的摆放形式，目的是为了您能更好地管理您的表。
二级主题	新建表所处的二级目标文件夹名称。

名称	描述
新建主题	单击**新建主题**，进入**主题管理**页面，您可以在该页面创建一级主题、二级主题。 新建主题后，单击图标，即可同步新建的主题。
描述	对新建表进行简单描述。

5. 创建表。

   您可以通过以下两种方式创建表：

   - 使用DDL模式创建表。

     单击工具栏中的**DDL模式**，在对话框中输入建表语句，单击**生成表结构**，即可自动填充**物理模型设计**、**表结构设计**中的相关内容。建表语句的详情请参见标准的建表语句。

   - 使用图形界面创建表。

     如果不适用于DDL模式建表，您也可以使用图形界面直接建表。

分类	参数	描述
物理模型设计	**分区类型**	包括**分区表**和**非分区表**。
	生命周期	MaxCompute的生命周期功能。如果选中生命周期，请在**选择生命周期（日）**中输入一个数字表示天数，该表（或分区）超过设置的天数，会清除未更新的数据。
	层级	通常分为ODS、CDM和ADS三个层级，您可以自定义层级名称。 关于物理层级的相关信息请参见数仓分层。
	物理分类	包括基础业务层、高级业务层和其它，您可以自定义分类名称。 单击**新建层级**，进入**表管理**页面，单击**层级管理**，即可新增表层级和表物理分类。 **说明** 物理分类仅为方便您的管理，不涉及底层实现。

分类	参数	描述
表结构设计	**添加字段**	单击**添加字段**，配置字段信息后，单击**保存**，即可新增一个字段。
	上移	调整未创建的表的字段顺序。如果为已经创建的表调整字段顺序，会要求删除当前已经创建的表，再新建一张同名表。生产环境中禁止该操作。
	下移	
	字段英文名	字段英文名，由字母、数字和下划线（_）组成。
	字段中文名	字段的中文名称。
	字段类型	MaxCompute数据类型，支持TINYINT、SMALLINT、INT、BIGINT、FLOAT、DOUBLE、DECIMAL、VARCHAR、CHAR、STRING、BINARY、DATETIME、DATE、TIMESTAMP、BOOLEAN、ARRAY、MAP和STRUCT。详情请参见数据类型。
	长度/设置	当选择的字段类型需要设置长度时，请在文本框中进行配置。
	描述	对字段进行描述。
	主键	勾选表示该字段是主键，或者是联合主键的其中一个字段。
	编辑	单击已保存字段后的**编辑**，修改当前字段的配置，并单击**保存**。
	删除	删除已经创建的字段。 **说明** 已经创建的表，删除字段重新提交时，会要求删除当前表，再去建一张同名表，在生产环境中禁止该操作。
分区字段设计 **说明** 当物理模型设计选择分区表后才显示分区字段设计。	**添加分区**	如果您在**物理模型设计**区域，设置**分区类型**为**分区表**，则需要配置分区。 您可以为当前表新建一个分区。如果为已经创建的表添加分区，会要求删除当前已经创建的表，再新建一张同名表。生产环境中禁止该操作。
	字段类型	建议统一采用STRING类型。
	日期分区格式	如果该分区字段是日期含义（尽管数据类型可能是STRING），则一个或自填一个日期格式，常用格式为*yyyymmdd*、*yyyy-mm-dd*。
	日期分区粒度	支持的分区粒度包括秒、分、时、日、月、季度和年。创建分区的粒度根据需要填写，如果需要填写多个分区粒度，则默认粒度越大，分区等级越高。例如，同时存在日、时、月三个分区，多级分区关系是一级分区（月），二级分区（日），三级分区（时）。

分类	参数	描述
	删除	可以删除一个分区。如果删除已创建的表的分区，会要求删除当前已经创建的表，再新建一张同名表。生产环境中禁止该操作。

6. 分别单击**提交到开发环境**和**提交到生产环境**。

   如果您使用的是简单模式的工作空间，仅需要单击**提交到生产环境**。

名称	描述
从开发环境加载	如果该表已经提交到开发环境，该按钮会高亮显示。单击后，开发环境已经创建的表信息会覆盖当前的页面信息。 **说明** 仅MaxCompute表支持该功能。
提交到开发环境	请确认当前编辑页面的必填项是否已经填写完整。如果有遗漏会禁止提交。
从生产环境加载	已经提交到生产环境的表的详细信息会覆盖当前页面。 **说明** 仅MaxCompute表支持该功能。
提交到生产环境	提交后，会在生产环境的工作空间创建该表。

后续步骤

新建表成功后，您可以进行查询、修改和删除等操作，详情请参见管理表。

5.1.2. 创建AnalyticDB for PostgreSQL表

本文为您介绍如何创建AnalyticDB for PostgreSQL表。

前提条件

- 您需要购买DataWorks标准版及以上版本，才可以绑定AnalyticDB for PostgreSQL计算引擎实例。
- 请根据业务需求新增独享调度资源组或自定义资源组，详情请参见购买资源组（创建订单）或新增和使用自定义调度资源组。

  绑定AnalyticDB for PostgreSQL引擎时，请选择新增的资源组并测试连通性通过，才可以创建AnalyticDB for PostgreSQL表。
- 您在**工作空间配置**页面绑定AnalyticDB for PostgreSQL引擎后，当前页面才会显示AnalyticDB for PostgreSQL目录。详情请参见配置工作空间。
- 绑定计算引擎后，您还需要在**数据地图**页面采集AnalyticDB for PostgreSQL元数据。详情请参见采集AnalyticDB for PostgreSQL元数据。

操作步骤

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，鼠标悬停至**+新建**图标，单击**AnalyticDB > 表**。

   您也可以打开相应的业务流程，右键单击**AnalyticDB for PostgreSQL**，选择**新建 > ADB可视化建表**。

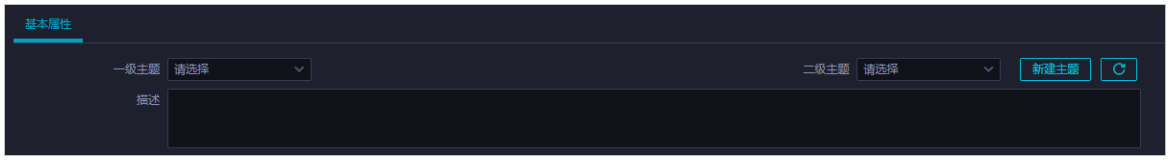
3. 在**新建表**对话框中，输入**表名**。

   > **注意**
   > - 表名的格式为schema_name.table_name。
   > - schema_name和table_name均以字母或下划线（_）开头，仅包含字母、数字或下划线（_），且长度不能超过63个字符。
   > - 如果绑定多个实例，则需要选择相应的引擎实例。

4. 单击**提交**，进入表的编辑页面。

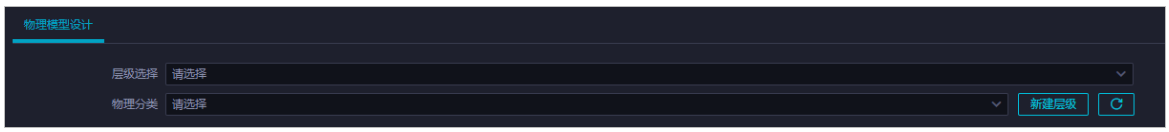
   该页面上方为您展示在**新建表**对话框中，配置的**表名**和**引擎实例**。

5. 在**基本属性**区域，配置各项参数。



名称	描述
一级主题	新建表所处的一级目标文件夹名称。 **说明** 一级、二级主题仅仅是DataWorks上文件夹的摆放形式，目的是为了您能更好地管理您的表。
二级主题	新建表所处的二级目标文件夹名称。
新建主题	单击**新建主题**，进入**主题管理**页面，您可以在该页面创建一级主题、二级主题。 新建主题后，单击图标。
描述	针对新建表的描述。

6. 在**物理模型设计**区域，配置各项参数。

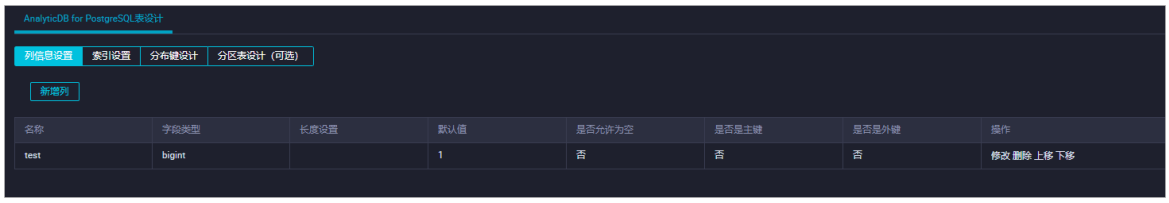


名称	描述

名称	描述
层级选择	通常分为ODS、CDM和ADS三个层级，您可以自定义层级名称。 关于物理层级的相关信息请参见数仓分层。
物理分类	包括基础业务层、高级业务层和其它，您可以自定义分类名称。 **说明** 物理分类仅为方便您的管理，不涉及底层实现。
新建层级	如果您需要新建层级和物理分类，请单击**新建层级**，在**层级管理**页面进行添加。 新建成功后，单击图标。

7. 在**AnalyticDB for PostgreSQL表设计**区域，配置各项参数。

   AnalyticDB for PostgreSQL表设计包括**列信息设置**、**索引设置**、**分布键设计**和**分区表设计（可选）**。



分类	名称	描述
列信息设置	**新增列**	单击后，请设置字段的相关信息。
	名称	输入字段的名称。
	字段类型	选择字段的类型。
	长度设置	仅部分字段类型可以自定义设置长度。
	默认值	输入字段的默认值。
	是否允许为空	设置该字段是否允许为空。
	是否是主键	设置该字段是否为主键。
	是否是外键	设置该字段是否为外键。
	操作	○ 对于新增的列，您可以进行**保存**、**取消**、**删除**、**上移**和**下移**等操作。 ○ 对于已有的列，您可以进行**修改**、**删除**、**上移**和**下移**等操作。
	新增列	单击后，请设置索引的相关信息。
	索引名称	输入索引的名称，请确保索引名称的唯一性。

分类	名称	描述
索引设置	包含列	单击**编辑**，在**至少选择一项索引**对话框，单击+，会显示之前已设置的列信息。 从**列信息**列表，选择要添加的列，单击**保存**。
	索引类型	包括**普通索引**、**主键索引**和**唯一索引**。
	索引方式	包括**树索引**、**位图索引**和**gist索引**。
	操作	○ 对于新增的索引，您可以进行**保存**、**取消**、**删除**、**上移**和**下移**等操作。 ○ 对于已有的索引，您可以进行**修改**、**删除**、**上移**和**下移**等操作。
分布键设计	包括**Hash分布（推荐）**、**复制表模式**和**随机分布（不推荐）**。	以**Hash分布（推荐）**为例，单击**新增列**，从**名称**列表，选择相应的列。该列的信息会自动显示，单击**保存**。 更多详情请参见该表格中的**列信息设置**。
分区表设计（可选）	分区表设计（可选）	您可以在该页签自行设计分区表，详情请参见表分区定义。

8. 分别单击**提交到开发环境**和**提交到生产环境**。

   如果您使用的是简单模式的工作空间，仅需要单击**提交到生产环境**。

9. 在**提交变更**对话框中，确认建表语句无误后，从**选择资源组**下拉列表中，选择需要的资源组，单击**确认执行**。

> 说明
> ○ 仅支持选择独享调度资源组或自定义调度资源组。
> ○ 此处选择的资源组需要和绑定计算引擎时通过测试连通性的资源组保持一致。

后续步骤

新建表成功后，您可以进行查询、修改和删除等操作，详情请参见管理表。

5.1.3. 创建EMR表

本文为您介绍如何创建EMR（E-MapReduce）表。

前提条件

-
-
- 您需要在**数据地图**模块采集EMR元数据后，才可以在新建表时选择到EMR库。详情请参见采集E-MapReduce元数据。

操作步骤

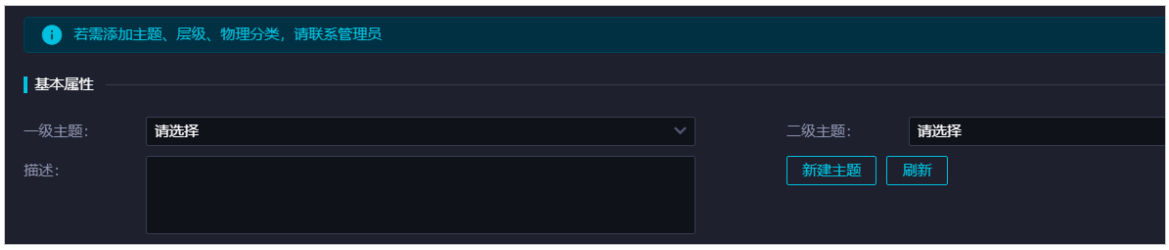
1. 进入**数据开发**页面。
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**工作空间列表**。
   iii. 选择工作空间所在地域后，单击相应工作空间后的**进入数据开发**。
2. 鼠标悬停至图标，单击**新建表** > **EMR** > **表**。

   您也可以找到相应的业务流程，右键单击**EMR**，单击**新建表**。
3. 在**新建表**对话框中，配置各项参数。

参数	描述
引擎类型	默认为EMR，且不可以修改。
引擎实例	从下拉列表中选择相应的引擎实例。
名称	待新建的EMR表名。

4. 单击**提交**，进入表的编辑页面。

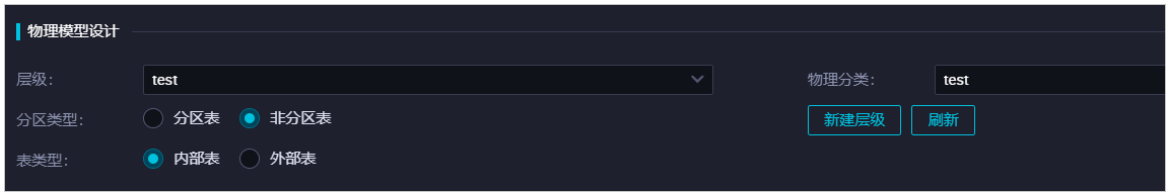
   该页面上方为您展示**新建表**对话框中的配置，您可以修改EMR引擎实例的**所属库**。如果您需要新建数据库，请单击**新建库**。在**新建库**对话框中，配置各项参数，单击**确认**。
5. 在**基本属性**区域，配置各项参数。



名称	描述
一级主题	新建表所处的一级目标文件夹名称。 **说明** 一级、二级主题仅仅是DataWorks上文件夹的摆放形式，目的是为了您能更好地管理您的表。
二级主题	新建表所处的二级目标文件夹名称。

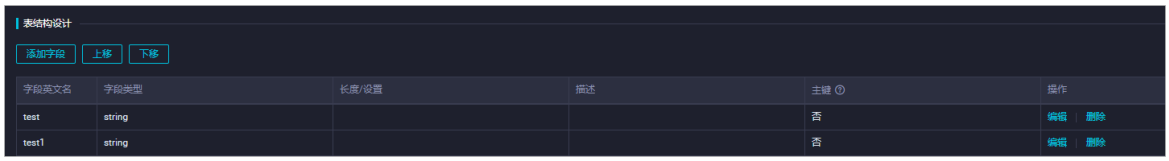
名称	描述
新建主题	单击**新建主题**，进入**主题管理**页面，您可以在该页面创建一级主题、二级主题。
刷新	新建主题后，单击**刷新**。
描述	对新建表进行简单描述。

6. 在**物理模型设计**区域，配置各项参数。



参数	描述
层级	从下拉列表中选择相应的层级和物理分类。如果您需要新建层级和物理分类，请联系工作空间管理员，单击**新建层级**，在**层级管理**页面进行添加。新建成功后，单击**刷新**。
物理分类	
分区类型	包括**分区表**和**非分区表**。
表类型	包括**内部表**和**外部表**。

7. 在**表结构设计**区域，配置各项参数。



参数	描述
添加字段	单击**添加字段**，配置字段信息后，单击**保存**，即可新增一个字段。
上移	调整未创建的表的字段顺序。如果为已经创建的表调整字段顺序，会要求删除当前已经创建的表，再新建一张同名表。生产环境中禁止该操作。
下移	
字段英文名	字段的英文名称，由字母、数字和下划线（_）组成。
字段类型	支持TINYINT、SMALLINT、INT、BIGINT、FLOAT、DOUBLE、DECIMAL、VARCHAR、CHAR、STRING、BINARY、DATETIME、DATE、TIMESTAMP、BOOLEAN、ARRAY、MAP和STRUCT。
长度/设置	当选择的字段类型需要设置长度时，请在文本框中进行配置。
描述	对字段进行描述。
主键	勾选表示该字段是主键。该主键为业务概念，您可以在业务上保证记录的唯一性，DataWorks对主键无约束。

参数	描述
编辑	单击已保存字段后的**编辑**，修改当前字段的配置，并单击**保存**。
删除	删除已经创建的字段。 **说明** 已经创建的表，删除字段重新提交时，会要求删除当前表，再去建一张同名表，在生产环境中禁止该操作。
添加分区	如果您在**物理模型设计**区域，设置**分区类型**为**分区表**，则需要配置分区。 您可以为当前表新建一个分区。如果为已经创建的表添加分区，会要求删除当前已经创建的表，再新建一张同名表。生产环境中禁止该操作。

8. 单击工具栏中的图标，提交EMR表至生产环境。

   如果您使用的是标准模式的工作空间，请先提交表至开发环境，再提交表至生产环境。

   注意

   提交时，您需要选择提交表时所用的调度资源组，当使用独享调度资源组提交表时，DataWorks平台将下发对应新建表的任务到引擎侧执行，并打印执行过程的执行日志，如果资源提交过程中出现问题，您可以先通过日志自助排查。如果您目前无可用的独享调度资源组，请购买并配置独享调度资源组便于使用，操作详情请参见新增和使用独享调度资源组。

5.2. 查看公共表

DataWorks支持查看MaxCompute、AnalyticDB for PostgreSQL、Hologres和EMR公共表（同一租户下的生产表），您可以在公共表页面选择引擎类型和引擎名进行查看。

前提条件

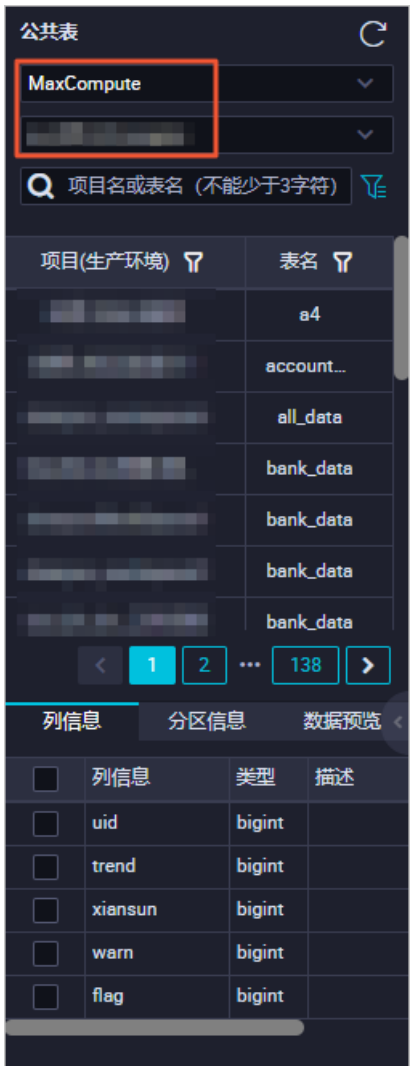
- 如果您需要查看MaxCompute公共表，请绑定MaxCompute计算引擎。详情请参见配置工作空间。
- 如果您需要查看AnalyticDB for PostgreSQL公共表，请绑定AnalyticDB for PostgreSQL计算引擎，并在**数据地图**页面采集AnalyticDB for PostgreSQL元数据。详情请参见配置工作空间和采集AnalyticDB for PostgreSQL元数据。
- 如果您需要查看Hologres公共表，请绑定Hologres计算引擎，并在**数据地图**页面采集Hologres元数据。详情请参见配置工作空间和采集Hologres元数据。
- 如果您需要查看EMR（E-MapReduce）公共表，请绑定EMR计算引擎，并在**数据地图**页面采集EMR元数据。详情请参见配置工作空间和采集E-MapReduce元数据。

操作步骤

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. 在左侧导航栏，单击**公共表**。

单击左下方的图标，即可展开或折叠左侧导航栏。

3. 选择引擎类型和引擎名称，查看相应的公共表。



本文以选择MaxCompute引擎为例，为您介绍各项参数。

参数	描述
项目	相应环境下的工作空间名称。 单击搜索框后的图标，选择需要查询的环境，即可切换至相应的环境中。 **说明** ◦ 标准模式工作空间下，公共表包括**开发环境**和**生产环境**。 简单模式工作空间下，公共表仅包括**生产环境**。 ◦ 蓝色表示当前环境。
表名	相应工作空间下表的名称。

参数	描述
列信息	查看当前表的字段数量、字段类型及描述。
分区信息	查看当前表的分区信息、分区数量，分区数最大为6万个（如果设置了生命周期，实际分区数以生命周期为主）。 **注意** 仅MaxCompute公共表显示分区信息。
数据预览	预览当前表数据。 **注意** • 仅MaxCompute公共表支持数据预览。 • 暂不支持外部表数据预览。

5.3. 外部表

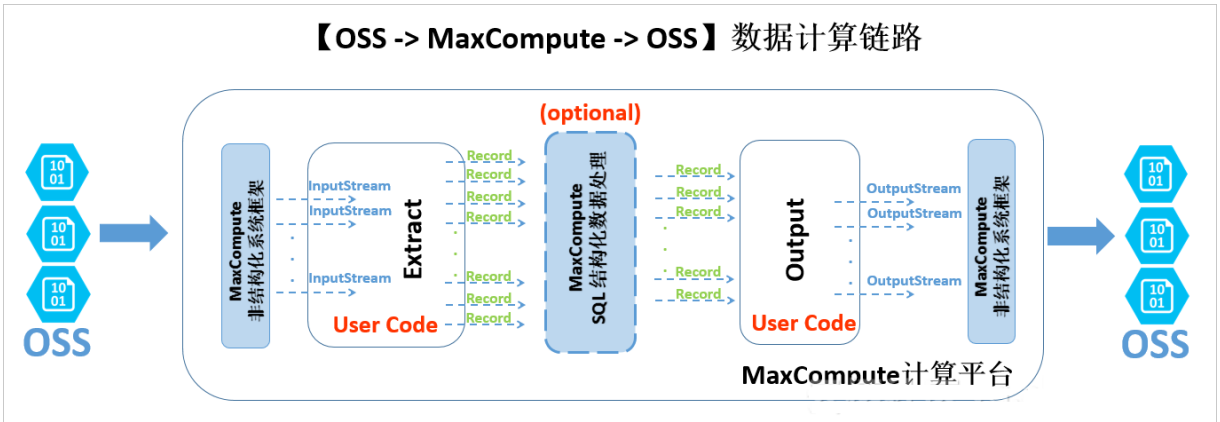
本文将为您介绍如何通过DataWorks创建、配置外部表，以及外部表支持的字段类型。

外部表概述

使用外部表前，您需要了解下表中的定义。

名称	描述
对象存储OSS	提供标准、低频、归档存储类型，能够覆盖不同的存储场景。同时，OSS能够与Hadoop开源社区及EMR、批量计算、MaxCompute、机器学习和函数计算等产品进行深度结合。
MaxCompute	大数据计算服务MaxCompute为您提供快速且完全托管的数据仓库解决方案，并可以通过与OSS的结合，高效经济地分析处理海量数据。
MaxCompute外部表	该功能基于MaxCompute新一代的V2.0计算框架，可以帮助您直接对OSS中的海量文件进行查询，无需将数据加载至MaxCompute表中。既减少了数据迁移的时间和人力，也节省了存储的成本。

下图为外部表的整体处理架构。



目前，MaxCompute主要支持OSS和OTS等非结构化存储的外部表。从数据的流动和处理逻辑的角度，非结构化处理框架在MaxCompute计算平台两端有耦合地进行数据导入和导出。以OSS外部表为例，处理逻辑如下：

1. 外部的OSS数据经过非结构化框架转换，使用JAVA InputStream类提供给您自定义代码接口。您可以自己实现Extract逻辑，只需要负责对输入的InputStream进行读取、解析、转化和计算，最终返回MaxCompute计算平台通用的Record格式。
2. 上述Record可以自由参与MaxCompute的SQL逻辑运算，该部分计算基于MaxCompute内置的结构化SQL运算引擎，并可能产生新的Record。
3. 经过运算的Record传递给用户自定义的Output逻辑，您可以进行进一步的计算转换，并最终通过系统提供的OutputStream，输出Record中需要输出的信息，由系统负责写入至OSS。

您可以通过DataWorks配合MaxCompute，对外部表进行可视化的创建、搜索、查询、配置、加工和分析等操作。

网络与权限认证

由于MaxCompute与OSS是两个独立的云计算与云存储服务，所以在不同的部署集群上的网络连通性有可能影响MaxCompute访问OSS的数据的可达性。在MaxCompute上访问OSS存储时，建议您使用OSS私网地址（即以-internal.aliyuncs.com结尾的host地址）。

MaxCompute需要有一个安全的授权通道访问OSS数据。MaxCompute结合了阿里云的访问控制服务（RAM）和令牌服务（STS）实现对数据的安全访问。MaxCompute在获取权限时，以表的创建者的身份在STS申请权限（OTS的权限设置与OSS一致）。

1. STS模式授权

   MaxCompute需要直接访问OSS的数据，因此需要将OSS数据相关权限赋给MaxCompute的访问账号。STS是阿里云为客户提供的一种安全令牌管理服务，它是资源访问管理（RAM）产品中的一员。通过STS服务，获得许可的云服务或RAM用户，可以自主颁发自定义时效和子权限的访问令牌。获得访问令牌的应用程序，可以使用令牌直接调用阿里云服务API操作资源。

   详情请参见OSS的STS模式授权。

   您可以通过以下两种方式进行授权：

   - 当MaxCompute和OSS的项目所有者是同一个账号时，请直接登录阿里云账号后进行一键授权。
     a. 打开新建表的编辑页面，找到**物理模型设计**模块。
     b. 勾选**表类型**后的**外部表**。

c. 单击**选择存储地址**后的**一键授权**。

d. 单击**云资源访问授权**对话框中的**同意授权**。

- 自定义授权，在RAM中授予MaxCompute访问OSS的权限。

a. 登录RAM控制台。

> **说明** 如果MaxCompute和OSS不是同一个账号，此处需要由OSS账号登录并授权。

b. 单击左侧导航栏中的**RAM角色管理**

c.

d. 输入**角色名称**和**备注**。

> **说明** 设置角色名称为AliyunODPSDefaultRole或AliyunODPSRoleForOtherUser。

e. **选择云账号**为**当前云账号**或**其他云账号**。

> **说明** 如果选择**其他云账号**，请输入其他云账号的ID。

f.

g. 配置**角色详情**。

在**RAM角色管理**页面，单击相应的**RAM角色名称**。在**信任策略管理**页签下，单击**修改信任策略**，根据自身情况输入下述策略内容。

```
--当MaxCompute和OSS的Owner是同一个账号。
{
"Statement": [
{
"Action": "sts:AssumeRole",
"Effect": "Allow",
"Principal": {
"Service": [
"odps.aliyuncs.com"
      ]
    }
  }
],
"Version": "1"
}
```

```
--当MaxCompute和OSS的Owner不是同一个账号。
{
"Statement": [
{
"Action": "sts:AssumeRole",
"Effect": "Allow",
"Principal": {
"Service": [
"MaxCompute的Owner云账号id@odps.aliyuncs.com"
      ]
    }
  }
],
"Version": "1"
}
```

配置完成后，单击**确定**。

h. 配置**角色授权策略**，并找到授予角色访问OSS必要的权限AliyunODPSRolePolicy，将权限AliyunODPSRolePolicy授权给该角色。如果您无法通过**搜索授权**找到，可以通过**精确授权**直接添加。

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "oss:ListBuckets",
        "oss:GetObject",
        "oss:ListObjects",
        "oss:PutObject",
        "oss:DeleteObject",
        "oss:AbortMultipartUpload",
        "oss:ListParts"
        ],
        "Resource": "*",
        "Effect": "Allow"
  },
  {
      "Action": [
        "ots:ListTable",
        "ots:DescribeTable",
        "ots:GetRow",
        "ots:PutRow",
        "ots:UpdateRow",
        "ots:DeleteRow",
        "ots:GetRange",
        "ots:BatchGetRow",
        "ots:BatchWriteRow",
        "ots:ComputeSplitPointsBySize"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

2. 使用OSS数据源

   如果您已创建并保存了OSS数据源，请进入**工作空间管理 > 数据源管理**页面进行查看和使用。

创建外部表

1. DDL模式建表

   进入**数据开发**页面，参见创建MaxCompute表进行DDL模式建表，您只需要遵守正常的MaxCompute语法即可。如果您的STS服务已成功授权，则无需设置odps.properties.rolearn属性。

   DDL建表语句示例如下，其中EXTERNAL参数说明该表为外部表。

```
CREATE EXTERNAL TABLE IF NOT EXISTS ambulance_data_csv_external(
vehicleId int,
recordId int,
patientId int,
calls int,
locationLatitute double,
locationLongtitue double,
recordTime string,
direction string
)
STORED BY 'com.aliyun.odps.udf.example.text.TextStorageHandler' --STORED BY用于指定自定义格式StorageHandler的类名或其它外部表文件格式，必选。
with SERDEPROPERTIES (
'delimiter'='\\|', --SERDEPROPERTIES序列化属性参数，可以通过DataAttributes传递到Extractor代码中，可选。
'odps.properties.rolearn'='acs:ram::xxxxxxxxxxxx:role/aliyunodpsdefaultrole'
)
LOCATION 'oss://oss-cn-shanghai-internal.aliyuncs.com/oss-odps-test/Demo/SampleData/CustomTxt/AmbulanceData/'     --外部表存放地址，必选。
USING 'odps-udf-example.jar'; --指定自定义格式时类定义所在的Jar包，如果未使用自定义格式无需指定。
```

关于STORED BY后接参数，其中CSV或TSV文件对应默认内置的StorageHandler，具体参数如下：

- CSV为 `com.aliyun.odps.CsvStorageHandler` ，定义如何读写CSV格式数据，数据格式约定列分隔符为英文逗号（,）、换行符为（\n）。实际参数输入示例： `STORED BY'com.aliyun.odps.CsvStorageHandler'` 。
- TSV为 `com.aliyun.odps.TsvStorageHandler` ，定义如何读写TSV格式数据，数据格式约定列分隔符为（\t）、换行符为（\n）。

STORED BY后接参数还支持ORC、PARQUET、SEQUENCEFILE、RCFILE、AVRO和TEXTFILE **开源格式外部表**，如下所示。对于textFile可以指定序列化类，例如 `org.apache.hive.hcatalog.data.JsonSerDe` 。

- org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe -> stored as textfile
- org.apache.hadoop.hive.ql.io.orc.OrcSerde -> stored as orc
- org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe -> stored as parquet
- org.apache.hadoop.hive.serde2.avro.AvroSerDe -> stored as avro
- org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe -> stored as sequencefile

对于开源格式外部表，建表语句如下。

```
CREATE EXTERNAL TABLE [IF NOT EXISTS] (<column schemas>)
[PARTITIONED BY (partition column schemas)]
[ROW FORMAT SERDE '']
STORED AS
[WITH SERDEPROPERTIES ( 'odps.properties.rolearn'='${roleran}'
[,'name2'='value2',...]
) ]
LOCATION 'oss://${endpoint}/${bucket}/${userfilePath}/';
```

SERDEPROPERTIES序列化属性列表如下所示。

属性名	属性值	默认值	描述
odps.text.option.gzip.input.enabled	true/false	false	打开或关闭读压缩
odps.text.option.gzip.output.enabled	true/false	false	打开或关闭写压缩
odps.text.option.header.lines.count	非负整数	0	跳过文本文件头N行
odps.text.option.null.indicator	字符串	空字符串	在解析或者写出NULL值时，代表NULL的字符串
odps.text.option.ignore.empty.lines	true/false	true	是否忽略空行
odps.text.option.encoding	UTF-8/UTF-16/US-ASCII	UTF-8	指定文本的字符编码

> **说明** MaxCompute目前仅支持通过内置extractor读取OSS上gzip压缩的CSV或TSV数据，您可以选择文件是否是gzip压缩，不同的文件格式对应不同的属性设置。

LOCATION参数，格式为：*oss://oss-cn-shanghai-internal.aliyuncs.com/Bucket名称/目录名称*。您可以通过图形对话框选择获得OSS目录地址，目录后无需加文件名称。

DDL模式创建的表会出现在表管理的表节点树下，可以通过修改其一级、二级主题来调整显示位置。

2. OTS外部表

   OTS外部表建表语句如下。

```
CREATE EXTERNAL TABLE IF NOT EXISTS ots_table_external(
odps_orderkey bigint,
odps_orderdate string,
odps_custkey bigint,
odps_orderstatus string,
odps_totalprice double
)
STORED BY 'com.aliyun.odps.TableStoreStorageHandler'
WITH SERDEPROPERTIES (
'tablestore.columns.mapping'=':o_orderkey,:o_orderdate,o_custkey, o_orderstatus,o_total
price', -- (3)
'tablestore.table.name'='ots_tpch_orders'
'odps.properties.rolearn'='acs:ram::xxxxx:role/aliyunodpsdefaultrole'
)
LOCATION 'tablestore://odps-ots-dev.cn-shanghai.ots-internal.aliyuncs.com';
```

   参数说明如下：

   - com.aliyun.odps.TableStoreStorageHandler是MaxCompute内置的处理TableStore数据的StorageHandler。
   - SERDEPROPERTIES是提供参数选项的接口，在使用TableStoreStorageHandler时，有两个必须指定的选项：tablestore.columns.mapping和 tablestore.table.name。

- tablestore.columns.mapping：必选项，用来描述MaxCompute将访问的Table Store表的列，包括主键和属性以（:）打头的用来表示Table Store主键，例如此语句中的 `:o_orderkey` 和 `:o_orderdate` ，其它均为属性列。

  Table Store支持1~4个主键，主键类型为STRING、INTEGER和BINARY，其中第一个主键为分区键。指定映射时，您必须提供指定Table Store表的所有主键，对于属性列则没有必要全部提供，可以只提供需要通过MaxCompute来访问的属性列。

- tablestore.table.name：需要访问的Table Store表名。如果指定的Table Store表名错误（不存在），则会报错， MaxCompute不会主动去创建Table Store表。

- LOCATION：用来指定Table Storeinstance名字、endpoint等具体信息。

3. 图形化建表

   进入**数据开发**页面，参见创建MaxCompute表进行图形化建表。外部表具有如下属性：

   - **基本属性**
     - 英文表名（在**新建表**时输入）
     - 中文表名
     - 一级、二级主题
     - 描述
   - **物理模型设计**
     - **表类型**：请选择为**外部表**。
     - **分区类型**：OTS类型外部表不支持分区。
     - **选择存储地址**：即LOCATION参数。您可以在**物理模型设计**栏中设置LOCATION参数。单击**点击选择**，即可选择存储地址。选择完成后，单击**一键授权**。
     - **选择存储格式**：根据业务需求进行选择，支持CSV、TSV、ORC、PARQUET、SEQUENCEFILE、RCFILE、AVRO、TEXTFILE和自定义文件格式。如果您选择了自定义文件格式，需要选择自定义的资源。在提交资源时，可以自动解析出其包含的类名并可以供用户选取。
     - rolearn：如果STS已授权，可以不填写。
   - **表结构设计**

参数	描述
字段类型	MaxCompute 2.0支持TINYINT、SMALLINT、INT、BIGINT、VARCHAR和STRING类型。

参数	描述
操作	支持新增、修改和删除。
长度/设置	对于VARCHAR类型，可以支持设置长度。对于复杂类型可以直接填写复杂类型的定义。

支持的字段类型

外部表支持的简单字段类型如下表所示。

类型	是否新增	格式举例	描述
TINYINT	是	1Y，-127Y	8位有符号整型，范围为-128~127。
SMALLINT	是	32767S， -100S	16位有符号整型，范围为-32,768~32,767。
INT	是	1000，-15645787	32位有符号整型，范围为-231~231-1。
BIGINT	否	100000000000L， -1L	64位有符号整型，范围为-263+1~263-1。
FLOAT	是	无	32位二进制浮点型。
DOUBLE	否	3.1415926 1E+7	8字节双精度浮点数，64位二进制浮点型。
DECIMAL	否	3.5BD，99999999999.99999999BD	10进制精确数字类型，整型部分范围为-1,036+1~1,036-1，小数部分精确到10~18。
VARCHAR(n)	是	无	变长字符类型，n为长度，取值范围为1~65,535。
STRING	否	“abc”，’bcd’，” alibaba”	字符串类型，目前长度限制为8MB。
BINARY	是	无	二进制数据类型，目前长度限制为8MB。
DATETIME	否	DATETIME ‘2017-11-11 00:00:00’	日期时间类型，使用东八区时间作为系统标准时间。范围0000年1月1日~9999年12月31日，精确到毫秒。
TIMESTAMP	是	TIMESTAMP ‘2017-11-11 00:00:00.123456789’	与时区无关的时间戳类型，范围为0000年1月1日~9999年12月31日23.59:59.999,999,999，精确到纳秒。
BOOLEAN	否	包括TRUE和FALSE	BOOLEAN类型，取值TRUE或FALSE。

外部表支持的复杂字段类型如下表所示。

类型	定义方法	构造方法
ARRAY	array< int >; array< struct< a:int, b:string >>	array(1, 2, 3); array(array(1, 2); array(3, 4))

类型	定义方法	构造方法
MAP	map< string, string >; map< smallint, array< string>>	map(“k1” , “v1” , “k2” , “v2”); map(1S, array(‘a’, ‘b’), 2S, array(‘x’, ‘y’))
STRUCT	struct< x:int, y:int>; struct< field1:bigint, field2:array< int>, field3:map< int, int>>	named_struct(‘x’, 1, ‘y’, 2); named_struct(‘field1’, 100L, ‘field2’, array(1, 2), ‘field3’, map(1, 100, 2, 200))

如果需要使用MaxCompute 2.0支持的新数据类型（TINYINT、SMALLINT、 INT、 FLOAT、VARCHAR、TIMESTAMP 、BINARY或复杂类型），需要在建表语句前加上语句 `set odps.sql.type.system.odps2=true;` ，set语句和建表语句一起提交执行。如果需要兼容HIVE，建议加上语句 `odps.sql.hive.compatible=true;` 。

查看和处理外部表

您可以在**数据开发**页面，单击左侧导航栏中的**表管理**，查询外部表，详情请参见管理表。处理外部表的方式与内部表基本一致。

5.4. 管理表

DataWorks支持查看、修改和删除MaxCompute、AnalyticDB for PostgreSQL、AnalyticDB for Mysql、CDH和EMR（E-MapReduce）表，您可以在表管理页面选择相应的引擎类型，进行查看和操作。

前提条件

- 如果您需要管理MaxCompute表，请绑定MaxCompute计算引擎。详情请参见配置工作空间。
- 如果您需要管理AnalyticDB for PostgreSQL表，请绑定AnalyticDB for PostgreSQL计算引擎，并在**数据地图**页面采集AnalyticDB for PostgreSQL元数据。详情请参见配置工作空间和采集AnalyticDB for PostgreSQL元数据。
- 如果您需要管理AnalyticDB for Mysql表，请绑定AnalyticDB for Mysql计算引擎，并在**数据地图**页面采集AnalyticDB for Mysql元数据。详情请参见配置工作空间和采集AnalyticDB for MySQL 3.0元数据。
- 如果您需要管理CDH表，请绑定CDH计算引擎，并在**数据地图**页面采集CDH数据。详情请参见配置工作空间和。

管理表

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. 在左侧导航栏，单击**表管理**。

   单击左下方的☰图标，即可展开或折叠左侧导航栏。
3. 在下拉列表中选择相应的引擎类型，查看和操作该类型的表。

本文以MaxCompute计算引擎为例，为您介绍如何查看、修改和删除表。新建表的详情请参见创建MaxCompute表。

操作	描述
查看表	单击搜索框后的图标，选择需要查询的环境，即可切换至相应的环境中。 **说明** ○ 标准模式工作空间下，公共表包括**开发环境**和**生产环境**。 简单模式工作空间下，公共表仅包括**生产环境**。 ○ 蓝色表示当前环境。 双击打开相应的表，即可在表的编辑页面查看详情。
重命名表	标准模式工作空间下，您可以右键重命名相应的开发表：选中相应的表名，选择**重命名表**。在**重命名表**对话框中，输入修改后的表名，单击**确认**。 **说明** 生产表不支持直接右键重命名，简单模式工作空间表管理功能不支持此操作。
导入数据	右键单击相应的表名，选择**导入数据**，具体配置请参见上传本地数据。

操作	描述
删除表	您可以删除开发环境和生产环境的表： ○ 删除开发环境的表： a. 拥有开发权限的用户右键单击相应的表名，选择**删除表**。 b. 在**删除表**对话框，选中**我已悉知风险，确认删除**。 c. 单击**确认**。 **说明** 生产表不支持直接右键删除表，简单模式工作空间表管理功能不支持此操作。 以tmp_pyodps开头的表为PyODPS类型，详情请参见创建PyODPS 2节点。 ○ 删除生产环境的表： a. 方式一：新建一个ODPS SQL节点，输入删除表命令，并且提交发布至生产环境执行一次，详情请参见创建ODPS SQL节点。 b. 方式二：在数据地图我的数据中删除。 

数仓分层

表的编辑页面中的**物理模型设计**，用于为您构建数仓分层。让您在管理数据时，可以对数据有更加清晰的规划和掌控。

常见的数仓分层如下：

- 数据引入层ODS（Operation Data Store）：存放未经过处理的原始数据至数据仓库系统，结构上与源系统保持一致，是数据仓库的数据准备区。主要完成基础数据引入到MaxCompute的职责，同时记录基础数据的历史变化。
- 数据公共层CDM（Common Data Model，又称通用数据模型层），包括DIM维度表、DWD和DWS，由ODS层数据加工而成。主要完成数据加工与整合，建立一致性的维度，构建可复用的面向分析和统计的明细事实表，以及汇总公共粒度的指标。
  - 公共维度层（DIM）：基于维度建模理念思想，建立整个企业的一致性维度。降低数据计算口径和算法不统一风险。

    公共维度层的表通常也被称为逻辑维度表，维度和维度逻辑表通常一一对应。
  - 公共汇总粒度事实层（DWS）：以分析的主题对象作为建模驱动，基于上层的应用和产品的指标需求，构建公共粒度的汇总指标事实表，以宽表化手段物理化模型。构建命名规范、口径一致的统计指标，为上层提供公共指标，建立汇总宽表、明细事实表。

    公共汇总粒度事实层的表通常也被称为汇总逻辑表，用于存放派生指标数据。

- 明细粒度事实层（DWD）：以业务过程作为建模驱动，基于每个具体的业务过程特点，构建最细粒度的明细层事实表。可以结合企业的数据使用特点，将明细事实表的某些重要维度属性字段做适当冗余，即宽表化处理。

  明细粒度事实层的表通常也被称为逻辑事实表。

- 数据应用层ADS（Application Data Service）：存放数据产品个性化的统计指标数据。根据CDM与ODS层加工生成。

您也可以根据业务需求创建其他分层数据层，创建数据分层的操作请参见创建数据分层以默认的五层数据分层为例，数据分层规划完成后，后续的表数据存储可根据规划分别存储至不同的数据分层中。。

6.创建及管理节点

6.1. 离线同步节点

离线同步节点支持MaxCompute、MySQL、DRDS、SQL Server、PostgreSQL、Oracle、MongoDB、DB2、OTS、OSS、FTP、HBase、LogHub、HDFS和Stream等数据源类型。本文为您介绍如何创建离线同步节点。

背景信息

当您输入表名时，页面会自动显示匹配表名的对象列表（当前仅支持精确匹配，请输入完整的正确的表名）。当前同步中心不支持的对象，会被打上**不支持**标签。

您可以将鼠标移动至列表对象上，页面会自动展示对象的详细信息，例如表所在的库、IP、Owner等，帮助您选择正确的表对象。选中后单击对象，列信息会自动填充。您也可以对列进行移动、删除、添加等操作。

离线同步支持的数据源请参见支持的数据源与读写插件。

操作步骤

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. 鼠标悬停至+新建图标，单击**数据集成 > 离线同步**。

   您也可以找到相应的业务流程，右键单击**数据集成**，选择**新建 > 离线同步**。
3. 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。

4. 单击**提交**。
5. 配置离线同步节点，详情请参见通过向导模式配置离线同步任务。
6. 提交节点。

   > **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   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，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。
7. 测试节点，详情请参见查看并管理周期任务。

6.2. 实时同步节点

DataWorks支持实时同步数据，本文为您介绍如何创建、编辑、提交和运维实时同步节点。

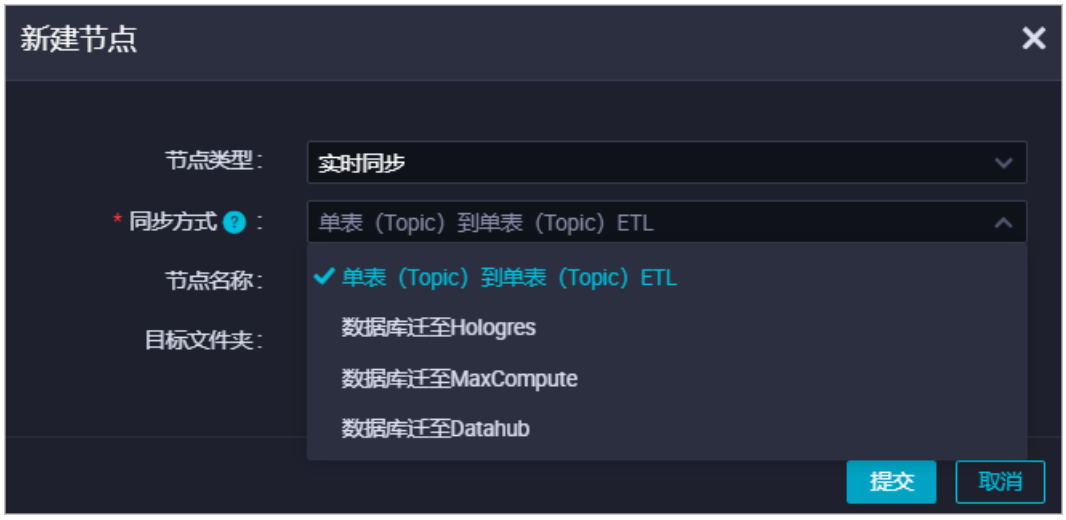
前提条件

目前实时同步处于公测阶段，支持的地域包括华东1（杭州）、华东2（上海）、华北2（北京）、华北3（张家口）、华南1（深圳）和西南1（成都）。

创建实时同步节点

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**进入数据开发**。
4. 鼠标悬停至+新建图标，单击**数据集成 > 实时同步**。

   您也可以找到相应的业务流程，右键单击**数据集成**，选择**新建 > 实时同步**。实时同步支持的数据源请参见实时同步支持的数据源。
5. 在**新建节点**对话框中，配置各项参数。



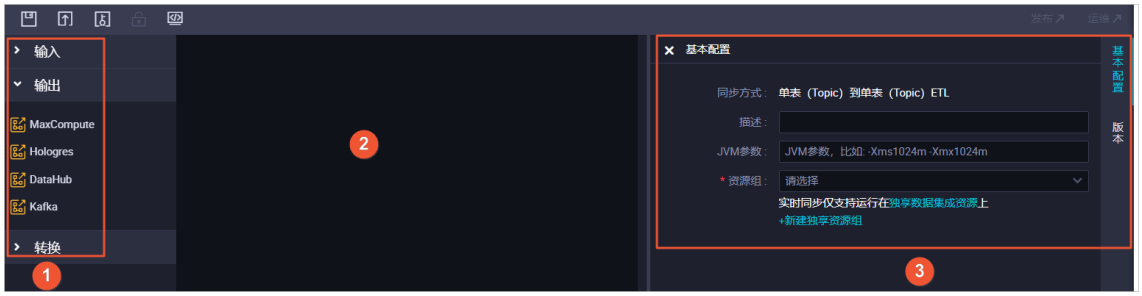
参数	描述
节点类型	默认为**实时同步**。
同步方式	包括**单表（Topic）到单表（Topic）ETL**、**数据库迁至Hologres**、**数据库迁至MaxCompute**和**数据库迁至DataHub**： ○ **单表（Topic）到单表（Topic）ETL**：实时同步单个表至一个或多个表中，支持同步过程中变换数据。 ○ **数据库迁至Hologres**：迁移一个整库下的所有或部分表至Hologres中，支持Hologres下自动创建目标表。 ○ **数据库迁至MaxCompute**：迁移一个整库下的所有或部分表至MaxCompute中。 ○ **数据库迁至DataHub**：迁移一个整库下的所有或部分Topic至DataHub中。
节点名称	节点名称必须是大小写字母、中文、数字、下划线（_）以及英文句号（.），且不能超过128个字符。
目标文件夹	存放节点的目录。

6. 单击**提交**。

编辑实时同步节点

选择不同的**同步方式**，实时同步节点的编辑页面也不同：

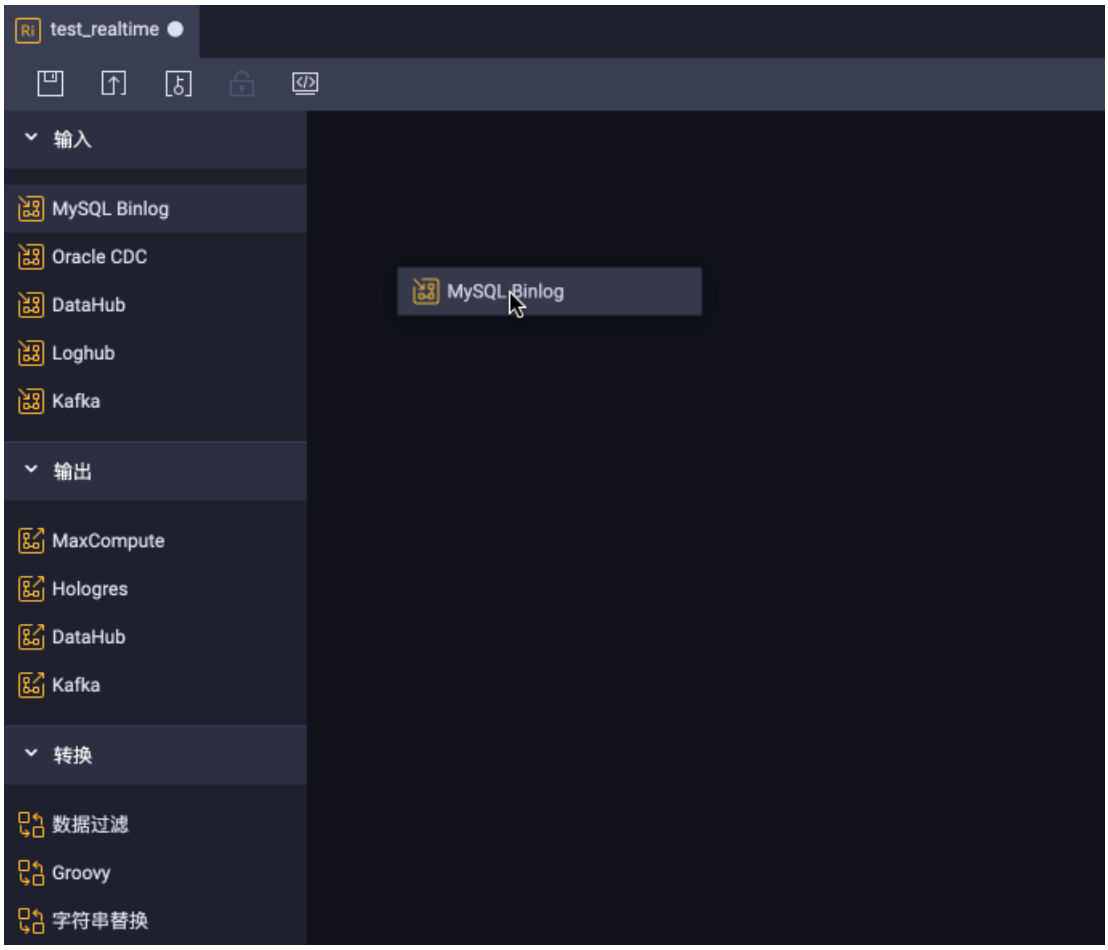
- 当选择**同步方式**为**单表（Topic）到单表（Topic）ETL**时，操作如下：
  i. 双击打开实时同步节点的编辑页面，单击右侧的**基本配置**，从**资源组**下拉列表中选择需要使用的资源组。



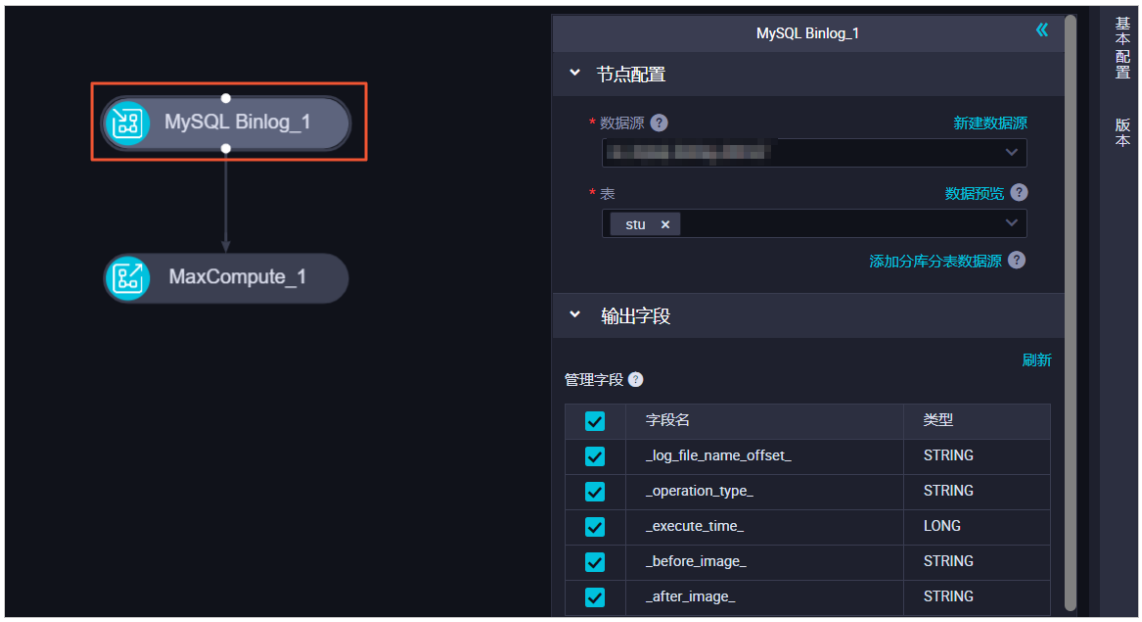
序号	描述
①	组件区域，包括**输入**、**输出**和**转换**三大模块。
②	节点的图形化编辑区域，您可以拖拽组件至该区域进行编辑。
③	属性配置区域，单击组件或右侧的**基本配置**时，会显示相应的属性配置面板。 **注意** 请务必选择**资源组**，否则提交节点时会报错。实时同步仅支持运行在独享数据集成资源组上，详情请参见新增和使用独享数据集成资源组。

  ii. 根据自身需求，从组件区域拖拽相应的组件至节点的编辑区域，并通过连线完成相应的节点关系连接，数据会根据连线从上游同步至下游。

下图为您展示新建MySQL数据实时同步至MaxCompute的过程。



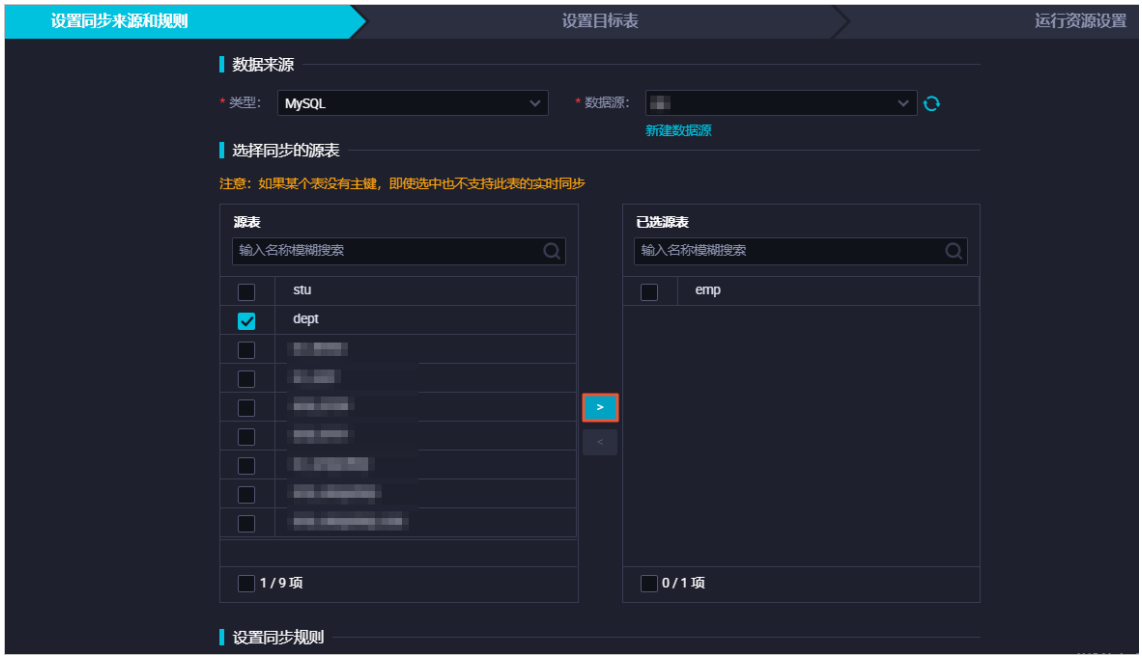
iii. 单击相应的节点，在**节点配置**对话框中，配置各项参数。详情请参见实时同步。



iv. 单击工具栏中的图标。

- 当选择**同步方式**为**数据库迁至Hologres**时，操作如下：

i. 双击打开实时同步节点的编辑页面，单击右侧的**基本配置**，从**资源组**下拉列表中选择需要使用的资源组。



> **注意** 请务必选择**资源组**，否则提交节点时会报错。实时同步仅支持运行在独享数据集成资源组上，详情请参见新增和使用独享数据集成资源组。

ii. 在**数据来源**区域，选择**类型**和**数据源**。

iii. 在**选择同步的源表**区域，选中需要同步的**源表**，单击 > 图标，将其移动至**已选源表**。

该区域会为您展示所选数据源下所有的表，您可以选择整库全表和部分表进行同步。

> **注意** 如果选中的表没有主键，将无法进行实时同步。

iv. （可选）在**设置同步规则**区域，单击**添加规则**，选择相应的规则进行添加。

同步规则包括**表名转换规则**和**目标表名规则**：

- **表名转换规则**：转换表名为目标表名，进行字符串替换。
- **目标表名规则**：支持对转换后的表名添加前缀和后缀。

v. 单击**下一步**。

vi. 在**设置目标表**页面，选择**目标Hologres数据源**和该数据源下的Schema。

vii. 单击**刷新源表和Hologres表映射**，创建需要同步的源表和目标Hologres表的映射关系。

viii. 查看任务的执行进度和表来源，单击**下一步**。

序号	描述
①	显示映射关系的创建进度。 **说明** 如果同步的表数量较多，会导致执行进度较慢，请耐心等待。
②	包括**自动建表**和**使用已有表**。 **说明** 暂不支持同步没有主键的表。但只要选择的表中包括有主键的表，会正常执行流程，没有主键的表会被忽略。
③	选择的**表建立方式**不同，此处显示的Hologres表名也不同： ■ 当**表建立方式**选择**自动建表**时，单击**下一步**，会显示**自动建表**对话框。请单击**开始建表**，创建成功后，单击**完成**。您可以单击表名称，查看和修改建表语句。 ■ 当**表建立方式**选择**使用已有表**时，请在下拉列表中选择需要的表。

ix. 在**运行资源设置**页面，配置**来源端读取支持最大连接数**和**目标端写入并发数**，并单击工具栏中的图标。

- 当选择**同步方式**为**数据库迁至MaxCompute**时，操作如下：
  i. 双击打开实时同步节点的编辑页面，单击右侧的**基本配置**，从**资源组**下拉列表中选择需要使用的资源组。
  ii. 在**数据来源**区域，选择**类型**和**数据源**。
  iii. 在**选择同步的源表**区域，选中需要同步的**源表**，单击图标，将其移动至**已选源表**。

  该区域会为您展示所选数据源下所有的表，您可以选择整库全表和部分表进行同步。

  注意 如果选中的表没有主键，将无法进行实时同步。

  iv. 在**设置同步规则**区域，单击**添加规则**，选择相应的规则进行添加。

  同步规则包括**表名转换规则**和**目标表名规则**：
  ■ **表名转换规则**：转换表名为目标表名，进行字符串替换。
  ■ **目标表名规则**：支持对转换后的表名添加前缀和后缀。
  v. 单击**下一步**。

vi. 在**设置目标表**页面，选择**目标MaxCompute（ODPS）数据源**，单击**MaxCompute（ODPS）时间自动分区设置**后的图标，在**编辑**对话框中，修改目标MaxCompute分区的设置（支持天和小时级别的分区）。

vii. 单击**刷新源表和MaxCompute（ODPS）表映射**，创建需要同步的源表和目标MaxCompute表的映射关系。

viii. 查看任务的执行进度和表来源，单击**下一步**。

序号	描述
①	显示映射关系的创建进度。 **说明** 如果同步的表数量较多，会导致执行进度较慢，请耐心等待。
②	包括**自动建表**和**使用已有表**。 **说明** 暂不支持同步没有主键的表。但只要选择的表中包括有主键的表，会正常执行流程，没有主键的表会被忽略。
③	选择的**表建立方式**不同，此处显示的MaxCompute表名也不同： ■ 当**表建立方式**选择**自动建表**时，单击**下一步**，会显示**自动建表**对话框。请单击**开始建表**，创建成功后，单击**完成**。您可以单击表名称，查看和修改建表语句。 ■ 当**表建立方式**选择**使用已有表**时，请在下拉列表中选择需要的表。

ix. 在**运行资源设置**页面，配置**来源端读取支持最大连接数**和**目标端写入并发数**，并单击工具栏中的图标。

- 当选择**同步方式**为**数据库迁至DataHub**时，操作如下：
  i. 双击打开实时同步节点的编辑页面，单击右侧的**基本配置**，从**资源组**下拉列表中选择需要使用的资源组。
  ii. 在**数据来源**区域，选择**类型**和**数据源**。
  iii. 在**选择同步的源表**区域，选中需要同步的**源表**，单击图标，将其移动至**已选源表**。

  该区域会为您展示所选数据源下所有的表，您可以选择整库全表和部分表进行同步。

> **注意** 如果选中的表没有主键，将无法进行实时同步。

iv. 在**设置同步规则**区域，单击**添加规则**，选择相应的规则进行添加。

同步规则包括**源表名和Topic转换规则**和**目标Topic规则**。

v. 单击**下一步**。

vi. 在**设置目标表**页面，选择**目标DataHub数据源**，单击**刷新源表和DataHub Topic映射**，创建需要同步的源表和目标Topic的映射关系。

vii. 查看任务的执行进度和Topic来源，单击**下一步**。

序号	描述
①	显示映射关系的创建进度。 **说明** 如果同步的Topic数量较多，会导致执行进度较慢，请耐心等待。
②	包括**自动建表**和**使用已有Topic**。
③	选择的**Topic建立方式**不同，此处显示的DataHub Topic也不同： ■ 当**Topic建立方式**选择**自动建表**时，单击**下一步**，会显示**自动建表**对话框。请单击**开始建表**，创建成功后，单击**完成**。 ■ 当**Topic建立方式**选择**使用已有Topic**时，请在下拉列表中选择需要的Topic。

viii. 在**运行资源设置**页面，配置**来源端读取支持最大连接数**和**目标端写入并发数**，并单击工具栏中的图标。

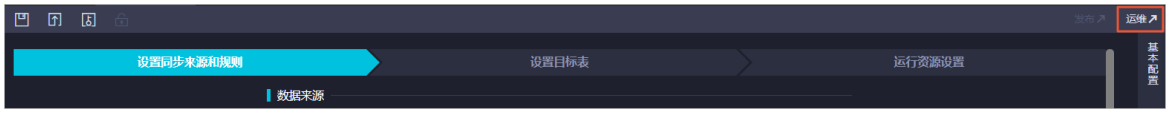
提交实时同步节点

1. 在实时同步节点的编辑页面，单击工具栏中的图标。
2. 在**提交新版本**对话框中，输入**变更描述**。
3. 单击**确认**。

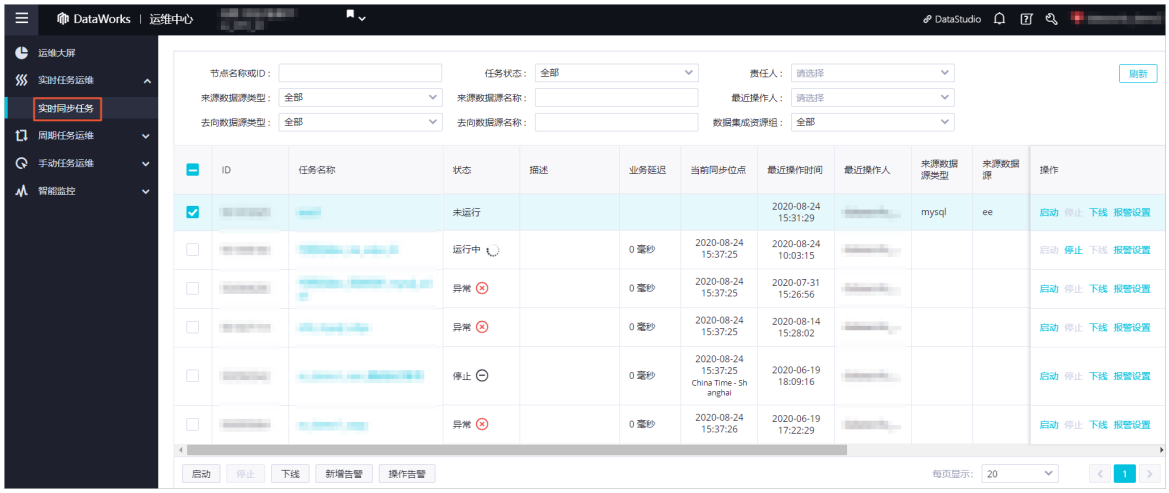
   如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。详情请参见发布管理。

运维实时同步节点

1. 提交或发布节点成功后，单击页面右上方的**运维**，进入**实时任务运维 > 实时同步任务**页面。



2. 您可以在**实时同步任务**页面，单击相应的**任务名称**，查看详细的运维信息。



您可以在该页面对实时同步节点进行**启动**、**停止**、**下线**和**报警设置**等操作：

- 启动非运行状态的任务：
  a. 单击相应任务后的**启动**。

b. 在**启动**对话框中，配置各项参数。

参数	描述
是否重置位点	如果选中该参数，请设置下次启动的时间位点。即**启动时间点位**和**时区**为必选项。
启动时间点位	选择启动节点的日期和时间。
时区	从**时区**下拉列表中选择时区。
任务自动结束	■ 配置脏数据的最大容忍条数。如果您配置为0，表示严格不允许脏数据存在。如果不配置，则代表容忍脏数据。 ■ 如果您不配置Failover次数，将根据5分钟Failover 100次来自动结束任务，避免频繁启动占用系统资源。

c. 单击**确认**。

- 停止运行中的任务：
  a. 单击相应任务后的**停止**。
  b. 在确认对话框中，单击**停止**。
- 下线非运行状态的任务：
  a. 单击相应任务后的**下线**。
  b. 在确认对话框中，单击**下线**。
- 单击相应任务后的**报警设置**，您可以在该页面查看**报警时间**和**报警规则**。
- 新增告警：
  a. 选中需要新增告警的任务，单击页面下方的**新增告警**。
  b. 在**新建规则**对话框中，配置各项参数。

参数	描述
名称	新建规则的名称，必填项。
描述	对新建规则进行简单描述。
指标	包括**任务状态**、**业务延迟**、**Failover**、**脏数据**和**DDL错误**。 **说明** ■ Failover：任务运行出错，各种原因都有可能。 ■ 脏数据：正常读取的数据，但无法正常写入。关于脏数据的完整定义，您可以参考：基本概念。 ■ DDL不支持：由于不支持来源DDL操作，导致的错误。
阈值	设置**WARNING**和**CRITICAL**的阈值，默认值为5分钟。 **说明** 无心跳：指管控与执行层之间的信号中断了，有可能是管控信号网络断了，或者任务异常挂了。
报警间隔	设置报警的时间间隔，默认值为5分钟内只发一次报警。
WARNING	包括**邮件**、**短信**、**电话**和**钉钉**。
CRITICAL	

参数	描述
接收人（非钉钉）	从**接收人（非钉钉）**下拉列表中选择接收人。

c. 单击**确认**。

- 操作告警：
  a. 选中需要操作告警的任务，单击页面下方的**操作告警**。
  b. 在**操作告警**对话框中，选中**操作类型**和**告警指标**。
     选中要操作的告警类型后，其对应的所有规则会被批量修改。
  c. 单击**确认**。

6.3. MaxCompute节点

6.3.1. 创建ODPS SQL节点

ODPS SQL采用类似SQL的语法，适用于海量数据（TB级）但实时性要求不高的分布式处理场景。

前提条件

您在**工作空间配置**页面添加**MaxCompute**计算引擎实例后，当前页面才会显示**MaxCompute**目录。详情请参见配置工作空间。

背景信息

因为每个作业从前期准备到提交等阶段都需要花费较长时间，因此如果要求处理几千至数万笔事务的业务，您可以使用ODPS SQL顺利完成。ODPS SQL是主要面向吞吐量的OLAP应用，详情请参见与标准SQL的主要区别及解决方法。

使用限制

ODPS SQL节点的使用限制如下：

- ODPS SQL不支持单独使用set、use语句，必须和具体的SQL语句一起执行，示例如下。

```
set a=b;
create table name(id string);
```

- ODPS SQL不支持关键字（set、use）语句后单独加注释，示例如下。

```
create table name(id string);
set a=b; --注释    //ODPS SQL不支持在set语句后添加"--注释"。
create table name1(id string);
```

- ODPS SQL不支持在已完结的语句结尾加注释，示例如下。

> **说明** SQL语句后添加英文分号（;），表示语句已完结。

```
select *   --注释    //"select *"语句未完结，因此"--注释"这个注释可以添加。
from dual; --注释    //"from dual;"语句已完结，因此"--注释"这个注释不支持添加。
show tables;
```

- 数据开发与调度运行的区别如下：

- 数据开发：合并当前任务代码内所有的关键字（set、use）语句，作为所有SQL的前置语句。
- 调度运行：按照顺序执行。

```
set a=b;
create table name1(id string);
set c=d;
create table name2(id string);
```

运行结果如下表所示。

执行SQL	数据开发	调度运行

第一条SQL语句	`set a=b;`
	`set c=d;`
	`create table name1(id string);`
		`set a=b;`
		`create table name1(id string);`

第二条SQL语句	`set a=b;`
	`set c=d;`
	`create table name2(id string);`
		`set c=d;`
		`create table name2(id string);`

- 调度参数配置必须是 `key=value` 的格式，且（=）前后不支持空格，示例如下。

```
time={yyyymmdd hh:mm:ss} //错误
a =b //错误
```

- 如果设置bizdate、date等关键字作为调度参数变量，格式必须是yyyymmdd。如果需要其它格式，请使用其它变量名称，避免冲突，示例如下。

```
bizdate=201908 //错误，不支持。
```

- 数据开发需要查询结果，仅支持select、read和with起始的SQL语句，否则无结果输出。
- 执行多条SQL语句时，请用分号（;）分隔，且需要换行。
  - 错误示例

    ```
    create table1;create table2
    ```

  - 正确示例

    ```
    create table1;
    create table2;
    ```

- MaxCompute 2.0扩展函数使用到新数据类型时，您需要在该函数的SQL语句前加 `set odps.sql.type.system.odps2=true;` ，并与SQL一起提交运行，以便正常使用新数据类型。
- SQL语句中添加注释时，不支持在注释中使用英文分号（;）。

  错误示例：

```
create table1;  //创建表格table1;再创建表格table2
create table2;
```

- 使用ODPS SQL节点进行SQL任务开发时，SQL代码大小不能超过200KB，SQL命令条数不能超过200条。

操作步骤

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. 鼠标悬停至+新建图标，单击**MaxCompute > ODPS SQL**。

   您也可以找到相应的业务流程，右键单击**MaxCompute**，选择**新建 > ODPS SQL**。
3. 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。

4. 单击**提交**。
5. 在节点的编辑页面，编辑并运行代码。

   新建节点成功后，编写符合语法的ODPS SQL代码，SQL语法请参见SQL概述。

   > **说明** 由于国际标准化组织发布的中国时区信息调整，通过DataWorks执行相关SQL时，日期显示某些时间段会存在时间差异：1900~1928年的日期时间差异5分52秒，1900年之前的日期时间差异9秒。

   DataWorks不允许节点代码中仅包含`set`语句。如果您需要运行SET语句，可以和其它SQL语句一起执行，如下所示。

   ```
   set odps.sql.allow.fullscan=true;
   select 1;
   ```

   SET语句的详情请参见SET操作。

   以创建一张表并向表中插入数据，查询结果为例，操作如下：

   i. 创建一张表test1。

   ```
   CREATE TABLE IF NOT EXISTS test1
   ( id BIGINT COMMENT '' ,
     name STRING COMMENT '' ,
     age BIGINT COMMENT '' ,
     sex STRING COMMENT '');
   ```

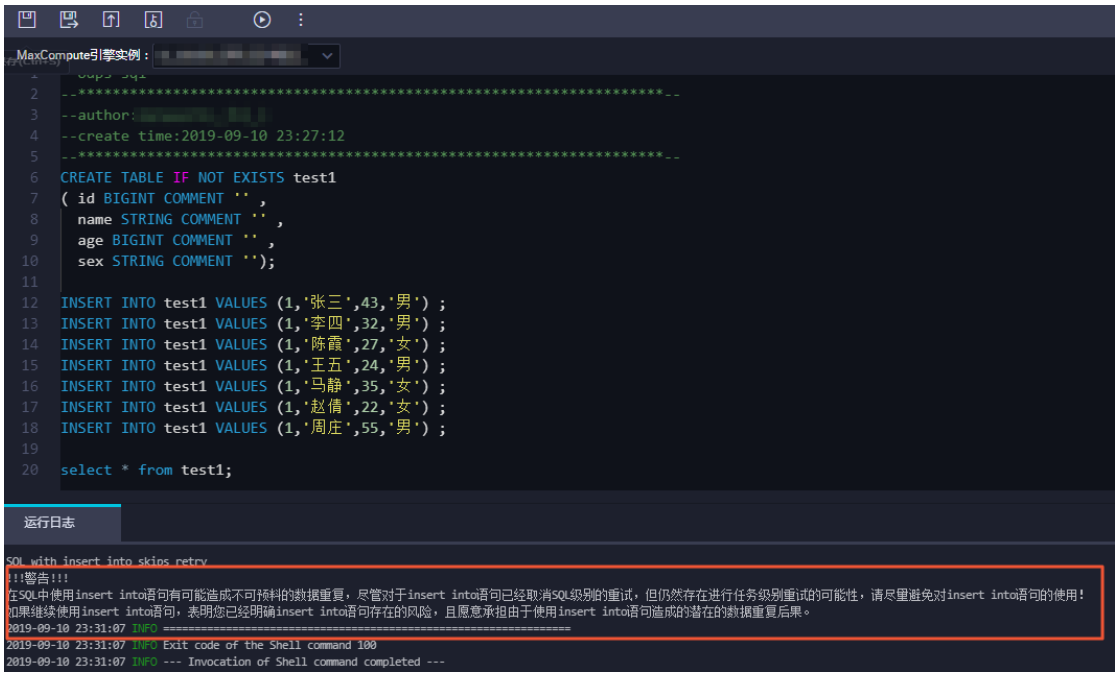
ii. 插入准备好的数据。

```
INSERT INTO test1 VALUES (1,'张三',43,'男') ;
INSERT INTO test1 VALUES (1,'李四',32,'男') ;
INSERT INTO test1 VALUES (1,'陈霞',27,'女') ;
INSERT INTO test1 VALUES (1,'王五',24,'男') ;
INSERT INTO test1 VALUES (1,'马静',35,'女') ;
INSERT INTO test1 VALUES (1,'赵倩',22,'女') ;
INSERT INTO test1 VALUES (1,'周庄',55,'男') ;
```

iii. 查询表数据。

```
select * from test1;
```

iv. SQL语句编辑完成后，单击工具栏中的图标，系统会按照从上往下的顺序执行SQL语句，并打印日志。



> **说明**
>
> - 如果当前工作空间绑定多个MaxCompute计算引擎，请选择需要的MaxCompute引擎实例后，再运行查询语句。
> - 如果您选中的MaxCompute引擎实例使用的是按量计费默认资源组，则可以在运行语句前，单击工具栏中的图标，预估此次运行产生的费用（实际费用请以账单为准）。

您在SQL中使用 `insert into` 语句有可能造成不可预料的数据重复。虽然已经对 `insert into` 语句取消SQL级别的重试，但仍然存在进行任务级别重试的可能性，请尽量避免使用 `insert into` 语句。如果使用，运行日志中会出现如下提示。

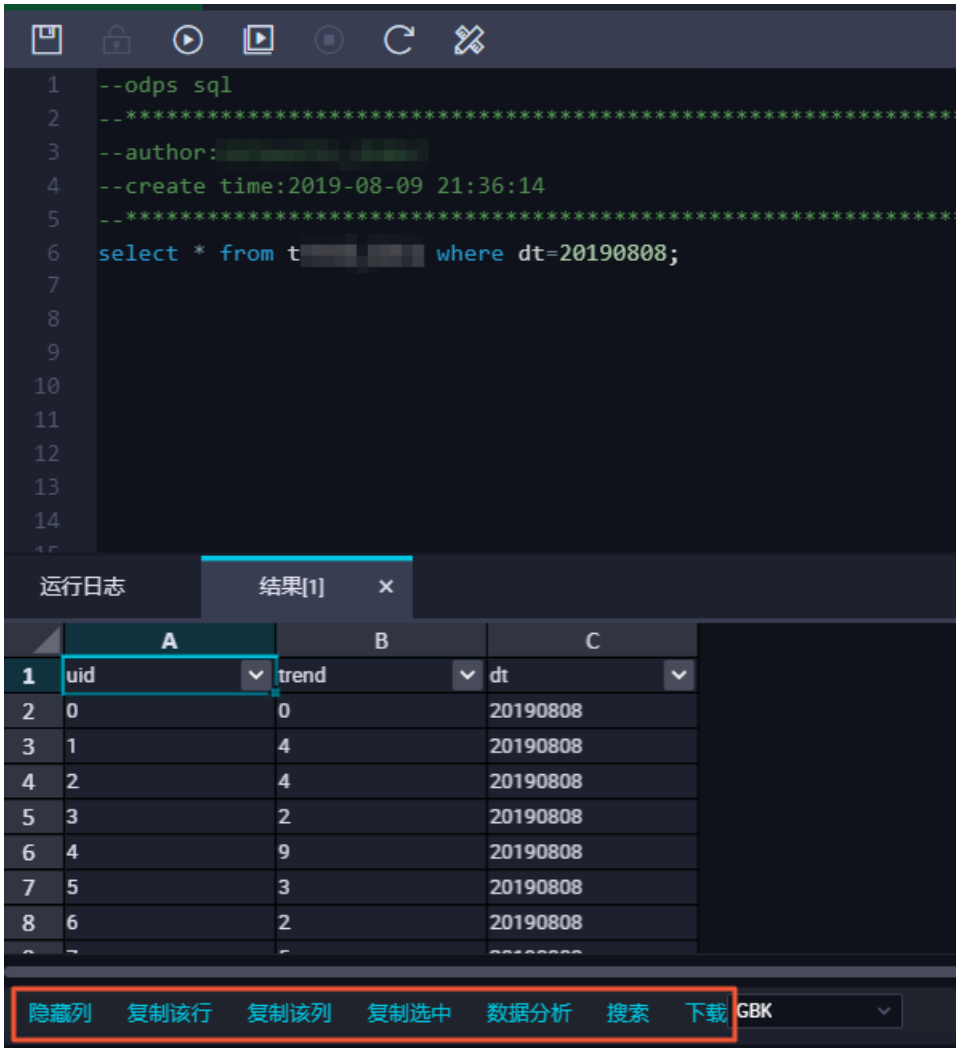
```
在SQL中使用insert into语句有可能造成不可预料的数据重复，尽管对于insert into语句已经取消SQL级别的重试，但仍然存在进行任务级别重试的可能性，请尽量避免对insert into语句的使用！
如果继续使用insert into语句，表明您已经明确insert into语句存在的风险，且愿意承担由于使用insert into语句造成的潜在的数据重复后果。
```

v. 执行无误后，单击工具栏中的图标，保存当前SQL代码。

vi. 查看执行结果。

DataWorks的查询结果接入了电子表格功能，方便您对数据结果进行操作。

查询的结果，会直接以电子表格的形式展示。您可以在DataWorks中执行操作，或者在电子表格中打开，也可以自由复制内容粘贴至本地Excel中。



操作	描述
隐藏列	选中需要隐藏的一列或多列后，单击**隐藏列**。
复制该行	左侧选中需要复制的一行或多行后，单击**复制该行**。
复制该列	顶部选中需要复制的一列或多列后，单击**复制该列**。
复制选中	选中需要复制的内容后，单击**复制选中**。
数据分析	单击**数据分析**，即可跳转至**数据分析**页面。
搜索	单击**搜索**后，在查询结果的右上角会出现搜索框，方便对表中的数据进行搜素。
下载	支持下载GBK和UTF-8两种格式。

6. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性。详情请参见配置基础属性。
7. 提交节点。

> **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

i. 单击工具栏中的图标。

ii. 在**提交新版本**对话框中，输入**变更描述**并选中**我确认继续执行提交操作**。

如果出现**输入输出和代码血缘分析不匹配**的告警，请确认是否忽略该告警，或是否需要调整依赖关系。详情请参见同周期调度依赖逻辑说明。

iii. 单击**确认**。

如果您使用的是标准模式的工作空间，提交成功后，请单击右上角的**发布**。具体操作请参见发布任务。

8. 测试节点，详情请参见查看并管理周期任务。

6.3.2. 创建SQL组件节点

SQL组件是一种带有多个输入参数和输出参数的SQL代码模板。使用SQL代码处理数据表时，通过过滤、连接和聚合源数据表，获取结果表。

前提条件

- 您需要购买DataWorks标准版及以上版本，才可以使用SQL组件节点功能。
- 您在**工作空间配置**页面添加MaxCompute计算引擎实例后，当前页面才会显示MaxCompute目录。详情请参见配置工作空间。
- 请确保您已准备好需要使用的组件，详情请参见创建组件。

背景信息

在组件的开发者发布新版本后，组件的使用者可以选择是否升级现有组件的使用实例至最新版本。

组件的版本机制支持开发者对组件不断升级，提升流程的执行效率并优化业务效果。

例如，用户A使用用户B旧版本的组件时，用户B升级组件至新版本。用户A收到组件的更新提醒后，可以根据自身需求决定是否升级。

升级根据组件模板开发的SQL组件节点时，需要单击**更新代码版本**，然后确认新版本SQL组件节点的参数配置是否继续有效。根据组件新版本的说明进行调整后，即可和普通SQL节点开发的流程一致，保存提交并进入发布流程。

操作步骤

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. 鼠标悬停至 +新建 图标，单击**MaxCompute > SQL组件节点**。

您也可以找到相应的业务流程，右键单击**MaxCompute**，选择**新建 > SQL组件节点**。

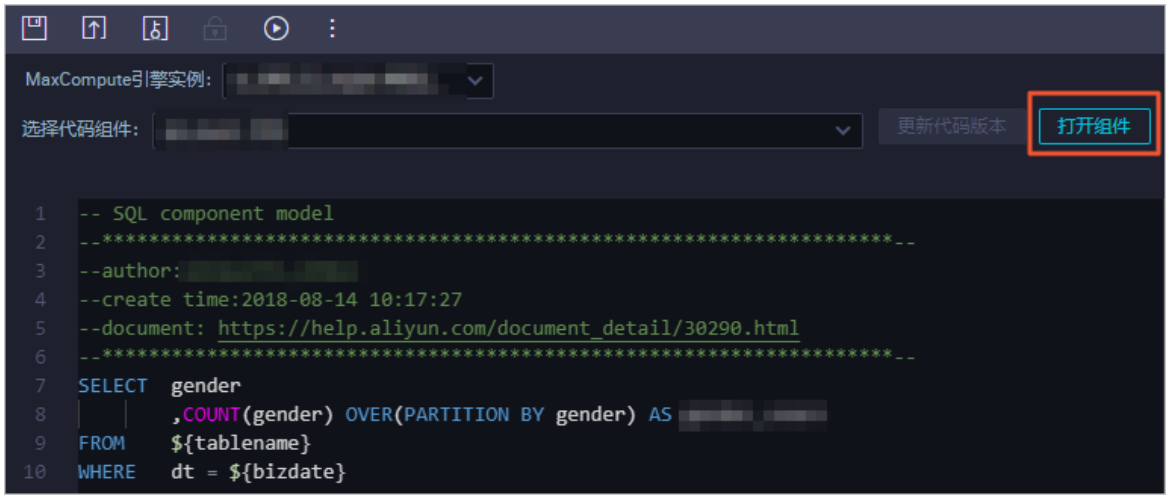
3. 在**新建节点**对话框中，输入**节点名称**，并选择**目标文件夹**。

> **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。

4. 单击**提交**。
5. 在SQL组件节点的编辑页面，**选择代码组件**。

   如果当前工作空间绑定了多个MaxCompute计算引擎，请选择**MaxCompute引擎实例**。

   选中组件后，您可以单击**打开组件**，进入组件的详情页面。



   为提高开发效率，数据任务的开发者可以使用工作空间成员和租户成员贡献的组件，来新建数据处理节点：

   - 本工作空间成员创建的组件在**组件**下。
   - 租户成员创建的组件在**公共组件**下。

6. 单击节点编辑页面右侧的**参数配置**，为选择的组件指定参数。
7. 保存并提交节点。

   > **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标，保存节点。

   ii. 单击工具栏中的图标。

   iii. 在**提交新版本**对话框中，输入**变更描述**。

   iv. 单击**确认**。

   如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

8. 测试节点，详情请参见查看并管理周期任务。

6.3.3. 创建ODPS Spark节点

ODPS Spark节点支持使用Java和Python处理数据，本文为您介绍如何新建和配置ODPS Spark节点。

背景信息

Python资源是针对Python UDF进行开发，能够获取的可以直接依赖的三方包较为有限，因此使用Python资源的局限性较大。如果您需要使用Python资源中未支持的三方包，请参见在MaxCompute UDF中运行Scipy。

PyODPS 2和PyODPS 3节点对Python资源的支持性更强大，详情请参见创建PyODPS 2节点和创建PyODPS 3节点。

本文分别为您介绍如何创建JAR资源和Python资源，并在创建ODPS Spark节点后加载不同类型的资源，您可以根据业务需求进行操作。

创建JAR资源

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. 鼠标悬停至**+新建**图标，单击**MaxCompute > 资源 > JAR**。

   您也可以找到相应的业务流程，右键单击**MaxCompute**，选择**新建 > 资源 > JAR**。
3. 在**新建资源**对话框中，输入**资源名称**，并选择**目标文件夹**。

   > **说明**
   > - 如果绑定多个实例，则需要选择**MaxCompute引擎实例**。
   > - 如果该JAR包已经在MaxCompute（ODPS）客户端上传过，则需要取消勾选**上传为ODPS资源**，否则上传会报错。
   > - 资源名称无需与上传的文件名保持一致。
   > - 资源名称命名规范：1~128个字符，字母、数字、下划线、小数点，大小写不敏感，JAR资源的后缀为.jar，Python资源的后缀为.py。
4. 单击**点击上传**，选择相应的文件进行上传。

   WordCount的示例代码请参见WordCount。
5. 单击**新建**。
6. 单击工具栏中的图标，上传文件。
7. 在**提交新版本**对话框中，输入**变更描述**，单击**确认**。

创建Python资源

1. 在**数据开发**页面，鼠标悬停至**+新建**图标，单击**MaxCompute > 资源 > Python**。

   您也可以找到相应的业务流程，右键单击**MaxCompute**，选择**新建 > 资源 > Python**。
2. 在**新建资源**对话框中，输入**资源名称**，并选择**目标文件夹**。

   > **说明**
   > - 如果绑定多个实例，则需要选择**MaxCompute引擎实例**。
   > - 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。
   > - 创建的Python资源仅支持Python 2.x和Python 3.x版本的Python代码。
3. 单击**新建**。

4. 在节点的编辑页面，输入Python代码。

   代码示例如下，仅进行校检数值判断，非数据业务处理逻辑。

```
# -*- coding: utf-8 -*-
import sys
from pyspark.sql import SparkSession
try:
    # for python 2
    reload(sys)
    sys.setdefaultencoding('utf8')
except:
    # python 3 not needed
    pass
if __name__ == '__main__':
    spark = SparkSession.builder\
        .appName("spark sql")\
        .config("spark.sql.broadcastTimeout", 20 * 60)\
        .config("spark.sql.crossJoin.enabled", True)\
        .config("odps.exec.dynamic.partition.mode", "nonstrict")\
        .config("spark.sql.catalogImplementation", "odps")\
        .getOrCreate()
def is_number(s):
    try:
        float(s)
        return True
    except ValueError:
        pass
    try:
        import unicodedata
        unicodedata.numeric(s)
        return True
    except (TypeError, ValueError):
        pass
    return False
print(is_number('foo'))
print(is_number('1'))
print(is_number('1.3'))
print(is_number('-1.37'))
print(is_number('1e3'))
```

5. 单击工具栏中的图标，上传文件。
6. 在**提交新版本**对话框中，输入**变更描述**，单击**确认**。

创建ODPS Spark节点

1. 在**DataStudio（数据开发）**页面，鼠标悬停至**+新建**图标，单击**MaxCompute > ODPS Spark**。

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

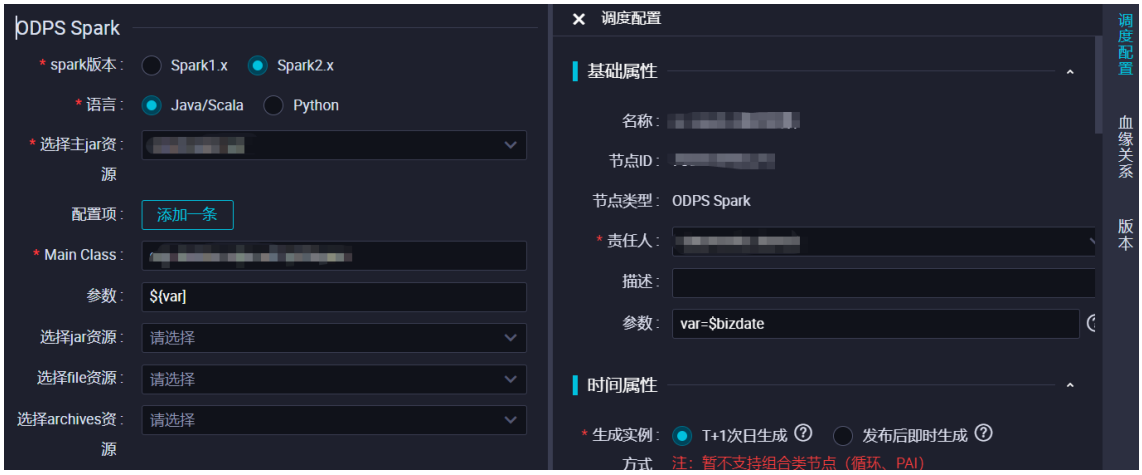
2. 在**新建节点**对话框中，输入**节点名称**，并选择**目标文件夹**。

> **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。

3. 单击**提交**。
4. 在ODPS Spark编辑页面，配置各项参数。ODPS Spark的详情请参见概述。

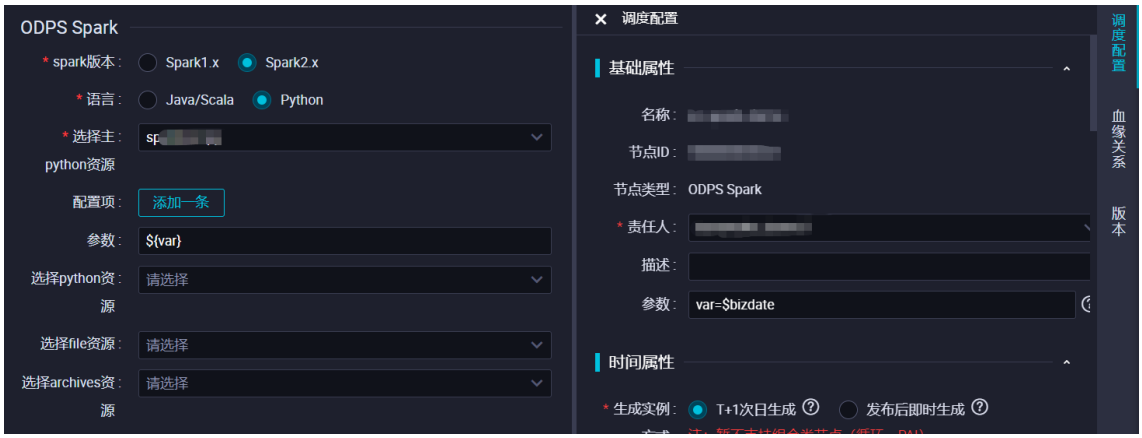
   ODPS Spark节点支持两种**spark版本**和**语言**。选择不同的**语言**，会显示相应不同的配置，您可以根据界面提示进行配置：

   - 选择**语言**为**Java/Scala**，配置如下。



参数	描述
spark版本	包括Spark1.x和Spark2.x两个版本。
语言	此处选择Java/Scala。
选择主jar资源	从下拉列表中选择您已上传的JAR资源。
配置项	单击**添加一条**，即可配置key和value。
Main Class	选择类名称。
参数	添加参数，多个参数之间用空格分隔。支持使用调度参数，配置中直接使用${变量名}，在右侧调度配置参数处给变量赋值。调度参数使用方式请参考文档：调度参数概述。 **说明** 您需要在配置调度参数后，再在编辑页面配置节点参数，系统会顺序执行。
选择jar资源	ODPS Spark节点根据上传的文件类型自动过滤，选择下拉框中显示的您已上传的JAR资源。
选择file资源	ODPS Spark节点根据上传的文件类型自动过滤，选择下拉框中显示的您已上传的File资源。
选择archives资源	ODPS Spark节点根据上传的文件类型自动过滤，选择下拉框中显示的您已上传的Archives资源，仅展示压缩类型的资源。

- 选择**语言**为**Python**，配置如下。



参数	描述
spark版本	包括**Spark1.x**和**Spark2.x**两个版本。
语言	此处选择**Python**。
选择主python资源	从下拉列表中选择您已创建的Python资源。
配置项	单击**添加一条**，即可配置key和value。
参数	添加参数，多个参数之间用空格分隔。支持使用调度参数，配置中直接使用${变量名}，在右侧调度配置参数处给变量赋值。调度参数使用方式请参考文档：调度参数概述
选择python资源	ODPS Spark节点根据上传的文件类型自动过滤，选择下拉框中显示的您已上传的Python资源。
选择file资源	ODPS Spark节点根据上传的文件类型自动过滤，选择下拉框中显示的您已上传的File资源。
选择archives资源	ODPS Spark节点根据上传的文件类型自动过滤，选择下拉框中显示的您已上传的Archives资源，仅展示压缩类型的资源。

5. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性，详情请参见配置基础属性。
6. 保存并提交节点。

> **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标，保存节点。
   ii. 单击工具栏中的图标。
   iii. 在**提交新版本**对话框中，输入**变更描述**。
   iv. 单击**确认**。

   如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。
7. 测试节点，详情请参见查看并管理周期任务。

6.3.4. 创建PyODPS 2节点

DataWorks提供PyODPS 2节点类型，集成了MaxCompute的Python SDK。您可以在DataWorks的PyODPS 2节点上，直接编辑Python代码，用于操作MaxCompute。

背景信息

MaxCompute提供了Python SDK方法说明，您可以使用Python的SDK来操作MaxCompute。

> **说明**
> - PyODPS 2节点底层的Python版本为2.7。
> - 推荐通过SQL或者Dataframe的方式处理数据，详情请参见DataFrame概述。不建议您直接调用pandas等第三方包来处理数据。
> - PyODPS 2节点获取到本地处理的数据不能超过50 MB，节点运行时占用的内存不能超过1 GB，否则节点任务会结束运行。请避免在PyODPS 2节点中写入过多的数据处理代码。
> - Hints参数的详情请参见SET操作。

PyODPS 2节点主要针对MaxCompute的Python SDK应用。对于纯Python代码的执行，您可以使用Shell节点执行上传至DataWorks的Python脚本。如果您需要在PyODPS 2节点中调用第三方包，请参见在PyODPS节点中调用第三方包。

PyODPS操作实践请参见使用MaxCompute分析IP来源最佳实践和PyODPS节点实现结巴中文分词，更多信息请参见PyODPS文档。

新建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. 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. 鼠标悬停至**+新建**图标，单击**MaxCompute > PyODPS 2**。

   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 > PyODPS 2**。

   如果您需要创建业务流程，请参见创建业务流程。
3. 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。
4. 单击**提交**。
5. 编辑PyODPS 2节点。
   i. 进入ODPS入口。

   DataWorks的PyODPS 2节点中，将会包含一个全局的变量odps或o，即ODPS入口，您无需手动定义ODPS入口。

   ```
   print(odps.exist_table('PyODPS_iris'))
   ```

ii. 执行SQL。

PyODPS 2支持ODPS SQL的查询，并可以读取执行的结果。execute_sql或run_sql方法的返回值是运行实例。

并非所有在MaxCompute客户端中可以执行的命令，都是PyODPS 2支持的SQL语句。调用非DDL或非DML语句时，请使用其它方法。

例如，执行GRANT、REVOKE等语句时，请使用run_security_query方法。PAI命令请使用run_xflow或execute_xflow方法。

```
o.execute_sql('select * from dual')  # 同步的方式执行，会阻塞直到SQL执行完成。
instance = o.run_sql('select * from dual')  # 异步的方式执行。
print(instance.get_logview_address())  # 获取logview地址。
instance.wait_for_success()  # 阻塞直到完成。
```

iii. 设置运行参数。

您可以通过设置hints参数，来设置运行时的参数，参数类型是dict。

```
o.execute_sql('select * from PyODPS_iris', hints={'odps.sql.mapper.split.size': 16}
)
```

对全局配置设置sql.settings后，每次运行时，都需要添加相关的运行时的参数。

```
from odps import options
options.sql.settings = {'odps.sql.mapper.split.size': 16}
o.execute_sql('select * from PyODPS_iris')  # 根据全局配置添加hints。
```

iv. 读取SQL执行结果。

运行SQL的实例能够直接执行open_reader的操作，有以下两种情况：

- SQL返回了结构化的数据。

```
with o.execute_sql('select * from dual').open_reader() as reader:
for record in reader:  # 处理每一个record。
```

- 可能执行的是desc等SQL语句，通过reader.raw属性，获取到原始的SQL执行结果。

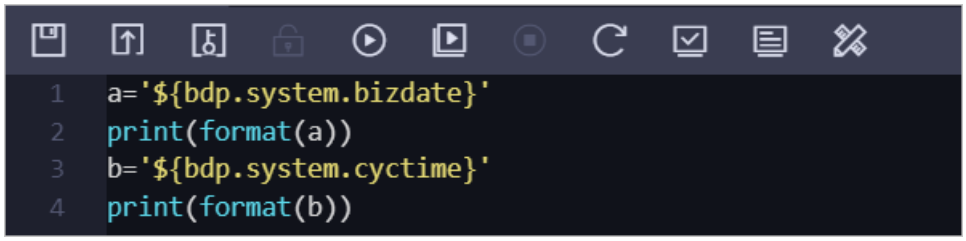
```
with o.execute_sql('desc dual').open_reader() as reader:
print(reader.raw)
```

> **说明** 如果使用了自定义调度参数，页面上直接触发运行PyODPS 2节点时，需要写死时间，PyODPS节点无法直接替换。

6. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性，详情请参见配置基础属性。

PyODPS 2节点可以使用系统定义的调度参数和自定义参数：

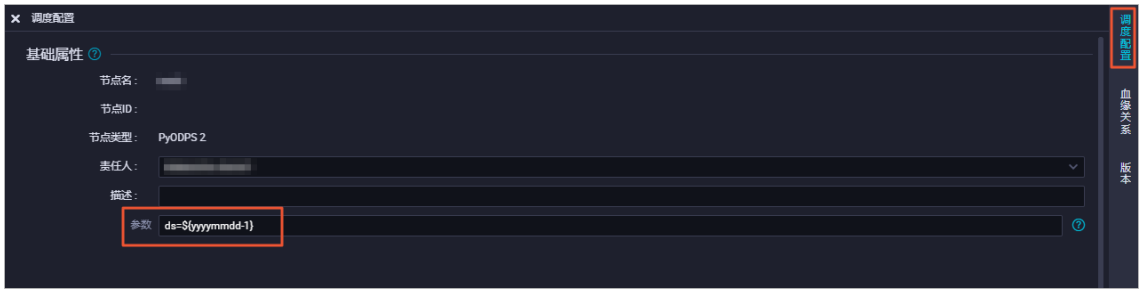
- 如果PyODPS 2使用系统定义的调度参数，可以直接在页面赋值。



```
1 a='${bdp.system.bizdate}'
2 print(format(a))
3 b='${bdp.system.cyctime}'
4 print(format(b))
```

> **说明** 由于公共资源组无法直接访问外网环境，建议您有公网访问需求时，使用自定义资源组或独享调度资源。仅DataWorks专业版提供自定义资源组，任意版本都可以购买独享调度资源，详情请参见DataWorks独享资源组。

- 您也可以在**调度配置 > 基础属性**区域，配置自定义参数。



> **说明** 自定义参数需要使用args['参数名']的形式调用，例如 `print (args['ds'])` 。

7. 提交节点。

> **注意** 提交节点前，您需要设置**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。

8. 测试节点，详情请参见查看并管理周期任务。

PyODPS节点预装模块列表

PyODPS节点包括以下预装模块：

- setuptools
- cython
- psutil
- pytz
- dateutil
- requests
- pyDes
- numpy
- pandas
- scipy
- scikit_learn
- greenlet
- six
- 其它Python 2.7内置已安装的模块，如smtplib等。

6.3.5. 创建PyODPS 3节点

本文为您介绍如何创建PyODPS 3节点，以及在DataWorks使用PyODPS 3的限制。

使用限制

- 当Python 3的子版本号不同（例如Python 3.8和Python 3.7）时，字节码的定义有所不同。

  目前MaxCompute使用的Python 3版本为3.7，当使用其它版本Python 3中的部分语法（例如Python 3.8中的finally block）时，执行会报错，建议您选择Python 3.7。
- PyODPS 3节点获取到本地处理的数据不能超过50 MB，节点运行时占用的内存不能超过1 GB，否则节点任务会结束运行。请避免在PyODPS 3节点中写入过多的数据处理代码。
- PyODPS 3支持运行在公共资源组和2020年4月之后购买的独享调度资源组上。如果您的独享调度资源组的创建时间较早，请提交工单升级资源组。

创建PyODPS 3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. 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. 鼠标悬停至**+新建**图标，单击**MaxCompute > PyODPS 3**。

   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 > PyODPS 3**。

   如果您需要创建业务流程，请参见创建业务流程。
3. 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。
4. 单击**提交**。
5. 编辑并执行PyODPS 3节点。

   例如，您在使用execute_sql接口时，需要手动设置SQL运行参数。详情请参见执行SQL。

   ```
   hints={'odps.sql.python.version': 'cp37', 'odps.isolation.session.enable': True}
   ```

   当您使用DataFrame自定义函数（df.map、df.map_reduce、df.apply和df.agg）时，请进行如下设置。

   ```
   hints={'odps.isolation.session.enable': True}
   ```

   PyODPS会根据客户端使用的Python版本决定UDF的运行环境，提交SQL查询语句。例如，通过公共Python UDF执行DataFrame，在客户端使用Python 3时，会根据Python 3进行解释。如果相应的UDF使用print语句等Python 2特有的语法或库，执行语句会报ScriptError的错误。如果您需要在PyODPS 2节点中调用第三方包，请参见在PyODPS节点中调用第三方包。
6. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性，详情请参见配置基础属性。
7. 保存并提交节点。

> **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

i. 单击工具栏中的图标，保存节点。

ii. 单击工具栏中的图标。

iii. 在**提交新版本**对话框中，输入**变更描述**。

iv. 单击**确认**。

如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

8. 测试节点，详情请参见查看并管理周期任务。

6.3.6. 创建ODPS Script节点

ODPS Script节点的SQL开发模式是MaxCompute基于2.0的SQL引擎提供的脚本开发模式。

背景信息

编译脚本时，ODPS Script节点可以将一个多语句的SQL脚本文件作为一个整体进行编译，无需逐条语句进行编译。将其作为一个整体提交运行，可以保证一个执行计划一次排队、一次执行，充分利用MaxCompute的资源。

Script Mode的SQL编译较为简单，您只需要按照业务逻辑，用类似于普通编程语言的方式进行编译，无需考虑如何组织语句。

```
--SET语句
set odps.sql.type.system.odps2=true;
[set odps.stage.reducer.num=***;]
[...]
--DDL语句
create table table1 xxx;
[create table table2 xxx;]
[...]
--DML语句
@var1 := SELECT [ALL | DISTINCT] select_expr, select_expr, ...
    FROM table3
    [WHERE where_condition];
@var2 := SELECT [ALL | DISTINCT] select_expr, select_expr, ...
    FROM table4
    [WHERE where_condition];
@var3 := SELECT [ALL | DISTINCT] var1.select_expr, var2.select_expr, ...
    FROM @var1 join @var2 on ...;
INSERT OVERWRITE|INTO TABLE [PARTITION (partcol1=val1, partcol2=val2 ...)]
    SELECT [ALL | DISTINCT] select_expr, select_expr, ...
    FROM @var3;
[@var4 := SELECT [ALL | DISTINCT] var1.select_expr, var.select_expr, ... FROM @var1
    UNION ALL | UNION
    SELECT [ALL | DISTINCT] var1.select_expr, var.select_expr, ... FROM @var2;
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table_name
    AS
    SELECT [ALL | DISTINCT] select_expr, select_expr, ...
    FROM var4;]
```

ODPS Script节点的使用限制如下：

- 脚本模式支持SET语句、部分DDL语句（结果是屏显类型的语句除外，例如 `desc` 、 `show` ）和DML语句。
- 一个脚本的完整形式是SET语句>DDL语句>DML语句。每种类型语句都可以有0到多个语句，但不同类型的语句不能混合。
- 多个语句以@开始，表示变量连接。
- 一个脚本，目前最多支持一个屏幕显示结果的语句（例如单独的Select语句），否则会报错。不建议您在脚本中执行屏幕显示的Select语句。
- 一个脚本，目前最多支持一个 `Create table as` 语句，并且必须是最后一句。建议将建表语句和Insert语句分开写。
- 脚本模式下，如果有一个语句失败，整个脚本的语句都不会执行成功。
- 脚本模式下，只有所有输入的数据都准备完成，才会生成一个作业进行数据处理。
- 脚本模式下，如果一个表被写入后又被读取，会报错。

```
insert overwrite table src2 select * from src where key > 0;
@a := select * from src2;
select * from @a;
```

为避免先写后读，可以进行如下修改。

```
@a := select * from src where key > 0;
insert overwrite table src2 select * from @a;
select * from @a;
```

示例如下。

```
create table if not exists dest(key string , value bigint) partitioned by (d string);
create table if not exists dest2(key string,value bigint ) partitioned by (d string);
@a := select * from src where value >0;
@b := select * from src2 where key is not null;
@c := select * from src3 where value is not null;
@d := select a.key,b.value from @a left outer join @b on a.key=b.key and b.value>0;
@e := select a.key,c.value from @a inner join @c on a.key=c.key;
@f := select * from @d union select * from @e union select * from @a;
insert overwrite table dest partition (d='20171111') select * from @f;
@g := select e.key,c.value  from @e join @c on e.key=c.key;
insert overwrite table dest2 partition (d='20171111') SELECT * from @g;
```

脚本模式适用于以下场景：

- 脚本模式更适合用于改写需要层层嵌套子查询的单个语句，或因为脚本复杂性而不得不拆成多个语句的脚本。
- 多个输入的数据源数据准备完成的时间相差很大（例如一个凌晨1点可以准备好，另一个上午7点可以准备好），不适合通过table variable衔接，可以拼接为一个大的脚本模式SQL。

操作步骤

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. 鼠标悬停至**+新建**图标，单击**MaxCompute > ODPS Script**。

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

3. 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。

4. 单击**提交**。
5. 在节点的编辑页面编辑脚本，详情请参见开发及提交SQL脚本。
6. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性，详情请参见配置基础属性。
7. 保存并提交节点。

   > **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标，保存节点。

   ii. 单击工具栏中的图标。

   iii. 在**提交新版本**对话框中，输入**变更描述**。

   iv. 单击**确认**。

   如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

8. 测试节点，详情请参见查看并管理周期任务。

6.3.7. 创建ODPS MR节点

MaxCompute提供MapReduce编程接口。您可以通过创建ODPS MR类型节点并提交任务调度，使用MapReduce Java API编写MapReduce程序来处理MaxCompute中的数据。

前提条件

您需要上传并提交、发布使用的资源后，再创建ODPS MR节点。

ODPS MR类型节点的编辑和使用方法，请参见WordCount示例。

操作步骤

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JAR资源。
   i. 鼠标悬停至**+新建**图标，单击**MaxCompute > 资源 > JAR**。

      您也可以找到相应的业务流程，右键单击**MaxCompute**，选择**新建 > 资源 > JAR**。

ii. 在**新建资源**对话框中，输入**资源名称**，并选择**目标文件夹**。

> **说明**
> - 如果绑定多个实例，则需要选择**MaxCompute引擎实例**。
> - 如果该JAR包已经在MaxCompute（ODPS）客户端上传过，则需要取消勾选**上传为ODPS资源**，否则上传会报错。
> - 资源名称无需与上传的文件名保持一致。
> - 资源名称命名规范：1~128个字符，字母、数字、下划线、小数点，大小写不敏感，JAR资源的后缀为.jar，Python资源的后缀为.py。

iii. 单击**点击上传**，在本地选择相应文件后，单击**打开**。

本文以mapreduce example.jar为例。

iv. 在**新建资源**对话框中，单击**确定**。

v. 单击工具栏中的和图标，保存并提交资源至调度开发服务器端。

3. 创建ODPS MR节点。

i. 鼠标悬停至+新建图标，单击**MaxCompute > ODPS MR**。

您也可以找到相应的业务流程，右键单击**MaxCompute**，选择**新建 > ODPS MR**。

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**。

4. 在节点的编辑页面输入代码，示例如下。

```
--创建输入表。
CREATE TABLE if not exists jingyan_wc_in (key STRING, value STRING);
--创建输出表。
CREATE TABLE if not exists jingyan_wc_out (key STRING, cnt BIGINT);
    ---创建系统dual。
    drop table if exists dual;
    create table dual(id bigint); --如果工作空间不存在该伪表，则需要创建并初始化数据。
    ---向系统伪表初始化数据。
    insert overwrite table dual select count(*)from dual;
    ---向输入表wc_in插入示例数据。
    insert overwrite table jingyan_wc_in select * from (
    select 'project','val_pro' from dual
    union all
    select 'problem','val_pro' from dual
    union all
    select 'package','val_a' from dual
    union all
    select 'pad','val_a' from dual
      ) b;
-- 引用刚刚上传的JAR包资源，可以在资源管理栏中找到该资源，右键引用资源。
--@resource_reference{"mapreduce-examples.jar"}
jar -resources mapreduce-examples.jar -classpath ./mapreduce-examples.jar com.aliyun.od
ps.mapred.open.example.WordCount jingyan_wc_in jingyan_wc_out
```

代码说明如下：

- `--@resource_reference`：您可以右键单击资源名称，选择**引用资源**，即可自动产生该条语句。
- `-resources`：引用到的JAR资源文件名。
- `-classpath`：JAR包的路径。由于已经引用了资源，此处路径统一为./下的JAR包。
- `com.aliyun.odps.mapred.open.example.WordCount`：执行过程调用JAR中的主类，需要和JAR中的主类名称保持一致。
- `jingyan_wc_in`：MR的输入表名称，已在上述代码中提前创建。
- `jingyan_wc_out`：MR的输出表名称，已在上述代码中提前创建。
- 一个MR调用多个JAR资源时，classpath写法为 `-classpath ./xxxx1.jar,./xxxx2.jar`，即两个路径之间用英文逗号（,）分隔。

5. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性，详情请参见配置基础属性。
6. 保存并提交节点。

> **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

i. 单击工具栏中的图标，保存节点。

ii. 单击工具栏中的图标。

iii. 在**提交新版本**对话框中，输入**变更描述**。

iv. 单击**确认**。

如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

7. 测试节点，详情请参见查看并管理周期任务。

6.4. EMR节点

6.4.1. 概述（DataWorks on EMR必读）

DataWorks支持基于EMR（E-MapReduce）计算引擎创建Hive、MR、Presto和Spark SQL等节点，实现EMR任务工作流的配置、定时调度和元数据管理等功能，帮助EMR用户更好地生产数据。本文为您介绍在DataWorks上进行EMR作业等操作的注意事项，建议操作前仔细查看。

使用限制

DataWorks上进行EMR作业和其他操作时，不同EMR的版本或配置情况，会对DataWorks的功能上有不同的限制和影响。

DataWorks功能应用/限制	EMR集群版本/配置要求
使用DataWorks数据地图中元数据表的产出信息、自动推荐功能。	EMR的集群版本大于3.33.0 或4.6版本，版本过低则不支持。
使用DataWorks的创建和使用EMR资源、创建EMR表和注册EMR函数功能。	EMR集群**没有开启**高安全模式（Kerberos或者LDAP），开启了则不支持。
创建使用除Hive节点、Spark节点外的其他节点。	EMR集群**没有开启**高安全模式（Kerberos或者LDAP），开启了则仅支持创建使用Hive节点、EMR Spark节点、EMR Spark SQL节点、EMR Spark Streaming节点，其他类型节点不支持使用。
• DataWorks暂不支持EMR的Flink任务。 • 仅DataWorks Hive节点支持采集EMR元数据、血缘信息，其他节点不支持。 • 如果您的EMR集群、或者DataWorks的独享调度资源组是2021年8月1日前购买创建的，需提交工单，申请升级DataWorks OnEMR的agent到最新版本。	

使用前的准备

在DataWorks上使用进行EMR任务前，您需要完成以下准备。

- 购买并配置独享调度资源组。

  运行EMR任务时，请使用独享调度资源组，因此在进行EMR任务前，您需要购买一个独享调度资源组，并与当前EMR集群所在的VPC连通网络。购买并配置独享调度资源组的操作请参见独享调度资源组概述。

- EMR集群配置检查。

  在DataWorks上进行EMR作业前，您需要检查EMR的部分关键配置是否满足要求，否则可能会导致在DataWorks上运行EMR作业时出错，主要需要保障EMR的配置满足以下要求：

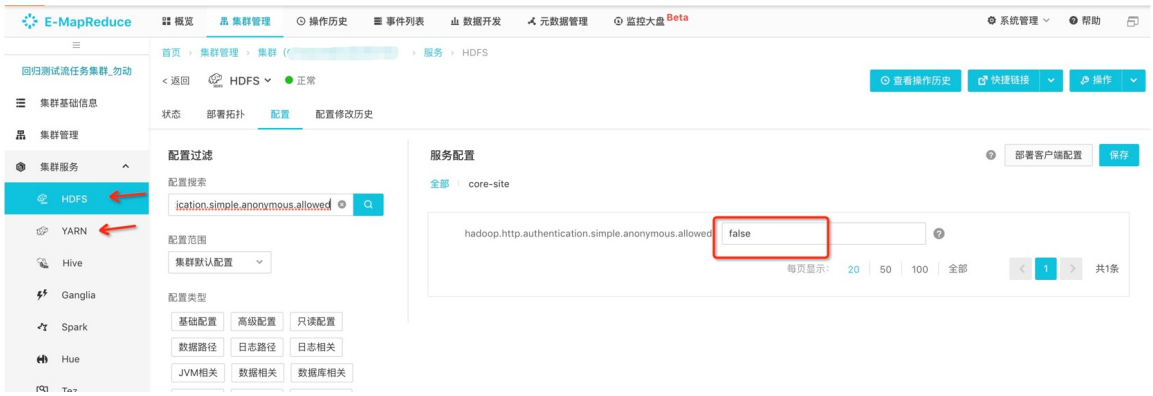
  - 您已创建阿里云EMR集群，且集群所在的安全组中入方向的安全策略包含以下策略。
    - 授权策略：允许
    - 协议类型：自定义 TCP
    - 端口范围：8898/8898
    - 授权对象：100.104.0.0/16

- 如果EMR启用了Ranger，则使用DataWorks进行EMR的作业开发前，您需要在EMR中修改配置，添加白名单配置并重启Hive，否则作业运行时会报错**Cannot modify spark.yarn.queue at runtime**或**Cannot modify SKYNET_BIZDATE at runtime**。
  a. 白名单的配置通过EMR的自定义参数，添加Key和Value进行配置，以Hive组件的配置为例，配置值如下。

  ```
  hive.security.authorization.sqlstd.confwhitelist.append=tez.*|spark.*|mapred.*|mapreduce.*|ALISA.*|SKYNET.*
  ```

  > **说明** 其中 `ALISA.*` 和 `SKYNET.*` 为DataWorks专有的配置。

  b. 白名单配置完成后需要重启服务，重启后配置才会生效。重启服务的操作详情请参见重启服务。
- 您需要在EMR控制台将集群HDFS配置项中的hadoop.http.authentication.simple.anonymous.allowed参数设置为true，并重启hdfs、yarn组件。



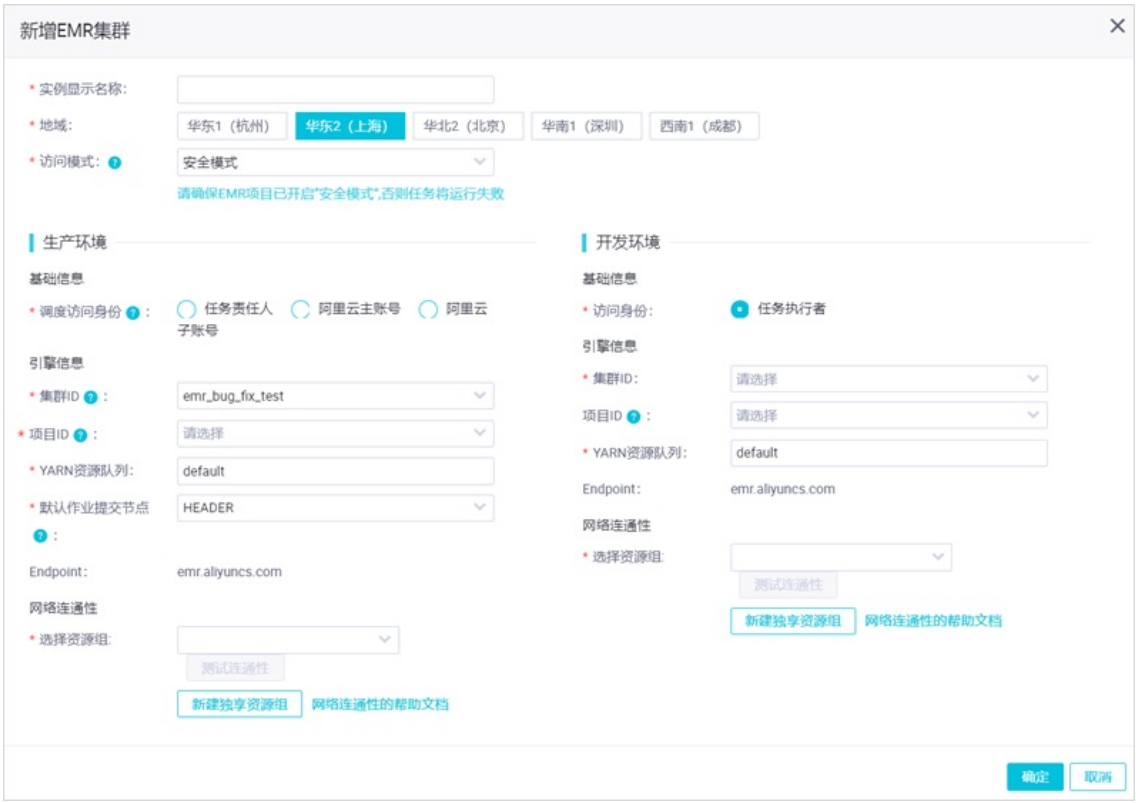
- 绑定EMR引擎到DataWorks的工作空间。

  您需要绑定EMR引擎到DataWorks工作空间后，才能创建EMR节点和进行EMR作业等操作，绑定EMR引擎时，需根据EMR集群的配置来选择不同的访问模式，以下为操作的注意事项，操作详情请参见准备工作：绑定EMR引擎。
  - EMR集群没有开启LDAP时，绑定引擎时，访问模式选择快捷模式。
  - EMR集群开启了LDAP时，绑定引擎时，访问模式选择安全模式。

    此种场景下，需要关注几个配置要点：
    - 在EMR管控台：对应的组件一键开启LDAP后，重启服务。

- 在绑定的EMR集群对应的项目中，开启安全模式。

- 在DataWorks绑定的EMR引擎时，访问模式选择安全模式。



- （重要）在EMR引擎配置页面，做好各个RAM用户对应的LDAP用户映射关系。

- （Impala适用）如果Impala开启了LDAP，需要在DataWorks运行作业，还需要做如下操作。
  a. 下载Impala JDBC驱动。

     开启LDAP认证后，JDBC访问Impala需要提供LDAP认证凭据，同时需要前往Cloudera官网下载Impala JDBC驱动并将其添加到至/usr/lib/hive-current/lib/目录下。您可单击Impala JDBC驱动进行下载。

  b. 下载完毕后，需要将下载后的JAR包拷贝到EMR集群的header/gateway节点的目录：*/usr/lib/flow-agent-current/zeppelin/interpreter/jdbc/*下。

  c. 在EMR控制台重启对应的服务Flow Agent Daemon。

EMR节点及任务调测

完成准备工作后，您即可以创建EMR节点，并在DataWorks上编译运行EMR作业。创建EMR节点及调测任务时，您需关注以下注意事项。

- 高级参数
  - "USE_GATEWAY":true，表示任务会被提交到EMR gateway上执行，默认提交到header节点。
  - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx"，设置spark任务运行参数，多个参数在该key中追加。
  - “queue”：提交作业的调度队列，默认为default队列。

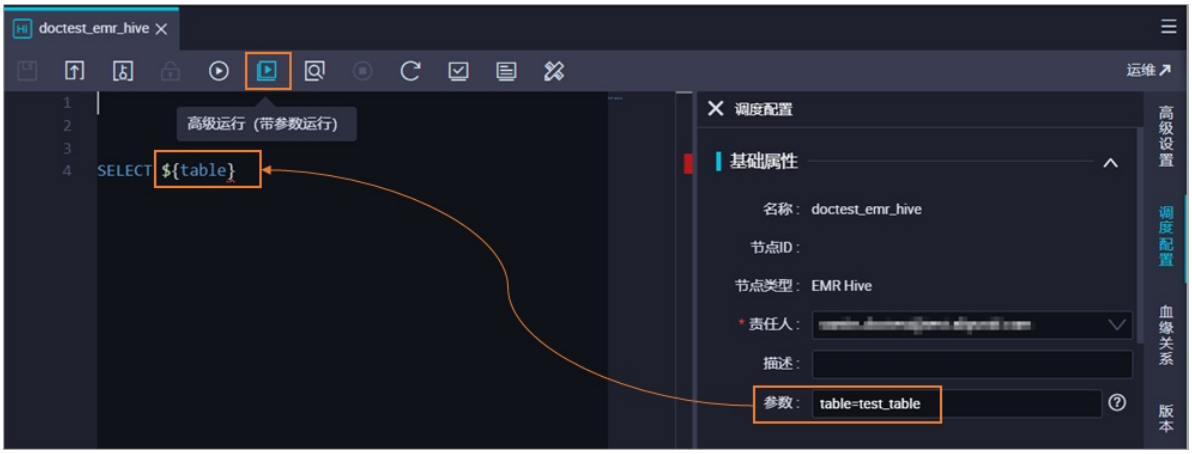
    > **说明** 与在绑定EMR集群设置的优先级比较，此处设置的队列优先级更高，后续版本会支持默认绑定的队列参数。

  - “vcores”：虚拟核数，默认为1，不建议修改此默认值。
  - “memory”：内存，默认为2048MB（用于设置启动器Launcher的内存配额），不建议修改此默认值。
  - “priority”：优先级，默认为1。

- “FLOW_SKIP_SQL_ANALYZE”：SQL语句执行方式，参数值为false表示每次执行一条SQL语句；参数值为true表示每次执行多条SQL语句。

- 调测运行

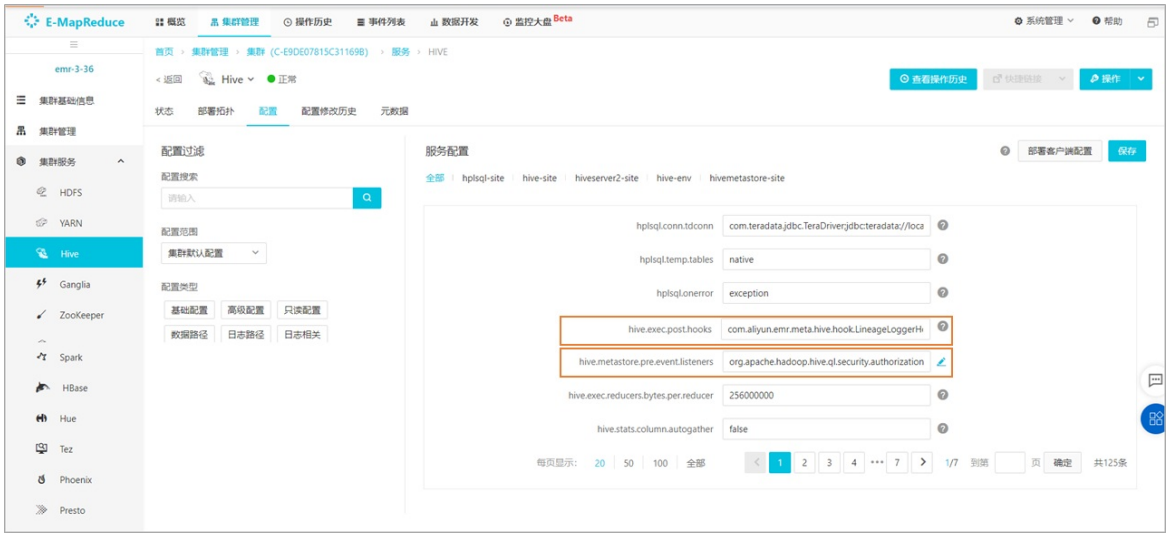
  任务代码中，如果有参数变量使用，务必在**调度配置 > 参数**中增加对应的变量申明，在数据开发使用**高级运行**来进行任务调测。



数据地图

使用DataWorks采集EMR元数据时，您需要检查EMR集群的配置是否满足要求，然后进行元数据采集配置。

1. 在EMR控制台中，检查EMR集群的**hive.metastore.pre.event.listeners**和**hive.exec.post.hooks**配置是生效的。



2. 在DataWorks的数据地图页面进行元数据采集配置，操作详情请参见收集和查看元数据。

常见问题：作业提交失败

- 问题现象

  在DataWorks上提交EMR作业时失败，报错如下。

> **说明** 提交作业失败，不是作业运行失败。

```
>>> [2021-07-29 07:49:05][INFO   ][InteractiveJobSubmitter]: Fail to submit job: ### Erro
rCode: E00007
java.io.IOException: Request Failed, code=500, message=
        at com.aliyun.emr.flow.agent.client.protocol.impl.FlowAgentClientRestProtocolImpl
.exchange(FlowAgentClientRestProtocolImpl.java:146)
```

- 可能原因

  DataWorks和EMR对接时，需使用EMR集群的FlowAgent组件，如果提交作业失败出现上述报错时，很有可能是FlowAgent这个组件出现了问题，您可以在EMR控制台重启这个组件快速解决问题。

- 解决方案

  您可以重启FlowAgent组件来解决问题。

  i. 进入FlowAgent页面。

     默认情况下FlowAgent这个组件是隐藏的，您无法直接通过EMR控制台入口进入FlowAgent组件页面。所以您需要先进入任意某个组件的页面，然后手动修改页面链接URL进入FlowAgent页面。

     以进入HDFS组件页面为例，进入后，HDFS组件页面的URL为：`https://emr.console.aliyun.com/#/cn-hangzhou/cluster/C-XXXXXXXXXXXXXX/service/HDFS`，您需要将链接最后的组件名手动修改为EMRFLOW，即，FlowAgent组件的页面链接为`https://emr.console.aliyun.com/#/cn-hangzhou/cluster/C-XXXXXXXXXXXXXX/service/EMRFLOW`。

  ii. 重启FlowAgent组件。

     在右上角单击**操作 > 重启All Components**，重启组件。

6.4.2. 准备工作：绑定EMR引擎

DataWorks支持基于EMR（E-MapReduce）计算引擎创建Hive、MR、Presto和Spark SQL等节点，实现EMR任务工作流的配置、定时调度和元数据管理等功能，帮助EMR用户更好地生产数据。在创建EMR节点进行数据开发等操作前，您需要先绑定EMR引擎到DataWorks的工作空间，本文为您介绍绑定EMR引擎的操作详情。

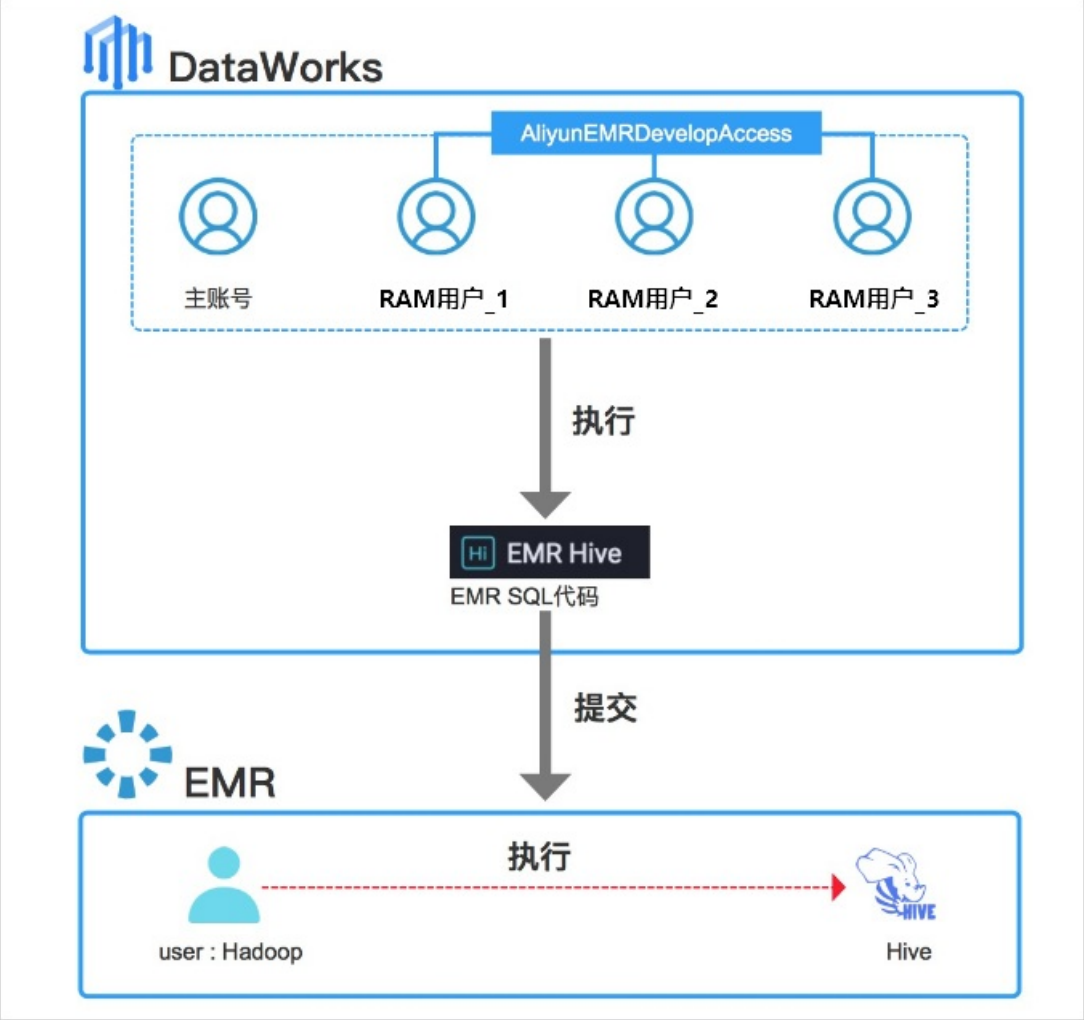
DataWorks为您提供**快捷模式**和**安全模式**两种绑定EMR引擎的模式，以实现不同类型的企业、安全要求场景。您可以基于**快捷模式**快速开展各类数据的工作，并可以基于**安全模式**实现更具安全性的数据权限管理。

快捷模式

当EMR计算引擎的绑定模式为**快捷模式**时，阿里云主账号或RAM用户在DataWorks运行代码或自动调度任务，都只是下发代码至EMR集群，实际运行的身份为集群内的Hadoop用户。

> **注意**
>
> - 该Hadoop用户拥有Hadoop集群的所有权限，请谨慎授权。
> - 在**快捷模式**下，为保证工作空间成员可以在DataStudio内正常运行EMR类的任务，请确保开发、管理员等相关角色拥有AliyunEMRDevelopAccess权限策略。
>   - 如果您使用阿里云主账号运行任务，该账号天然拥有AliyunEMRDevelopAccess权限策略。
>   - 如果您使用RAM用户运行任务，则需要授予该用户AliyunEMRDevelopAccess权限策略，详情请参见为RAM用户授权。

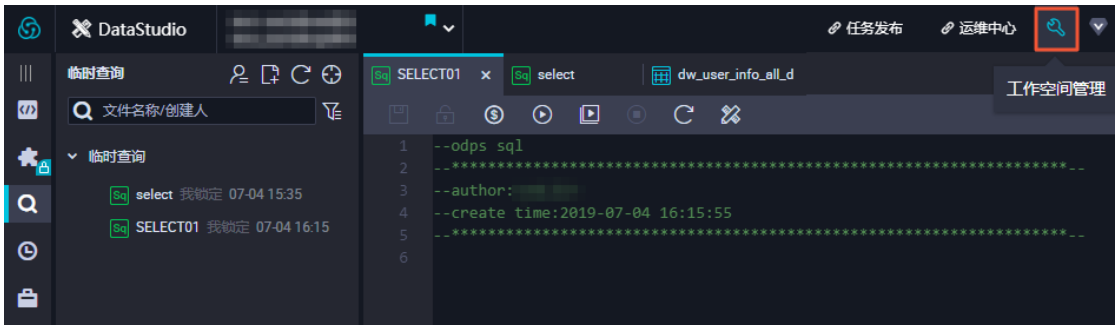
快捷模式适用于对任务执行者数据权限无强管控要求的工作空间。



以**快捷模式**新增EMR集群：

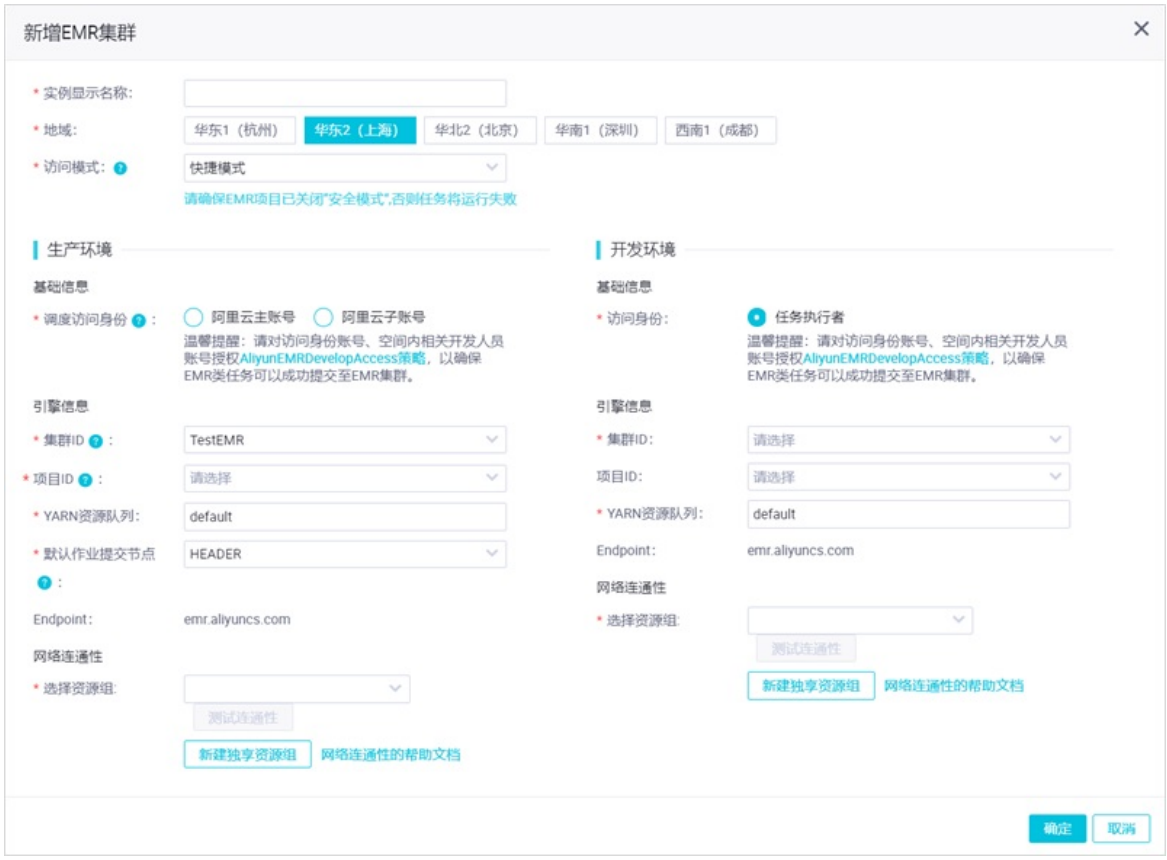
1. 进入**工作空间配置**页面：
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**工作空间列表**。
   iii. 选择工作空间所在地域后，单击相应工作空间后的**工作空间配置**。在**工作空间配置**对话框中，单击**更多设置**，进入**工作空间配置**页面。

您也可以单击相应工作空间后的**进入数据开发**。在**数据开发**页面，单击右上方的图标，进入**工作空间配置**页面。



2. 在**计算引擎信息**区域，单击**E-MapReduce**。
3. 在**E-MapReduce**页签下，单击**增加实例**。
4. 在**新增EMR集群**对话框中，配置各项参数。

DataWorks简单模式和标准模式工作空间的配置不同。DataWorks标准模式的工作空间需要分别配置生产环境和开发环境的参数。



参数	描述
实例显示名称	自定义实例的名称。
地域	当前工作空间所在的地域，不可以修改。
访问模式	从下拉列表中选择**快捷模式**。
调度访问身份	当提交任务至调度系统后，DataWorks调度系统自动运行任务时，提交代码至EMR集群的身份。您可以选择**阿里云主账号**和**阿里云子账号**。 **说明** ○ 仅生产环境涉及配置该参数。 ○ 在**快捷模式**下，为保证工作空间成员可以在DataStudio内正常运行EMR类的任务，请确保开发、管理员等相关角色拥有AliyunEMRDevelopAccess权限策略。 ■ 如果您使用阿里云主账号运行任务，该账号天然拥有AliyunEMRDevelopAccess权限策略。 ■ 如果您使用RAM用户运行任务，则需要授予该用户AliyunEMRDevelopAccess权限策略，详情请参见为RAM用户授权。

参数	描述
访问身份	在开发环境运行任务时，提交代码至EMR引擎所使用的身份。此处默认为**任务执行者**。 **说明** • 仅DataWorks标准模式的工作空间会显示该参数，并且仅开发环境涉及配置该参数。 • **任务执行者**可以为阿里云主账号或RAM用户。 在**快捷模式**下，为保证工作空间成员可以在DataStudio内正常运行EMR类的任务，请确保开发、管理员等相关角色拥有AliyunEMRDevelopAccess权限策略。 ▪ 如果您使用阿里云主账号运行任务，该账号天然拥有AliyunEMRDevelopAccess权限策略。 ▪ 如果您使用RAM用户运行任务，则需要授予该用户AliyunEMRDevelopAccess权限策略，详情请参见为RAM用户授权。
集群ID	从下拉列表中选择调度访问身份账户所在的EMR集群，作为任务的运行环境。
项目ID	从下拉列表中选择调度访问身份账户所在的EMR项目，作为任务的运行环境。 **说明** 如果EMR项目开启**访问模式**为**安全模式**，则无法被选择。
YARN资源队列	当前集群下的队列名称。如果无特殊需求，请输入*default*。
默认作业提交节点	选择EMR作业通过什么节点提交至EMR集群中。如果您的EMR集群关联了Gateway集群，此处可选择关联的Gateway集群，其他场景可选择默认的**HEADER**。 **说明** 此处选择**默认作业提交节点**为Gateway集群后，后续此EMR引擎空间下的所有EMR节点默认通过Gateway集群提交作业，如果某个节点不需要通过Gateway集群提交作业，您可以在节点的高级配置中，手动添加并设置**USE_GATEWAY**参数为False，详情可参见各EMR节点的高级参数配置帮助文档内容。
Endpoint	EMR的Endpoint，不可以修改。

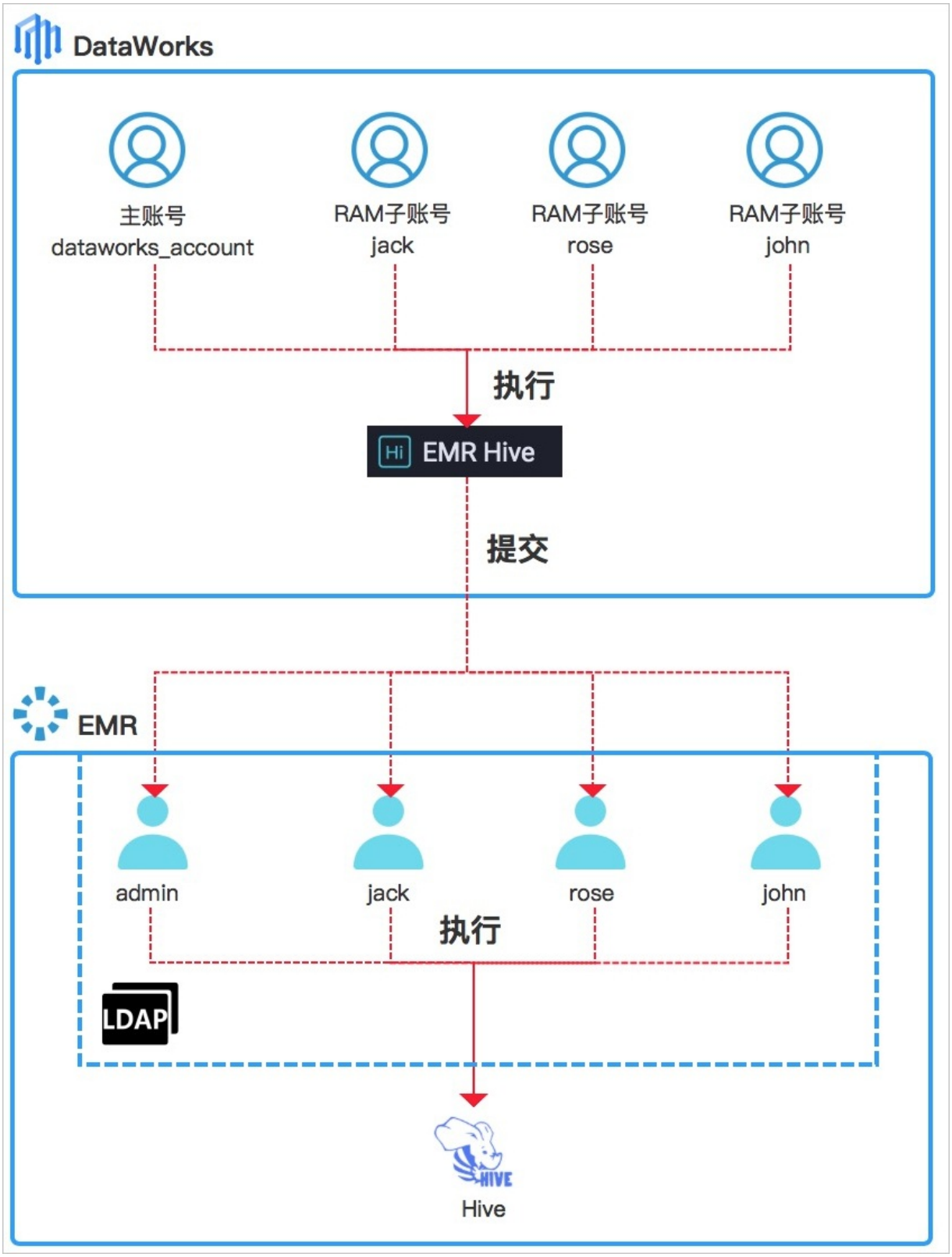
5. 选择资源组。
   i. 选择已与当前DataWorks工作空间配置网络连通性的独享调度资源组。如果您没有可用的独享调度资源组，则需要新建。新建独享调度资源组并配置网络连通性，详情请参见新增和使用独享调度资源组。
   ii. 单击**测试连通性**，验证独享调度资源组与E-MapReduce引擎的网络连通性。
6. 单击**确定**。

安全模式

当EMR计算引擎的绑定模式为**安全模式**时，阿里云主账号或RAM用户在下发代码的同时，EMR集群内会匹配对应的同名用户来运行任务。管理者可以使用EMR集群内的Ranger组件对每个用户进行权限管控，最终实现不同阿里云主账号、任务责任人或RAM用户在DataWorks上运行EMR任务时，拥有对应不同数据权限的目的，进行更安全的数据权限隔离。

> **说明**
>
> 在**安全模式**下，为保证工作空间成员可以在DataStudio内正常运行EMR类的任务，请确保开发、管理员等相关角色已被加入EMR集群的LDAP内，并拥有AliyunEMRDevelopAccess或AliyunEMRFullAccess权限策略，以及其他相关的数据权限，以避免任务执行失败。
>
> - 如果您使用阿里云主账号运行任务，该账号天然存在目标EMR集群的LDAP中，并拥有AliyunEMRDevelopAccess及AliyunEMRFullAccess权限策略。
> - 如果您使用RAM用户运行任务，则需要添加该用户至目标EMR集群的LDAP中，详情请参见下文中的*导入阿里云RAM用户至EMR LDAP*操作。同时，需要授予该用户AliyunEMRDevelopAccess或AliyunEMRFullAccess权限策略，详情请参见为RAM用户授权。

安全模式适用于对任务执行者有数据权限管控隔离要求的工作空间。



以**安全模式**新增EMR集群：

1. 开启EMR项目的安全模式。
   i. 登录EMR管理控制台。
   ii. 在顶部菜单栏，单击**数据开发**。

iii. 在**项目列表**页面，单击相应项目后的**作业编辑**。

iv. 在顶部菜单栏，单击**项目管理**。

v. 在左侧导航栏，单击**通用配置**，开启**安全模式**。

2. 导入阿里云RAM用户至EMR LDAP。

   i. 在EMR管理控制台的顶部菜单栏，单击**集群管理**。

   ii. 单击相应集群后的**详情**。

   iii. 在左侧导航栏，单击**用户管理**。

   iv. 在**用户管理**页面，单击**添加用户**。

   v. 在**添加用户**对话框中，配置各项参数，添加涉及的阿里云RAM用户至EMR集群LDAP中。

   建议添加如下用途的阿里云RAM用户至EMR LDAP：

   - 有可能在DataStudio创建EMR类任务、测试运行的相关人员。
   - 有可能在DataStudio创建、提交和发布EMR类任务的相关人员。

   vi. 单击**确定**。

3. 配置EMR Ranger，并对阿里云账号对应的Hadoop用户进行权限管控。详情请参见Ranger Usersync集成LDAP和组件集成Ranger。

4. 增加EMR实例。

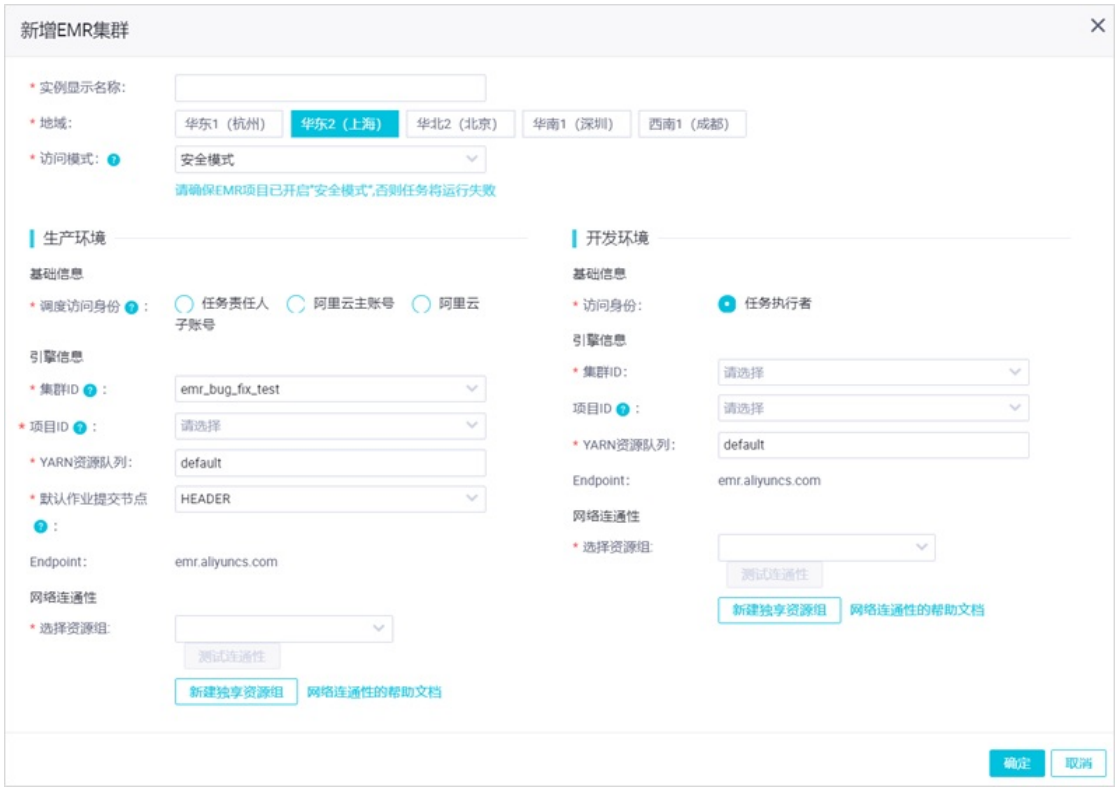
   i. 进入DataWorks工作空间配置页面。

   ii. 在**计算引擎信息**区域，单击**E-MapReduce**。

   iii. 在**E-MapReduce**页签下，单击**增加实例**。

   iv. 在**新增EMR集群**对话框中，配置各项参数。

DataWorks简单模式和标准模式工作空间的配置不同。DataWorks标准模式的工作空间需要分别配置生产环境和开发环境的参数。



参数	描述
实例显示名称	自定义实例的名称。
地域	当前工作空间所在的地域。
访问模式	从下拉列表中选择**安全模式**，并在**请注意**对话框中，单击**确定**。 **说明** 一个引擎实例在同一时间仅能使用一种访问模式，修改访问模式会引起访问身份权限变化，请谨慎操作。

参数	描述
调度访问身份	当任务被提交、发布至生产环境后，DataWorks调度系统自动运行任务时，提交代码至EMR集群的身份，同时该身份对应的Hadoop用户将实际运行代码。 您可以选择**任务责任人**、**阿里云主账号**和**阿里云子账号**： ■ **任务责任人**：任务调度运行时，以责任人身份提交并运行代码。该项为安全模式的核心功能，用于隔离用户间的数据权限。**任务责任人**可以为阿里云主账号或RAM用户。 ■ **阿里云主账号**：任务调度运行时，以阿里云主账号作为调度身份，提交代码至EMR集群的身份。 ■ **阿里云子账号**：任务调度运行时，以某个单一的阿里云RAM用户作为调度身份，提交代码至EMR集群的身份。 说明 ■ 仅生产环境涉及配置该参数。 ■ 如果您使用阿里云主账号运行任务，该账号天然存在目标EMR集群的LDAP中，并拥有AliyunEMRDevelopAccess及AliyunEMRFullAccess权限策略。 ■ 如果您使用RAM用户运行任务，则需要添加该用户至目标EMR集群的LDAP中，详情请参见下文中的*导入阿里云RAM用户至EMR LDAP*操作。同时，需要授予该用户AliyunEMRDevelopAccess或AliyunEMRFullAccess权限策略，详情请参见为RAM用户授权。
访问身份	在开发环境运行任务时，提交代码至EMR引擎所使用的身份。此处默认为**任务执行者**，同时执行者对应的Hadoop用户将实际运行代码。 说明 ■ 仅DataWorks标准模式的工作空间显示该参数，并且仅开发环境涉及配置该参数。 ■ 请确保该类执行者已被加入至EMR集群LDAP中，并被授予AliyunEMRDevelopAccess或AliyunEMRFulAccess权限策略，以及其他相关的数据权限，以避免任务执行失败。**任务执行者**可以为阿里云主账号或RAM用户。 ■ 如果您使用阿里云主账号运行任务，该账号天然存在目标EMR集群的LDAP中，并拥有AliyunEMRDevelopAccess及AliyunEMRFullAccess权限策略。 ■ 如果您使用RAM用户运行任务，则需要添加该用户至目标EMR集群的LDAP中，详情请参见下文中的*导入阿里云RAM用户至EMR LDAP*操作。同时，需要授予该用户AliyunEMRDevelopAccess或AliyunEMRFullAccess权限策略，详情请参见为RAM用户授权。

参数	描述
集群ID	从下拉列表中选择已开启安全模式的EMR项目调度访问身份账户所在的EMR集群，作为任务的运行环境。
项目ID	从下拉列表中选择已开启安全模式的EMR项目。 **说明** 如果EMR项目未开启**访问模式**为**安全模式**，则无法被选择。
YARN资源队列	当前集群下的队列名称。如果无特殊需求，请输入*default*。
默认作业提交节点	选择EMR作业通过什么节点提交至EMR集群中。如果您的EMR集群关联了Gateway集群，此处可选择关联的Gateway集群，其他场景可选择默认的**HEADER**。 **说明** 此处选择**默认作业提交节点**为Gateway集群后，后续此EMR引擎空间下的所有EMR节点默认通过Gateway集群提交作业，如果某个节点不需要通过Gateway集群提交作业，您可以在节点的高级配置中，手动添加并设置**USE_GATEWAY**参数为False，详情可参见各EMR节点的高级参数配置帮助文档内容。
Endpoint	EMR的Endpoint，不可以修改。

v. 选择资源组。

a. 选择已与当前DataWorks工作空间配置网络连通性的独享调度资源组。如果您没有可用的独享调度资源组，则需要新建。新建独享调度资源组并配置网络连通性，详情请参见新增和使用独享调度资源组。

b. 单击**测试连通性**，验证独享调度资源组与E-MapReduce引擎的网络连通性。

vi. 单击**确定**。

5. 配置访问身份的映射关系。

成功添加**安全模式**的E-MapReduce引擎实例后，后续实际执行实例任务时，使用的是EMR集群配置的访问身份对应的OPEN LDAP集群账号，您需要进入**EMR集群配置**页面，配置访问身份的映射关系。

i. 成功添加E-MapReduce引擎后，在弹出的**请注意**对话框，单击**去配置开发环境**及**去配置生产环境**，配置访问身份的映射关系。

ii. 在**EMR集群配置**页面，单击已绑定的EMR集群右上角的**编辑**。

iii. 在**编辑EMR集群配置**对话框，配置引擎权限映射。

您可以使用如下两种方式配置引擎权限映射关系：

- 引用已创建的映射关系：您可以直接引用当前工作空间中已创建的权限映射关系。
- 创建新的权限映射关系：在**配置引擎权限映射**区域，选择需要配置映射关系的**云账号**及**LDAP账号**，并输入**LDAP账号**账号的密码。

> **说明**
>
> - 阿里云主账号或拥有AliyunEMRFullAccess权限策略的RAM用户可以为本工作空间所有成员配置身份映射，其余工作空间成员仅可以为自己配置身份映射。
> - 您可以添加多个**云账号**与**LDAP账号**的映射关系。DataWorks支持多个**云账号**映射至同一个**LDAP账号**

iv. 单击**确定**，完成创建。

6.4.3. 创建EMR Presto节点

您可以通过创建EMR（E-MapReduce）Presto节点，进行大规模结构化和非结构化数据的交互式分析查询。

前提条件

EMR引擎类型包括新版数据湖（DataLake）及数据湖（Hadoop），不同类型引擎创建节点前需执行的准备工作不同。您需要根据实际情况完成EMR侧及DataWorks侧的准备工作，详情请参见准备工作：EMR引擎配置、准备工作：DataWorks配置。

使用限制

- 阿里政务云和金融云平台不支持创建EMR Presto节点。
- 仅支持使用独享调度资源组运行该类型任务。

- DataWorks目前已不支持新绑定Hadoop类型的集群，但您之前已经绑定的Hadoop集群仍然可以继续使用。

操作步骤

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**数据开发**。
2. 创建**业务流程**。

   如果您已有**业务流程**，则可以忽略该步骤。
   i. 鼠标悬停至+新建图标，选择**新建业务流程**。
   ii. 在**新建业务流程**对话框，输入**业务名称**。
   iii. 单击**新建**。
3. 创建**EMR Presto**节点。
   i. 鼠标悬停至+新建图标，选择**新建节点 > EMR > EMR Presto**。

   您也可以找到相应的业务流程，右键单击业务流程，选择**新建节点 > EMR > EMR Presto**。
   ii. 在**新建节点**对话框中，输入**名称**，并选择**引擎实例**、**节点类型**及**路径**。

   > **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**，进入EMR Presto节点编辑页面。
4. 使用EMR Presto节点进行数据开发。
   i. 选择资源组。

   在工具栏单击图标，在**参数**对话框选择已创建的调度资源组。

   > **说明**
   > - 访问公共网络或VPC网络环境的数据源需要使用与数据源测试连通性成功的调度资源组。详情请参见配置资源组与网络连通。
   > - 如果您后续执行任务需要修改使用的资源组，也可以在此处选择需要更换的调度资源组。

ii. 使用SQL语句创建任务。

在SQL编辑区域输入任务代码，示例如下。

```
show tables;
select '${var}'; //可以结合调度参数使用。
select * from userinfo ;
```

> **说明**
>
> - SQL语句最大不能超过130KB。
> - 使用EMR Presto节点查询数据时，返回的查询结果最大支持10000条数据，并且数据总量不能超过10M。
> - 如果您的工作空间绑定多个EMR引擎，则需要根据业务需求选择合适的引擎。如果仅绑定一个EMR引擎，则无需选择。

如果您需要修改代码中的参数赋值，请单击界面上方工具栏的**高级运行**。参数赋值逻辑详情请参见运行，高级运行和开发环境冒烟测试赋值逻辑有什么区别。

> **说明**
>
> - 调度参数使用详情，请参考调度参数概述。
> - Presto作业配置，请参考Presto SQL作业配置。

iii. 保存并运行SQL语句。

在工具栏，单击图标，保存编写的SQL语句，单击图标，运行创建的SQL任务。

5. 编辑**高级设置**。

不同类型EMR集群涉及配置的高级参数有差异，具体如下表。

集群类型	高级参数
新版数据湖（DataLake）	○ “memory”：内存，默认为2048MB（用于设置启动器Launcher的内存配额）。 ○ “priority”：优先级，默认为1。 ○ “FLOW_SKIP_SQL_ANALYZE”：SQL语句执行方式。取值如下： ■ `true`：表示每次执行多条SQL语句。 ■ `false`：表示每次执行一条SQL语句。 ○ “DATAWORKS_SESSION_DISABLE”：适用于开发环境直接测试运行场景。取值如下： ■ `true`：表示每次运行SQL语句都会新建一个JDBC Connection。 ■ `false`：表示用户在一个节点里运行不同的SQL语句时会复用同一个JDBC Connection。 默认值为 `false`。

集群类型	高级参数
数据湖（Hadoop）	○ "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx"，设置Spark任务的运行参数，多个参数在该Key中追加。 ○ “queue”：提交作业的调度队列，默认为default队列。 ○ “vcores”： 虚拟核数，默认为1。 ○ “memory”：内存，默认为2048MB（用于设置启动器Launcher的内存配额）。 ○ “priority”：优先级，默认为1。 ○ “FLOW_SKIP_SQL_ANALYZE”：SQL语句执行方式。取值如下： ■ `true`：表示每次执行多条SQL语句。 ■ `false`：表示每次执行一条SQL语句。 ○ “USE_GATEWAY”：设置本节点提交作业时，是否通过Gateway集群提交。取值如下： ■ `true`：通过Gateway集群提交。 ■ `false`：不通过Gateway集群提交，默认提交到header节点。 **说明** 如果本节点所在的集群未关联Gateway集群，此处手动设置参数取值为 `true` 时，后续提交EMR作业时会失败。

6. 任务调度配置。

   如果您需要周期性执行创建的节点任务，可以单击节点编辑页面右侧的**调度配置**，根据业务需求配置该节点任务的调度信息：

   - 配置任务调度的基本信息，详情请参见配置基础属性。
   - 配置时间调度周期、重跑属性和上下游依赖关系，详情请参见时间属性配置说明及配置同周期调度依赖。

     说明 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   - 配置资源属性，详情请参见配置资源属性。访问公网或VPC网络时，请选择与目标节点网络连通的调度资源组作为周期调度任务使用的资源组。详情请参见配置资源组与网络连通。

7. 提交并发布节点任务。

   i. 单击工具栏中的图标，保存节点。

   ii. 单击工具栏中的图标，提交节点任务。

   iii. 在**提交新版本**对话框中，输入**变更描述**。

   iv. 单击**确定**。

   如果您使用的是标准模式的工作空间，任务提交成功后，需要将任务发布至生产环境进行发布。请单击顶部菜单栏左侧的**任务发布**。具体操作请参见发布任务。

8. 查看周期调度任务。

   i. 单击编辑界面右上角的**运维**，进入生产环境运维中心。

ii. 查看运行的周期调度任务，详情请参见查看并管理周期任务。

如果您需要查看更多周期调度任务详情，可单击顶部菜单栏的**运维中心**，详情请参见运维中心概述。

6.4.4. 创建EMR Hive节点

您可以创建EMR（E-MapReduce） HIVE节点，通过类SQL语句协助读写、管理存储在分布式存储系统上的大数据集的数据仓库，完成海量日志数据的分析和开发工作。

前提条件

EMR引擎类型包括新版数据湖（DataLake）及数据湖（Hadoop），不同类型引擎创建节点前需执行的准备工作不同。您需要根据实际情况完成EMR侧及DataWorks侧的准备工作，详情请参见准备工作：EMR引擎配置、准备工作：DataWorks配置。

使用限制

- 仅支持使用独享调度资源组运行该类型任务。
- DataWorks目前已不支持新绑定Hadoop类型的集群，但您之前已经绑定的Hadoop集群仍然可以继续使用。

操作步骤

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**数据开发**。
2. 创建**业务流程**。

   如果您已有**业务流程**，则可以忽略该步骤。
   i. 鼠标悬停至**+新建**图标，选择**新建业务流程**。
   ii. 在**新建业务流程**对话框，输入**业务名称**。
   iii. 单击**新建**。
3. 创建**EMR Hive**节点。
   i. 鼠标悬停至**+新建**图标，选择**新建节点 > EMR > EMR Hive**。

   您也可以找到相应的业务流程，右键单击业务流程，选择**新建节点 > EMR > EMR Hive**。
   ii. 在**新建节点**对话框中，输入**名称**，并选择**引擎实例**、**节点类型**及**路径**。

   > **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**，进入EMR Hive节点编辑页面。
4. 使用EMR Hive节点进行数据开发。

i. 选择资源组。

在工具栏单击图标，在**参数**对话框选择已创建的调度资源组。

> **说明**
> - 访问公共网络或VPC网络环境的数据源需要使用与数据源测试连通性成功的调度资源组。详情请参见配置资源组与网络连通。
> - 如果您后续执行任务需要修改使用的资源组，也可以在此处选择需要更换的调度资源组。

ii. 使用SQL语句创建任务。

在SQL编辑区域输入任务代码，示例如下。

```
show tables;
select '${var}'; //可以结合调度参数使用。
select * from userinfo ;
```

> **说明**
> - SQL语句最大不能超过130KB。
> - 使用EMR Hive节点查询数据时，返回的查询结果最大支持10000条数据，并且数据总量不能超过10M。
> - 如果您的工作空间绑定多个EMR引擎，则需要根据业务需求选择合适的引擎。如果仅绑定一个EMR引擎，则无需选择。

如果您需要修改代码中的参数赋值，请单击界面上方工具栏的**高级运行**。参数赋值逻辑详情请参见运行，高级运行和开发环境冒烟测试赋值逻辑有什么区别。

> **说明**
> - 调度参数使用详情，请参考调度参数概述。
> - Hive作业配置，请参考Hive SQL作业配置。

iii. 保存并运行SQL语句。

在工具栏，单击图标，保存编写的SQL语句，单击图标，运行创建的SQL任务。

5. 编辑**高级设置**。

不同类型EMR集群涉及配置的高级参数有差异，具体如下表。

集群类型	高级参数

集群类型	高级参数
新版数据湖（DataLake）	○ “queue”：提交作业的调度队列，默认为default队列。 ○ “priority”：优先级，默认为1。 ○ “FLOW_SKIP_SQL_ANALYZE”：SQL语句执行方式。取值如下： ■ `true`：表示每次执行多条SQL语句。 ■ `false`：表示每次执行一条SQL语句。 ○ “DATAWORKS_SESSION_DISABLE”：适用于开发环境直接测试运行场景。取值如下： ■ `true`：表示每次运行SQL语句都会新建一个JDBC Connection。 ■ `false`：表示用户在一个节点里运行不同的SQL语句时会复用同一个JDBC Connection。 默认值为 `false`。 **说明** 您也可以直接在高级配置里追加自定义Hive Connection参数。
数据湖（Hadoop）	○ "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx"，设置Spark 任务运行参数，多个参数在该Key中追加。 ○ “queue”：提交作业的调度队列，默认为default队列。 ○ “vcores”： 虚拟核数，默认为1。 ○ “memory”：内存，默认为2048MB（用于设置启动器Launcher的内存配额）。 ○ “priority”：优先级，默认为1。 ○ “FLOW_SKIP_SQL_ANALYZE”：SQL语句执行方式。取值如下： ■ `true`：表示每次执行多条SQL语句。 ■ `false`：表示每次执行一条SQL语句。 ○ “USE_GATEWAY”：设置本节点提交作业时，是否通过Gateway集群提交。取值如下： ■ `true`：通过Gateway集群提交。 ■ `false`：不通过Gateway集群提交，默认提交到header节点。 **说明** 如果本节点所在的集群未关联Gateway集群，此处手动设置参数取值为 `true` 时，后续提交EMR作业时会失败。

6. 任务调度配置。

   如果您需要周期性执行创建的节点任务，可以单击节点编辑页面右侧的**调度配置**，根据业务需求配置该节点任务的调度信息：

   ○ 配置任务调度的基本信息，详情请参见配置基础属性。

   ○ 配置时间调度周期、重跑属性和上下游依赖关系，详情请参见时间属性配置说明及配置同周期调度依赖。

> **说明** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

- 配置资源属性，详情请参见配置资源属性。访问公网或VPC网络时，请选择与目标节点网络连通的调度资源组作为周期调度任务使用的资源组。详情请参见配置资源组与网络连通。

7. 提交并发布节点任务。
   i. 单击工具栏中的图标，保存节点。
   ii. 单击工具栏中的图标，提交节点任务。
   iii. 在**提交新版本**对话框中，输入**变更描述**。
   iv. 单击**确定**。

   如果您使用的是标准模式的工作空间，任务提交成功后，需要将任务发布至生产环境进行发布。请单击顶部菜单栏左侧的**任务发布**。具体操作请参见发布任务。
8. 查看周期调度任务。
   i. 单击编辑界面右上角的**运维**，进入生产环境运维中心。
   ii. 查看运行的周期调度任务，详情请参见查看并管理周期任务。

   如果您需要查看更多周期调度任务详情，可单击顶部菜单栏的**运维中心**，详情请参见运维中心概述。

6.4.5. 创建并使用EMR MR节点

您可以通过创建EMR（E-MapReduce） MR节点，将一个大规模数据集拆分为多个Map任务并行处理，实现大规模数据集的并行运算。本文为您介绍如何创建EMR MR节点，并以使用MR节点实现从OSS中读取文本，统计其中单词的数量为例，为您展示EMR MR节点的作业开发流程。

前提条件

-
-
-
-
- 使用EMR MR节点进行作业开发时，如果需要引用开源代码资源，您需先将开源代码作为资源上传至**EMR JAR资源**节点中，详情请参见创建和使用EMR资源。
- 使用EMR MR节点进行作业开发时，如果需要引用自定义函数时，您需要先将自定义函数作为资源上传至**EMR JAR资源节**点中，新建注册此函数，详情请参见注册EMR函数。
- 如果您使用本文的作业开发示例执行相关作业流程，则还需要完成如下操作：
  - 已创建OSS的存储空间Bucket，详情请参见创建存储空间。
  - 已创建好IDEA项目。

背景信息

本文以使用MR节点实现从OSS中读取文本，统计其中单词的数量为例，为您展示EMR MR节点的作业开发流程。涉及的文件名称、Bucket名称及路径等信息，在实际使用中，您需要替换为实际使用的相关信息。

准备初始数据及JAR资源包

1. 准备初始数据。

创建*input01.txt*文件，文件内容如下。

```
hadoop emr hadoop dw
hive hadoop
dw emr
```

2. 创建初始数据及JAR资源的存放目录。
   i. 登录OSS管理控制台。
   ii. 单击左侧导航栏的**Bucket列表**
   iii. 单击目标Bucket名称，进入**文件管理**页面。
   本文示例使用的Bucket为*onaliyun-bucket-2*。
   iv. 单击**新建目录**，创建初始数据及JAR资源的存放目录。
   - 配置**目录名**为*emr/datas/wordcount02/inputs*，创建初始数据的存放目录。
   - 配置**目录名**为*emr/jars*，创建JAR资源的存放目录。

   v. 上传初始数据文件至初始数据的存放目录。
      a. 进入*/emr/datas/wordcount02/inputs*路径。
      b. 单击**上传文件**
      c. 在**待上传文件**区域单击**扫描文件**，添加*input01.txt*文件至Bucket。

      d. 单击**上传文件**

3. 使用MapReduce读取OSS文件并生成JAR包。
   i. 打开已创建的IDEA项目，添加pom依赖。

```
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-mapreduce-client-common</artifactId>
    <version>2.8.5</version> <!--因为EMR-MR用的是2.8.5-->
</dependency>
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-common</artifactId>
    <version>2.8.5</version>
</dependency>
```

ii. 在MapReduce中读写OSS文件，需要配置如下参数。

```
conf.set("fs.oss.accessKeyId", "${accessKeyId}");
conf.set("fs.oss.accessKeySecret", "${accessKeySecret}");
conf.set("fs.oss.endpoint","${endpoint}");
```

参数说明如下：

- `${accessKeyId}`：阿里云账号的AccessKey ID。
- `${accessKeySecret}`：阿里云账号的AccessKey Secret。
- `${endpoint}`：OSS对外服务的访问域名。由您集群所在的地域决定，对应的OSS也需要是在集群对应的地域，详情请参见访问域名和数据中心

以Java代码为例，修改Hadoop官网WordCount示例，即在代码中添加AccessKey ID和AccessKey Secret的配置，以便作业有权限访问OSS文件。

```
package cn.apache.hadoop.onaliyun.examples;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class EmrWordCount {
    public static class TokenizerMapper
            extends Mapper<Object, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        public void map(Object key, Text value, Context context
        ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }
    public static class IntSumReducer
            extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();
        public void reduce(Text key, Iterable<IntWritable> values,
                           Context context
        ) throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
```

```
        }
    }
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs(
);
        if (otherArgs.length < 2) {
            System.err.println("Usage: wordcount <in> [<in>...] <out>");
            System.exit(2);
        }
        conf.set("fs.oss.accessKeyId", "${accessKeyId}"); //
        conf.set("fs.oss.accessKeySecret", "${accessKeySecret}"); //
        conf.set("fs.oss.endpoint", "${endpoint}"); //
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(EmrWordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        for (int i = 0; i < otherArgs.length - 1; ++i) {
            FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
        }
        FileOutputFormat.setOutputPath(job,
                new Path(otherArgs[otherArgs.length - 1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

iii. 编辑完上述Java代码后将该代码生成JAR包。示例生成的JAR包为*onaliyun_mr_wordcount-1.0-SNAPSHOT.jar.*。

创建并使用EMR MR节点

本文以使用MR节点实现从OSS中读取文本，统计其中单词的数量为例，为您展示EMR MR节点的作业开发流程。

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. 鼠标悬停至**+新建**图标，单击**EMR > EMR MR**。

   您也可以找到相应的业务流程，右键单击**EMR**，选择**新建 > EMR MR**。
3. 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。
4. 单击**提交**。

5. 创建EMR JAR资源。

   创建EMR JAR资源，详情请参见创建和使用EMR资源。示例将上述步骤生成的JAR包存储在JAR资源的存放目录*emr/jars*下。首次使用需要进行**一键授权**。

6. 引用EMR JAR资源。

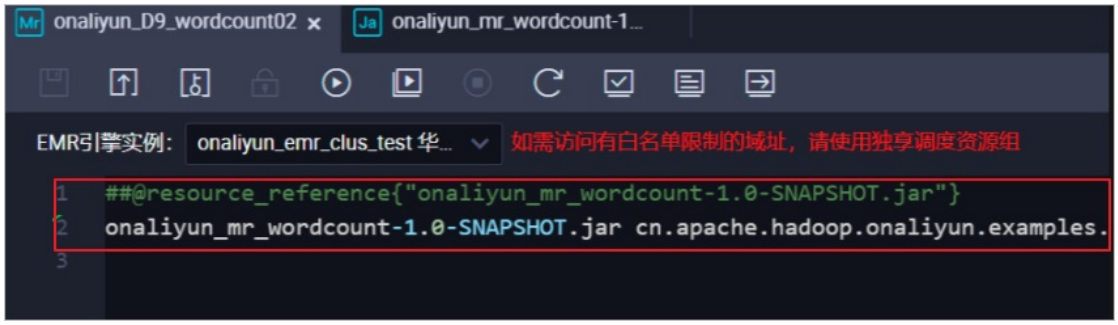
   i. 打开创建的**EMR MR**节点，停留在代码编辑页面。

   ii. 在**EMR > 资源**节点下，找到待引用资源（示例为*onaliyun_mr_wordcount-1.0-SNAPSHOT.jar.*），右键选择**引用资源**。

iii. 选择引用后，当**EMR MR**节点的代码编辑页面出现如下引用成功提示时，表明已成功引用代码资源。此时，需要执行下述命令。

如下命令涉及的资源包、Bucket名称、路径信息等为本文示例的内容，使用时，您需要替换为实际使用的信息。

```
##@resource_reference{"onaliyun_mr_wordcount-1.0-SNAPSHOT.jar"}
onaliyun_mr_wordcount-1.0-SNAPSHOT.jar cn.apache.hadoop.onaliyun.examples.EmrWordCo
unt oss://onaliyun-bucket-2/emr/datas/wordcount02/inputs oss://onaliyun-bucket-2/em
r/datas/wordcount02/outputs
```



7.
8.
9. 保存并提交节点。

> **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

i. 单击工具栏中的图标，保存节点。

ii. 单击工具栏中的图标。

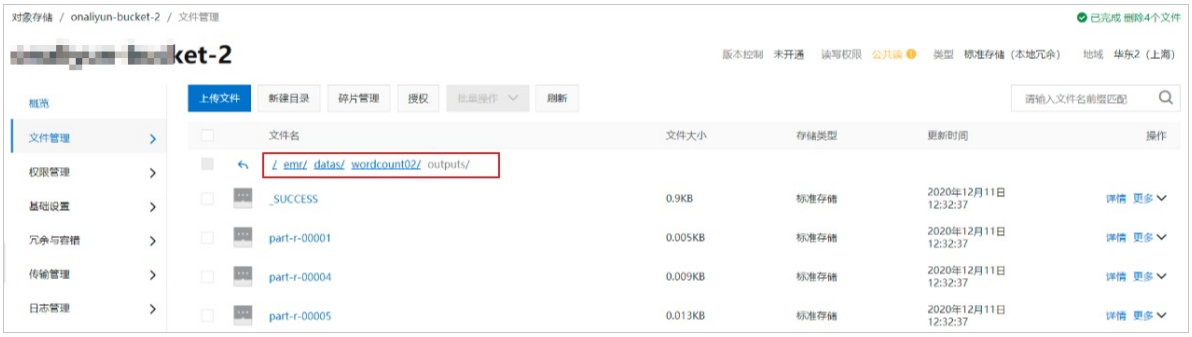
iii. 在**提交新版本**对话框中，输入**变更描述**。

iv. 单击**确认**。

如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

查看结果

- 登录OSS管理控制台，您可以在目标Bucket的初始数据存放目录下查看写入结果。示例路径为*emr/datas/wordcount02/inputs*。



- 在DataWorks读取统计结果。

i. 新建EMR Hive节点，详情请参见创建EMR Hive节点。

ii. 在EMR Hive节点中创建挂载在OSS上的Hive外表，读取表数据。代码示例如下。

```
CREATE EXTERNAL TABLE IF NOT EXISTS wordcount02_result_tb
(
    `word` STRING COMMENT '单词',
    `cout` STRING COMMENT '计数'
)
ROW FORMAT delimited fields terminated by '\t'
location 'oss://onaliyun-bucket-2/emr/datas/wordcount02/outputs/';
SELECT * FROM wordcount02_result_tb;
```

运行结果如下图。

6.4.6. 创建EMR Spark SQL节点

您可以通过创建EMR（E-MapReduce） Spark SQL节点，实现分布式SQL查询引擎处理结构化数据，提高作业的执行效率。

前提条件

操作步骤

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. 鼠标悬停至**+新建**图标，单击**EMR > EMR Spark SQL**。

   您也可以找到相应的业务流程，右键单击**EMR**，选择**新建 > EMR Spark SQL**。
3. 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。
4. 单击**提交**。
5. 在节点编辑页面，输入代码。

   SQL语句示例如下。

```
show tables;
CREATE TABLE IF NOT EXISTS userinfo_new_${var} (
ip STRING COMMENT'ip地址',
uid STRING COMMENT'用户ID'
)PARTITIONED BY(
dt STRING
); //可以结合调度参数使用。
```

> **说明**
>
> - SQL语句最大不能超过130KB。
> - 使用EMR Spark SQL节点查询数据时，返回的查询结果最大支持10000条数据，并且数据总量不能超过10M。
> - 如果您的工作空间绑定多个EMR引擎，则需要根据业务需求选择合适的引擎。如果仅绑定一个EMR引擎，则无需选择。

调度参数使用详情可参考调度参数概述。

如果您需要修改代码中的参数赋值，请单击界面上方工具栏的**高级运行**。参数赋值逻辑详情请参见运行，高级运行和开发环境冒烟测试赋值逻辑有什么区别。

Spark SQL相关文档请参考Spark SQL作业配置。

6.
7.
8. 保存并提交节点。

> **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标，保存节点。

   ii. 单击工具栏中的图标。

   iii. 在**提交新版本**对话框中，输入**变更描述**。

   iv. 单击**确认**。

   如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

9. 测试节点，详情请参见查看并管理周期任务。

6.4.7. 创建并使用EMR Spark节点

DataWorks的EMR（E-MapReduce） SPARK节点，用于进行复杂的内存分析，构建大型、低延迟的数据分析应用。本文为您介绍如何创建EMR Spark节点，并通过测试计算Pi及Spark对接MaxCompute两个示例，为您介绍EMR Spark节点的功能。

前提条件

创建EMR Spark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. 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. 鼠标悬停至**+新建**图标，单击**EMR > EMR Spark**。

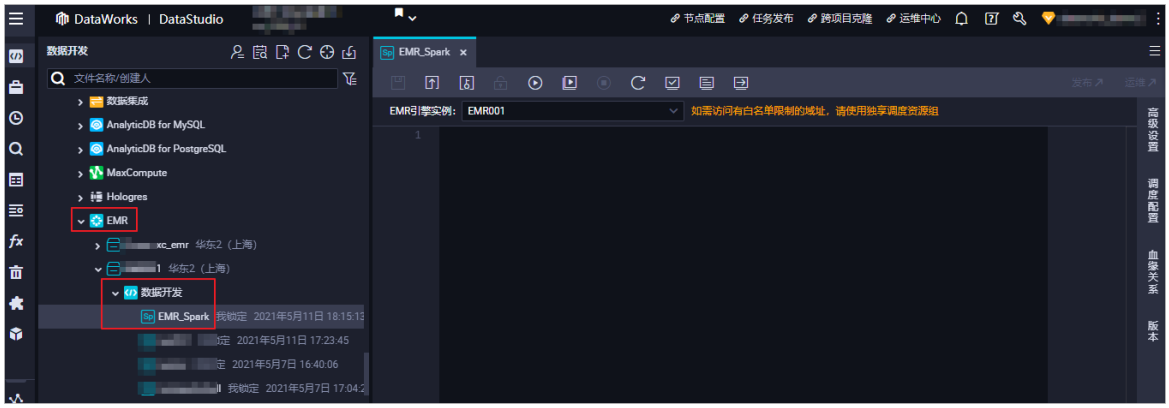
   您也可以找到相应的业务流程，右键单击**EMR**，选择**新建 > EMR Spark**。
3. 在**新建节点**对话框中，输入**节点名称**，并选择**目标文件夹**。

   > **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。

4.
5.
6. 单击**提交**。

   您可以在**EMR**节点下，单击所使用的目标引擎，在**数据开发**中找到新创建的EMR Spark节点。

   > **说明** 如果您的工作空间绑定多个EMR引擎，需要选择EMR引擎。如果仅绑定一个EMR引擎，则无需选择。



保存并提交节点任务

1. 保存并提交节点。

   > **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标，保存节点。

ii. 单击工具栏中的图标。

iii. 在**提交新版本**对话框中，输入**变更描述**。

iv. 单击**确认**。

如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

2. 测试节点，详情请参见查看并管理周期任务。

数据开发示例一：使用计算Pi测试当前EMR Spark环境是否可用

示例一以Spark自带示例项目*计算Pi*为例，测试当前EMR Spark环境是否可用。示例详情请参见示例项目使用说明。

1. 获取Spark自带示例的JAR包*spark-examples_2.11-2.4.5.jar*的存放路径。

   Spark组件安装在*/usr/lib/spark-current*路径下，您需要登录阿里云E-MapReduce控制台，进入目标EMR集群查询完整的路径*/usr/lib/spark-current/examples/jars/spark-examples_2.11-2.4.5.jar*，详情请参见EMR常用文件路径。

   ```
   [root@emr-header-1 ~]# cd /usr/lib/spark-current/examples/jars
   [root@emr-header-1 jars]# ll
   total 2132
   -rw-rw-r-- 1 hadoop hadoop  153982 May 28  2020 scopt_2.11-3.7.0.jar
   -rw-rw-r-- 1 hadoop hadoop 2025084 May 28  2020 spark-examples_2.11-2.4.5.jar
   ```

2. 在创建的EMR Spark节点编辑页面，输入运行代码。创建EMR Spark节点，详情请参见创建EMR Spark节点。

   示例运行代码如下。

   ```
   --class org.apache.spark.examples.SparkPi --master local[8] /usr/lib/spark-current/exam
   ples/jars/spark-examples_2.11-2.4.5.jar 100
   ```

   您仅需填写*spark-submit*后面的内容即可，在作业提交时会自动补全*spark-submit*的内容。实际执行的界面代码如下。

   ```
   # spark-submit [options] --class [MainClass] xxx.jar args
   spark-submit --class org.apache.spark.examples.SparkPi --master local[8] /usr/lib/spark
   -current/examples/jars/spark-examples_2.11-2.4.5.jar 100
   ```

3. 保存并提交运行节点任务，详情请参见*保存并提交节点任务*章节内容。

当返回结果为 `1097: Pi is roughly 3.1415547141554714` 时，表示运行成功，EMR Spark环境可用。

数据开发示例二：Spark对接MaxCompute

本示例以Spark对接MaxCompute，实现通过Spark统计MaxCompute表的行数为例，为您介绍EMR Spark节点的功能应用。更多应用场景请参见EMR Spark开发指南。

执行本示例前，您需要准备如下相关环境及测试数据：

- 准备环境。
  - DataWorks工作空间绑定EMR引擎和MaxCompute引擎，详情请参见配置工作空间
  - 开通OSS并创建Bucket，详情请参见创建存储空间
  - 安装了scala的本地IDE（IDEA）。
- 准备测试数据。

  在DataWorks数据开发页面创建ODPS SQL节点，执行建表语句并插入数据。示例语句如下，设置第一列为BIGINT类型，同时，插入了两条数据记录。创建ODPS SQL节点，详情请参见创建ODPS SQL节点

```
DROP TABLE IF EXISTS emr_spark_read_odpstable ;
CREATE TABLE IF NOT EXISTS emr_spark_read_odpstable
(
    id BIGINT
    ,name STRING
)
;
INSERT INTO TABLE emr_spark_read_odpstable VALUES (111,'zhangsan'),(222,'lisi') ;
```

1. 在Spark中创建Maven工程，添加pom依赖，详情请参见Spark准备工作。

   添加pom依赖，代码如下。

```
    <dependency>
        <groupId>com.aliyun.emr</groupId>
        <artifactId>emr-maxcompute_2.11</artifactId>
        <version>1.9.0</version>
    </dependency>
```

   您可以参考如下插件代码，在实际使用中请以实际代码为准。

```
        <build>
        <sourceDirectory>src/main/scala</sourceDirectory>
        <testSourceDirectory>src/test/scala</testSourceDirectory>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.7.0</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
              <plugin>
                <artifactId>maven-assembly-plugin</artifactId>
                <configuration>
                    <descriptorRefs>
                        <descriptorRef>jar-with-dependencies</descriptorRef>
                    </descriptorRefs>
                </configuration>
                <executions>
                    <execution>
                        <id>make-assembly</id>
```

```
                        <phase>package</phase>
                        <goals>
                            <goal>single</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <groupId>net.alchim31.maven</groupId>
                <artifactId>scala-maven-plugin</artifactId>
                <version>3.2.2</version>
                <configuration>
                    <recompileMode>incremental</recompileMode>
                </configuration>
                <executions>
                    <execution>
                        <goals>
                            <goal>compile</goal>
                            <goal>testCompile</goal>
                        </goals>
                        <configuration>
                            <args>
                                <arg>-dependencyfile</arg>
                                <arg>${project.build.directory}/.scala_dependencies</ar
g>
                            </args>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
```

2. 在Spark中统计MaxCompute表第一列BIGINT类型的行数，详情请参见Spark对接MaxCompute。

   示例代码如下。

```
/*
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements.  See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You under the Apache License, Version 2.0
 * (the "License"); you may not use this file except in compliance with
 * the License.  You may obtain a copy of the License at
 *
 *    http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.aliyun.emr.example.spark
import com.aliyun.odps.TableSchema
```

```
import com.aliyun.odps.data.Record
import org.apache.spark.aliyun.odps.OdpsOps
import org.apache.spark.{SparkConf, SparkContext}
object SparkMaxComputeDemo {
  def main(args: Array[String]): Unit = {
    if (args.length < 6) {
      System.err.println(
        """Usage: SparkMaxComputeDemo <accessKeyId> <accessKeySecret> <envType> <projec
t> <table> <numPartitions>
          |
          |Arguments:
          |
          |    accessKeyId      Aliyun Access Key ID.
          |    accessKeySecret  Aliyun Key Secret.
          |    envType          0 or 1
          |                     0: Public environment.
          |                     1: Aliyun internal environment, i.e. Aliyun ECS etc.
          |    project          Aliyun ODPS project
          |    table            Aliyun ODPS table
          |    numPartitions    the number of RDD partitions
        """.stripMargin)
      System.exit(1)
    }
    val accessKeyId = args(0)
    val accessKeySecret = args(1)
    val envType = args(2).toInt
    val project = args(3)
    val table = args(4)
    val numPartitions = args(5).toInt
    val urls = Seq(
      Seq("http://service.odps.aliyun.com/api", "http://dt.odps.aliyun.com"), // public
environment
      Seq("http://odps-ext.aliyun-inc.com/api", "http://dt-ext.odps.aliyun-inc.com") //
Aliyun internal environment
    )
    val conf = new SparkConf().setAppName("E-MapReduce Demo 3-1: Spark MaxCompute Demo
(Scala)")
    val sc = new SparkContext(conf)
    val odpsOps = envType match {
      case 0 =>
        OdpsOps(sc, accessKeyId, accessKeySecret, urls(0)(0), urls(0)(1))
      case 1 =>
        OdpsOps(sc, accessKeyId, accessKeySecret, urls(1)(0), urls(1)(1))
    }
    val odpsData = odpsOps.readTable(project, table, read, numPartitions)
    println(s"Count (odpsData): ${odpsData.count()}")
  }
  def read(record: Record, schema: TableSchema): Long = {
    record.getBigint(0)
  }
}
```

统计MaxCompute数据完成后，请将该数据生成JAR包。示例生成的JAR包为*emr_spark_demo-1.0-SNAPSHOT-jar-with-dependencies.jar*。

> **说明** 与ODPS相关的依赖均属于第三方包，您需要将第三方包一并生成JAR包上传至目标EMR集群。

3. 上传运行资源。
   i. 登录OSS管控台。
   ii. 上传运行资源（即上一步骤生成的JAR包）至指定OSS路径。

   本示例中，使用的路径为*oss://oss-cn-shanghai-internal.aliyuncs.com/onaliyun-bucket-2/emr_BE/spark_odps/*，您需要上传*emr_spark_demo-1.0-SNAPSHOT-jar-with-dependencies.jar*至该路径。首次使用OSS路径时，需要先进行一键授权，详情请参见创建并使用EMR MR节点。

   > **说明** 由于DataWorks EMR资源的使用上限为50M，而添加依赖的JAR包通常大于50M，所以您需要在OSS控制台上传。如果您的运行资源小于50M，您也可以选择在DataWorks直接上传，详情请参见创建和使用EMR资源

4. 创建EMR Spark节点，并执行节点任务。

   本示例创建的节点命名为*emr_spark_odps*。创建EMR Spark节点，详情请参见创建EMR Spark节点。

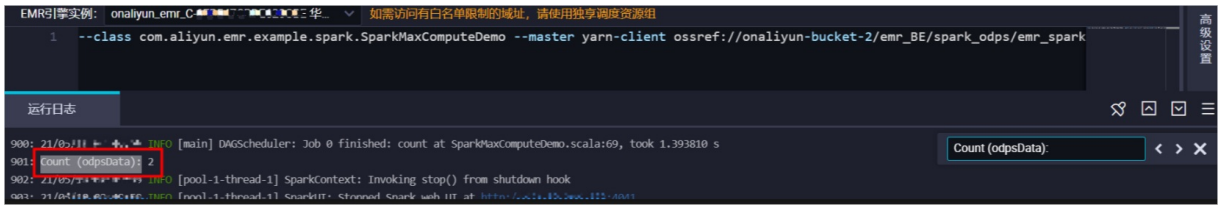
   在*emr_spark_odps*节点的编辑页面，选择所使用的EMR引擎实例，输入如下代码。

```
--class com.aliyun.emr.example.spark.SparkMaxComputeDemo --master yarn-client ossref://
onaliyun-bucket-2/emr_BE/spark_odps/emr_spark_demo-1.0-SNAPSHOT-jar-with-dependencies.j
ar <accessKeyId> <accessKeySecret> 1 onaliyun_workshop_dev emr_spark_read_odpstable 1
```

   其中<accessKeyId>、<accessKeySecret>、<envType>、<project>、<table>、<numPartitions>等参数信息您需要替换为实际使用的相关信息。

5. 保存并提交运行节点任务，详情请参见*保存并提交节点任务*章节内容。

您可以查看运行日志，当返回结果中表记录条数为*2*时，表示统计结果符合预期。



6.4.8. 创建并使用EMR Shell节点

您可以通过创建EMR（E-MapReduce）Shell节点执行Shell脚本任务。

前提条件

创建EMR Shell节点并进行数据开发

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**业务流程**。

如果您已有**业务流程**，则可以忽略该步骤。

i. 鼠标悬停至+新建图标，选择**业务流程**。
ii. 在**新建业务流程**对话框，输入**业务名称**。
iii. 单击**新建**

3. 创建**EMR Shell**节点。

i. 鼠标悬停至+新建图标，选择**EMR > EMR Shell**。

您也可以找到相应的业务流程，右键单击业务流程，选择**新建 > EMR > EMR Shell**。

ii. 在**新建节点**对话框中，输入**节点名称**，并选择**节点类型**及**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**，进入**EMR Shell**节点编辑页面。

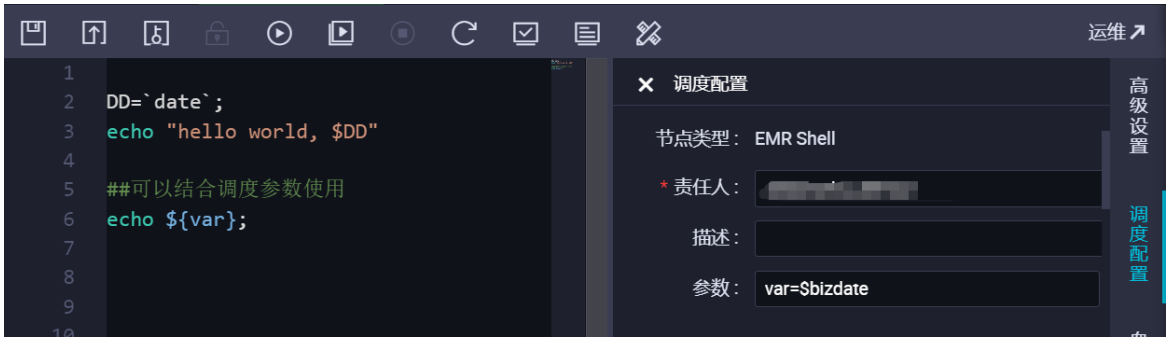
4. 使用**EMR Shell**节点进行数据开发。

示例语句如下。

```
DD=`date`;
echo "hello world, $DD"
##可以结合调度参数使用
echo ${var};
```

调度参数详情请参见调度参数概述。

如果您需要修改代码中的参数赋值，请单击界面上方工具栏的**高级运行**。参数赋值逻辑详情请参见运行，高级运行和开发环境冒烟测试赋值逻辑有什么区别。



更多配置内容，详情请参见Shell作业配置。

5.

6. 任务调度配置。

如果您需要周期性执行创建的节点任务，可以单击节点编辑页面右侧的**调度配置**，根据业务需求配置该节点任务的调度信息：

- 配置任务调度的基本信息，详情请参见配置基础属性。

- 配置时间调度周期、重跑属性和上下游依赖关系，详情请参见时间属性配置说明及配置同周期调度依赖。

  > **说明** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

- 配置资源属性，详情请参见配置资源属性。请选择与所选EMR引擎实例网络连通的独享调度资源组，作为周期调度任务使用的资源组。详情请参见配置资源组与网络连通。

7. 提交并发布节点任务。
   i. 单击工具栏中的图标，保存节点。
   ii. 单击工具栏中的图标，提交节点任务。
   iii. 在**提交新版本**对话框中，输入**变更描述**。
   iv. 单击**确认**。

   如果您使用的是标准模式的工作空间，任务提交成功后，需要将任务发布至生产环境进行发布。请单击右上方的**发布**。具体操作请参见发布任务。
8. 查看周期调度任务。
   i. 单击编辑界面右上角的**运维**，进入生产环境运维中心。
   ii. 查看运行的周期调度任务，详情请参见查看并管理周期任务。

   如果您需要查看更多周期调度任务详情，可单击顶部菜单栏的**运维中心**，详情请参见运维中心概述。

6.4.9. 创建EMR Impala节点

您可以创建EMR（E-MapReduce） Impala节点，对PB级大数据进行快速、实时的交互式SQL查询。

前提条件

操作步骤

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. 鼠标悬停至+新建图标，单击**EMR > EMR Impala**。

   您也可以找到相应的业务流程，右键单击**EMR**，选择**新建 > EMR Impala**。
3. 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。

4. 单击**提交**。
5. 在节点编辑页面，输入需要执行的任务代码。

   SQL代码示例如下。

```
show tables;
CREATE TABLE IF NOT EXISTS userinfo (
ip STRING COMMENT'ip地址',
uid STRING COMMENT'用户ID'
)PARTITIONED BY(
dt STRING
);
ALTER TABLE userinfo ADD IF NOT EXISTS PARTITION(dt=${bizdate}'); //可以结合调度参数使用
。
select * from userinfo ;
```

> **说明**
>
> - SQL语句最大不能超过130KB。
> - 使用EMR Impala节点查询数据时，返回的查询结果最大支持10000条数据，并且数据总量不能超过10M。
> - 如果您的工作空间绑定多个EMR引擎，则需要根据业务需求选择合适的引擎。如果仅绑定一个EMR引擎，则无需选择。

调度参数详情请参见调度参数概述。

如果您需要修改代码中的参数赋值，请单击界面上方工具栏的**高级运行**。参数赋值逻辑详情请参见运行，高级运行和开发环境冒烟测试赋值逻辑有什么区别。

EMR Impala更多作业配置，详情请参考Impala SQL作业配置。

6.
7.
8. 保存并提交节点。

> **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标，保存节点。

   ii. 单击工具栏中的图标。

   iii. 在**提交新版本**对话框中，输入**变更描述**。

   iv. 单击**确认**。

如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

9. 测试节点，详情请参见查看并管理周期任务。

6.4.10. 创建并使用EMR Spark Streaming节点

EMR Spark Streaming节点用于处理高吞吐量的实时流数据，并具备容错机制，可以帮助您快速恢复出错的数据流。本文为您介绍如何创建EMR Spark Streaming节点并进行数据开发。

前提条件

-
- 您在DataWorks工作空间的配置页面添加了E-MapReduce计算引擎实例后，数据开发页面才会显示EMR目录。详情请参见配置工作空间。
- 准备资源组。

  购买独享调度资源组，详情请参见新增和使用独享调度资源组。

使用限制

- EMR Spark Streaming节点仅支持使用独享调度资源组。
- 如果您使用的独享调度资源组和EMR集群是6月10号之前创建的，则需要提交工单升级相关组件。

创建EMR Spark Streaming节点并进行数据开发

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**业务流程**。

   如果您已有**业务流程**，则可以忽略该步骤。
   i. 鼠标悬停至+新建图标，选择**新建业务流程**。
   ii. 在**新建业务流程**对话框，输入**业务名称**。
   iii. 单击**新建**。
3. 创建**EMR Spark Streaming**节点。
   i. 鼠标悬停至+新建图标，单击**EMR > EMR Spark Streaming**。

   您也可以找到相应的业务流程，右键单击目标业务流程，选择**新建 > EMR > EMR Spark Streaming**。
   ii. 在**新建节点**对话框中，输入**节点名称**，并选择**节点类型**及**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**，进入**EMR Spark Streaming**节点编辑页面。
4. 使用**EMR Spark Streaming**节点进行数据开发。
   i. 选择目标EMR引擎。

   在节点编辑页面的**EMR引擎实例**列表，选择需要使用的目标EMR引擎。

ii. 编写作业代码。

在EMR Spark Streaming节点的编辑页面，输入需要执行的作业代码。

示例：在EMR Spark Streaming节点中输入以下代码。

```
--master yarn-cluster --executor-cores 2 --executor-memory 2g --driver-memory 1g --num-executors 2 --class com.aliyun.emr.example.spark.streaming.JavaLoghubWordCount /tmp/examples-1.2.0-shaded.jar <logService-project> <logService-store> <group> <endpoint> <access-key-id> <access-key-secret>
```

EMR Spark Streaming节点将自动生成 `spark-submit` ，您无需手动输入。最终下发至引擎的代码为：

```
spark-submit --master yarn-cluster --executor-cores 2 --executor-memory 2g --driver-memory 1g --num-executors 2 --class com.aliyun.emr.example.spark.streaming.JavaLoghubWordCount /tmp/examples-1.2.0-shaded.jar <logService-project> <logService-store> <group> <endpoint> <access-key-id> <access-key-secret>
```

- `/tmp/examples-1.2.0-shaded.jar` 为您实际需要执行的任务代码所生成的JAR包。具体的任务代码示例，详情请参见实时Spark Streaming消费示例。

  > **说明** 目前您的任务JAR包仅支持如下两种存放路径：
  > - JAR包存放在EMR集群的Master机器中。
  > - JAR包存放在对象存储服务（Object Storage Service，OSS）中。推荐您使用OSS进行存放。使用OSS存放JAR包，详情请参见控制台使用流程。

- `access-key-id` 及 `access-key-secret` 需要替换为您所使用的阿里云账号的AccessKey ID及AccessKey Secret。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，进入AccessKey管理，获取AccessKey ID及AccessKey Secret。

更多Spark Streaming的参数，详情请参见Spark文档。

iii. 配置调度资源组。

- 单击工具栏中的图标，在**参数**对话框中选择需要使用的调度资源组。
- 单击**确定**。

iv. 保存并运行任务。

在工具栏中，单击图标，保存节点任务，单击图标，运行节点任务。

5. 任务调度配置。

如果您需要周期性执行创建的节点任务，可以单击节点编辑页面右侧的**配置**，根据业务需求配置该节点任务的调度信息：

- 配置任务调度的基本信息，详情请参见配置基础属性。
- 配置任务调度的时间周期。

  您可以选择任务的启动方式及出错重跑属性，详情请参见配置时间属性。
- 配置资源属性，详情请参见配置资源属性。

6. 提交并发布节点任务。

i. 单击工具栏中的图标，保存节点。

ii. 单击工具栏中的图标，提交节点任务。

iii. 在**提交新版本**对话框中，输入**变更描述**。

iv. 单击**确定**。

如果您使用的是标准模式的工作空间，任务提交成功后，需要将任务发布至生产环境进行发布。请单击顶部菜单栏左侧的**任务发布**。具体操作请参见发布任务。

7. 查看实时计算任务。

i. 单击编辑界面右上角的**运维**，进入运维中心。

ii. 查看运行的实时计算任务，详情请参见实时计算任务运行与管理。

6.5. Hologres SQL节点

Hologres与MaxCompute在底层无缝连接，您无须移动数据，即可使用标准的PostgreSQL语句查询分析MaxCompute中的海量数据，快速获取查询结果。

前提条件

您在**工作空间配置**页面添加**Hologres**计算引擎实例后，当前页面才会显示**Hologres**目录。详情请参见绑定Hologres计算引擎。

操作步骤

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**业务流程**。

如果您已有**业务流程**，则可以忽略该步骤。

i. 鼠标悬停至+新建图标，选择**业务流程**。

ii. 在**新建业务流程**对话框，输入**业务名称**。

iii. 单击**新建**。

3. 创建**Hologres SQL**节点。

i. 鼠标悬停至+新建图标，单击**Hologres > Hologres SQL**。

您也可以找到相应的业务流程，右键单击**Hologres**，选择**新建 > Hologres SQL**。

ii. 在**新建节点**对话框中，输入**节点名称**，并选择已创建的**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. 在节点的编辑页面，编辑并运行代码。

新建节点成功后，编写符合语法的Hologres SQL代码，SQL语法请参见SELECT。

> **说明** 使用不包含 `limit` 限制条件的 `SELECT` 语法查询数据时，默认只显示200条查询结果。如果您需要显示更多数据，则可以在 `SELECT` 语法后添加 `limit` 限制。最多支持显示10000条查询结果。
>
> 例如，使用 `select col_1,col_2 from your_table_name where pt>0 limit 500;` 语句查询数据时，可显示500条查询结果。

5. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性。详情请参见配置基础属性。
6. 保存并提交节点。

> **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标，保存节点。
   ii. 单击工具栏中的图标。
   iii. 在**提交新版本**对话框中，输入**变更描述**。
   iv. 单击**确认**。

   如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。
7. 测试节点，详情请参见查看并管理周期任务。

6.6. AnalyticDB for PostgreSQL节点

您可以在DataWorks中创建AnalyticDB for PostgreSQL节点，构建在线ETL数据处理流程。

前提条件

- 您需要购买DataWorks标准版及以上版本，才可以绑定AnalyticDB for PostgreSQL计算引擎实例。
- 您在**工作空间配置**页面添加AnalyticDB for PostgreSQL引擎后，当前页面才会显示AnalyticDB for PostgreSQL目录。详情请参见配置工作空间。
- 新增独享调度资源组，详情请参见购买资源组（创建订单）。

> **说明** AnalyticDB for PostgreSQL节点仅支持使用独享调度资源组。

背景信息

AnalyticDB for PostgreSQL节点用于接入阿里云产品分析型数据库PostgreSQL版，详情请参见分析型数据库PostgreSQL版。

操作步骤

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，鼠标悬停至+新建图标，单击**AnalyticDB > ADB for PostgreSQL**。

您也可以打开相应的业务流程，右键单击**AnalyticDB for PostgreSQL**，选择**新建 > ADB for PostgreSQL**。

3. 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。

4. 单击**提交**。
5. 编辑AnalyticDB for PostgreSQL节点：
   i. 从**选择数据源**下拉列表中选择数据源。

   > **注意**
   > - 绑定AnalyticDB for PostgreSQL计算引擎时，会自动创建一个数据源。
   > - 仅支持选择连接串方式配置的数据源。

   ii. 编辑SQL语句。

   选择相应的数据源后，根据PostgreSQL支持的语法，编写SQL语句。详情请参见SQL语法。

   iii. 单击工具栏中的图标，将其保存至服务器。

   iv. 单击工具栏中的图标，执行编辑的SQL语句。

   第一次运行该节点时，您需要在**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，单击**确定**。

   下一次运行会自动使用第一次选择的资源组和变量的赋值，如果您需要修改资源组或变量的赋值，请单击工具栏中的图标，使用**高级运行**功能。

   > **说明** 因为访问专有网络环境的数据源需要使用独享调度资源组执行任务，所以此处必须选择测试连通性成功的独享调度资源组。

6. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性，详情请参见配置基础属性。

   配置资源属性时，请选择调度资源组为已经和AnalyticDB for PostgreSQL网络连通的独享调度资源组，作为周期调度时使用的资源组。

7. 保存并提交节点。

   > **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标，保存节点。

   ii. 单击工具栏中的图标。

   iii. 在**提交新版本**对话框中，输入**变更描述**。

   iv. 单击**确认**。

   如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

8. 测试节点，详情请参见查看并管理周期任务。

6.7. AnalyticDB for MySQL节点

您可以通过创建AnalyticDB for MySQL节点，直接使用SQL语句对目标AnalyticDB for MySQL数据源进行数据开发。本文为您介绍如何创建并使用AnalyticDB for MySQL节点。

前提条件

- 准备相应版本软件并配置环境。
  - 购买DataWorks标准版及以上版本，详情请参见版本服务计费说明。
  - 购买AnalyticDB for MySQL，详情请参见创建集群。
  - 在DataWorks**工作空间配置**页面添加AnalyticDB for MySQL引擎，详情请参见配置工作空间。
- 准备资源组。

  购买独享调度资源组，详情请参见购买资源组（创建订单）。
- 准备数据源。

  创建AnalyticDB for MySQL数据源，详情请参见配置AnalyticDB for MySQL 3.0数据源。

背景信息

AnalyticDB for MySQL是阿里云的一种分析型数据库，详情请参见云原生数据仓库MySQL版。

使用限制

- DataWorks标准版及以上版本，才可以绑定AnalyticDB for MySQL计算引擎实例。
- AnalyticDB for MySQL节点仅支持使用独享调度资源组。
- AnalyticDB for MySQL节点仅支持对通过连接串模式创建的AnalyticDB for MySQL数据源和引擎绑定AnalyticDB for MySQL数据源进行数据开发。您可以进入**数据源管理**页面，单击目标数据源**操作**列的**编辑**，在数据源编辑页面查看创建数据源时所使用的模式，详情请参见配置AnalyticDB for MySQL 3.0数据源。

创建AnalyticDB for MySQL节点并进行数据开发

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**业务流程**。

   如果您已有**业务流程**，则可以忽略该步骤。
   i. 鼠标悬停至**+新建**图标，选择**新建业务流程**。
   ii. 在**新建业务流程**对话框，输入**业务名称**。
   iii. 单击**新建**。
3. 创建**AnalyticDB for MySQL**节点。
   i. 鼠标悬停至**+新建**图标，选择**AnalyticDB for MySQL > ADB for MySQL**。

      您也可以找到相应的业务流程，右键单击业务流程，选择**新建 > AnalyticDB for MySQL > ADB for MySQL**。

ii. 在**新建节点**对话框中，输入**节点名称**，并选择**节点类型**及**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**，进入AnalyticDB for MySQL节点编辑页面。

4. 使用**AnalyticDB for MySQL**节点进行数据开发。

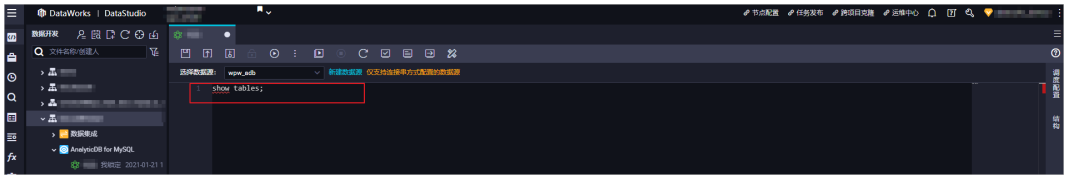
i. 选择数据源。

在**选择数据源**下拉框，选择进行数据开发需要使用的目标数据源。如果下拉列表中没有需要的数据源，请单击右侧的**新建数据源**，在**数据源管理**页面新建，详情请参见配置AnalyticDB for MySQL 3.0数据源。

> **说明**
>
> - 绑定AnalyticDB for MySQL计算引擎时，会自动创建一个数据源，默认使用该数据源进行数据开发，您也可以选择需要使用的数据源。
> - AnalyticDB for MySQL节点仅支持对通过连接串模式创建的AnalyticDB for MySQL数据源和引擎绑定AnalyticDB for MySQL数据源进行数据开发。您可以进入**数据源管理**页面，单击目标数据源**操作**列的**编辑**，在数据源编辑页面查看创建数据源时所使用的模式，详情请参见配置AnalyticDB for MySQL 3.0数据源。

ii. 使用SQL语句创建任务。

a. 在SQL编辑区域编写SQL任务。



示例查询目标数据源中的表，语句如下。实际使用时，您可以根据AnalyticDB for MySQL支持的语法，编写需要执行的语句。

```
show tables;
```

b. 选择资源组。

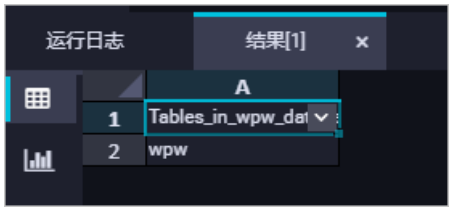
在工具栏单击图标，在**参数**对话框选择已创建的独享调度资源组。

> **说明**
> - 您需要使用与数据源测试连通性成功的独享调度资源组。详情请参见配置资源组与网络连通。
> - 如果您后续执行任务需要修改使用的资源组，也可以在此处选择需要更换的调度资源组。

c. 保存并运行SQL语句。

在工具栏，单击图标，保存编写的SQL语句，单击图标，运行创建的SQL任务。

本文示例SQL语句的运行结果如下。



5. 任务调度配置。

如果您需要周期性执行创建的节点任务，可以单击节点编辑页面右侧的**调度配置**，根据业务需求配置该节点任务的调度信息：

- 配置任务调度的基本信息，详情请参见配置基础属性。
- 配置时间调度周期、重跑属性和上下游依赖关系，详情请参见时间属性配置说明及配置同周期调度依赖。

> **说明** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

- 配置资源属性，详情请参见配置资源属性。请选择与AnalyticDB for MySQL数据源网络连通的独享调度资源组，作为周期调度任务使用的资源组。详情请参见配置资源组与网络连通。

6. 提交并发布节点任务。

i. 单击工具栏中的图标，保存节点。

ii. 单击工具栏中的图标，提交节点任务。

iii. 在**提交新版本**对话框中，输入**变更描述**。

iv. 单击**确认**。

如果您使用的是标准模式的工作空间，任务提交成功后，需要将任务发布至生产环境进行发布。请单击右上方的**发布**。具体操作请参见发布任务。

7. 查看周期调度任务。

i. 单击编辑界面右上角的**运维**，进入运维中心。

ii. 查看运行的周期调度任务，详情请参见查看并管理周期任务。

6.8. MySQL节点

您可以通过创建MySQL节点，直接使用SQL语句对目标MySQL数据源进行数据开发。本文为您介绍如何创建并使用MySQL节点。

前提条件

- MySQL节点仅支持使用独享调度资源组，独享调度资源组的使用请参考文档：新增和使用独享调度资源组。
- 已创建连接串形式添加的MySQL数据源，详情请参见配置MySQL数据源。同时，创建的数据源需要与进行任务调度的资源组测试连通性成功。

> **说明** MySQL节点仅支持使用生产环境的数据源。

使用限制

- MySQL节点仅支持对连接串模式创建的生产环境MySQL数据源进行数据开发。您可以参考配置MySQL数据源进入**数据源管理**页面，单击目标数据源**操作**列的**编辑**，在数据源编辑页面查看创建数据源时所使用的模式。
- 数据源访问公网时，需要配置白名单。为保证开发任务不受资源组连通性阻碍，建议使用独享调度资源组。
- 数据源访问VPC网络时，仅支持使用独享调度资源组进行数据开发。

> **说明** 当前节点类型不支持mysql8.0及以上版本。

网络联通方案与建议

数据源与资源组的网络联通方案与建议如下：

- 独享调度资源组的资源可以随时调配，且可以保障任务产出，建议执行任务使用独享调度资源组。
- 需要访问VPC网络的数据源，请使用独享调度资源组。
- 需要访问公网的数据源，推荐使用独享调度资源组。
- 需要访问公网的数据源，如果使用公共调度资源组，则需要配置白名单。

创建并使用MySQL节点进行数据开发

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**数据开发**。

2. 创建**业务流程**。

如果您已有**业务流程**，则可以忽略该步骤。

i. 鼠标悬停至+新建图标，选择**新建业务流程**。

ii. 在**新建业务流程**对话框，输入**业务名称**。

iii. 单击**新建**。

3. 创建**MySQL**节点。

i. 鼠标悬停至+新建图标，选择**新建节点 > 数据库 > MySQL**。

您也可以找到相应的业务流程，右键单击业务流程，选择**新建节点 > 数据库 > MySQL**。

ii. 在**新建节点**对话框中，输入**名称**，并选择**节点类型**及**路径**。

> **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**，进入MySQL节点编辑页面。

4. 使用**MySQL**节点进行数据开发。

i. 选择数据源。

在**选择数据源**下拉框，选择进行数据开发需要使用的目标数据源。如果下拉列表中没有需要的数据源，请单击右侧的**新建数据源**，在**数据源管理**页面新建，详情请参见配置MySQL数据源。

> **说明**
>
> MySQL节点仅支持对连接串模式创建的生产环境MySQL数据源进行数据开发。您可以参考配置MySQL数据源进入**数据源管理**页面，单击目标数据源**操作**列的**编辑**，在数据源编辑页面查看创建数据源时所使用的模式。

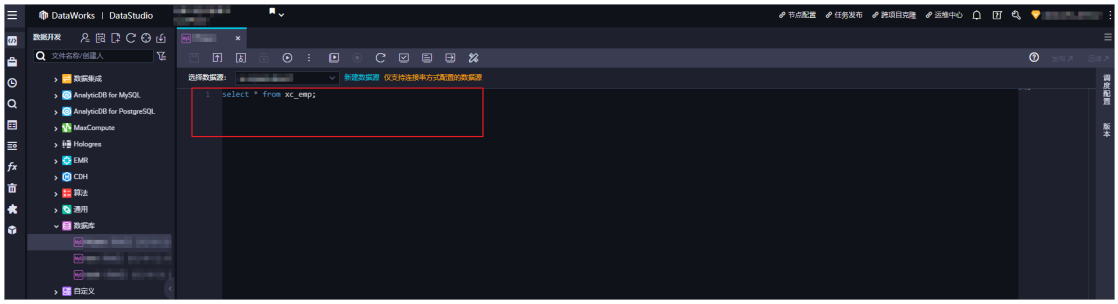
ii. 选择资源组。

在工具栏单击图标，在**参数**对话框选择已创建的调度资源组。

> **说明**
>
> - 访问公共网络或VPC网络环境的数据源需要使用与数据源测试连通性成功的调度资源组。详情请参见配置资源组与网络连通。
> - 如果您后续执行任务需要修改使用的资源组，也可以在此处选择需要更换的调度资源组。

iii. 使用SQL语句创建任务。

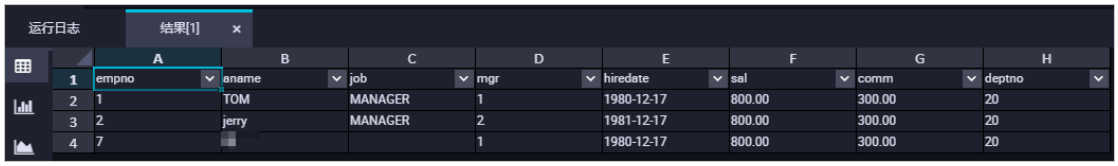
在SQL编辑区域使用SQL语句创建任务。



示例查询*xc_emp*表的内容，语句如下。实际使用时，您可以根据MySQL支持的语法，编写需要执行的语句。

```
select * from xc_emp;
```

运行结果如下。



	A	B	C	D	E	F	G	H
1	empno	aname	job	mgr	hiredate	sal	comm	deptno
2	1	TOM	MANAGER	1	1980-12-17	800.00	300.00	20
3	2	jerry	MANAGER	2	1981-12-17	800.00	300.00	20
4	7			1	1980-12-17	800.00	300.00	20

如果任务执行失败，您可以查看任务运行失败的错误提示，参考任务运行失败常见问题：界面提示sql execute failed! 暂不支持的jdbc驱动进行排查处理。

iv. 保存并运行SQL语句。

在工具栏，单击图标，保存编写的SQL语句，单击图标，运行创建的SQL任务。

5. 任务调度配置。

如果您需要周期性执行创建的节点任务，可以单击节点编辑页面右侧的**调度配置**，根据业务需求配置该节点任务的调度信息：

- 配置任务调度的基本信息，详情请参见配置基础属性。
- 配置时间调度周期、重跑属性和上下游依赖关系，详情请参见时间属性配置说明及配置同周期调度依赖。

> **说明** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

- 配置资源属性，详情请参见配置资源属性。访问公网或VPC网络的MySQL数据源，请选择与MySQL数据源网络连通的调度资源组，作为周期调度任务使用的资源组。详情请参见配置资源组与网络连通。

6. 提交并发布节点任务。

i. 单击工具栏中的图标，保存节点。

ii. 单击工具栏中的图标，提交节点任务。

iii. 在**提交新版本**对话框中，输入**变更描述**。

iv. 单击**确定**。

如果您使用的是标准模式的工作空间，任务提交成功后，需要将任务发布至生产环境进行发布。请单击顶部菜单栏左侧的**任务发布**。具体操作请参见发布任务。

7. 查看周期调度任务。
   i. 单击编辑界面右上角的**运维**，进入生产环境运维中心。
   ii. 查看运行的周期调度任务，详情请参见查看并管理周期任务。

   如果您需要查看更多周期调度任务详情，可单击顶部菜单栏的**运维中心**，详情请参见运维中心概述。

任务运行失败常见问题：界面提示sql execute failed! 暂不支持的jdbc驱动

- 问题描述

  添加MySQL数据源时，选择了非连接串模式创建的数据源，导致运行任务时失败，报错信息为`sql execute failed! 暂不支持的jdbc驱动`。

- 问题原因

  出现上述报错通常都是选择了非连接串模式创建的MySQL数据源导致。

- 解决方案

  重新选择使用连接串模式创建的数据源。您可以参考配置MySQL数据源进入**数据源管理**页面，单击目标数据源**操作**列的**编辑**，在数据源编辑页面查看创建数据源时所使用的模式。

6.9. 机器学习（PAI）节点

机器学习节点用于调用您在机器学习平台中构建的任务，并按照节点配置进行调度生产。

前提条件

您需要在机器学习平台创建机器学习实验后，才可以在DataWorks中进行添加。

本文以心脏病预测案例实验为例，为您介绍如何加载机器学习实验至DataWorks的机器学习节点中。创建机器学习实验的详情请参见心脏病预测案例。

> **说明** 在进行此文档操作前，请先在DataStudio界面左上角所有产品中进入机器学习PAI模块，根据心脏病预测案例创建机器学习实验，加载到数据开发DataStudio后，再根据当前文档进行相应的操作。

操作步骤

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，鼠标悬停至**+新建**图标，单击**算法 > 机器学习（PAI）**。

   您也可以打开相应的业务流程，右键单击**算法**，选择**新建 > 机器学习（PAI）**。
3. 在**新建节点**对话框中，输入**节点名称**，并选择**目标文件夹**。

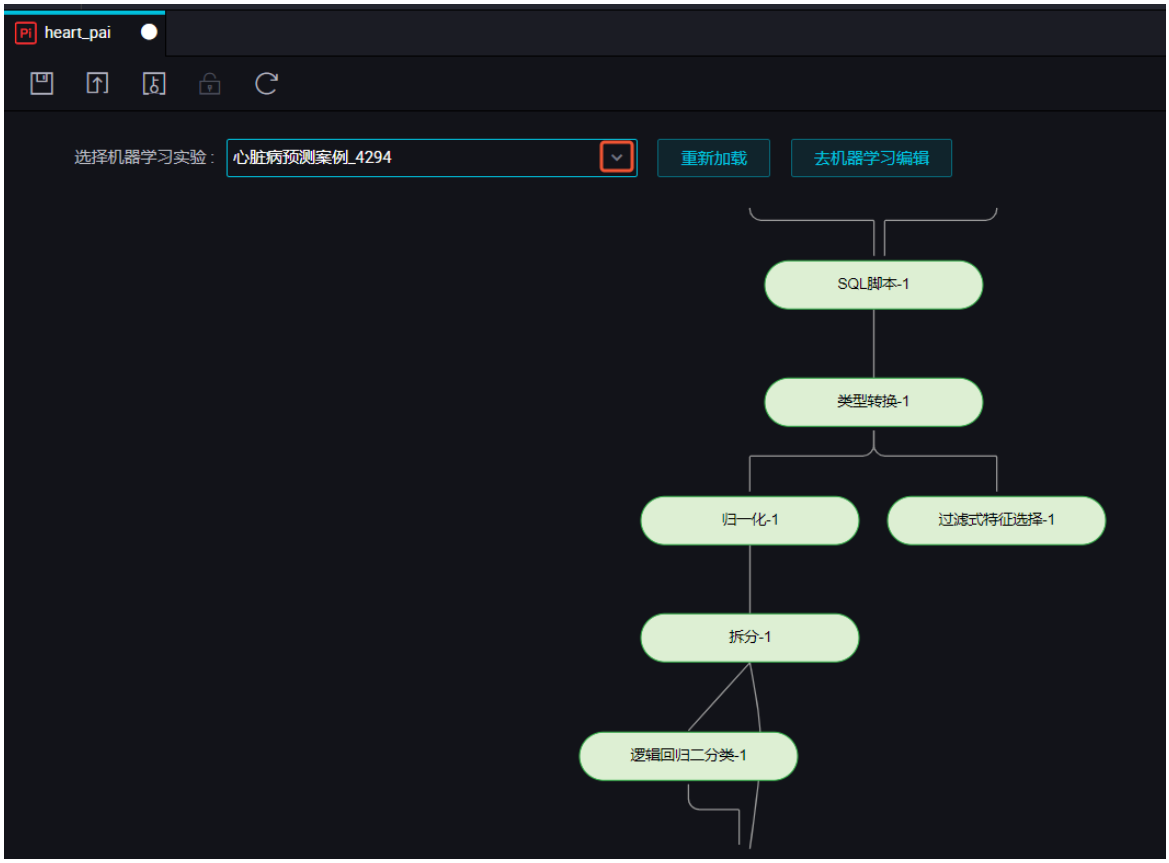
   > **说明** 节点名称必须是大小写字母、中文、数字、下划线（_）和小数点（.），且不能超过128个字符。

4. 单击**提交**。

5. 在节点的编辑页面，从**选择机器学习实验**下拉列表中选择已创建的机器学习实验。

   如果您需要修改机器学习实验，请单击**去机器学习编辑**，进入实验编辑页面进行编辑。

   > **说明** 您需要先进入机器学习界面创建机器学习实验，此处才能下拉编辑并跳转对应的机器学习实验，如果您界面下拉选择机器学习实验时显示是空，请在左上角所有产品中找到机器学习PAI模块，创建机器学习实验。



6. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性，详情请参见配置基础属性。
7. 保存并提交节点。

   > **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标，保存节点。

   ii. 单击工具栏中的图标。

   iii. 在**提交新版本**对话框中，输入**变更描述**。

   iv. 单击**确认**。

   如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。
8. 测试节点，详情请参见查看并管理周期任务。

6.10. ClickHouse SQL节点

您可以创建ClickHouse SQL节点，实现分布式SQL查询引擎处理结构化数据，提高作业的执行效率。本文为您介绍如何创建ClickHouse SQL节点并进行数据开发。

前提条件

- 您已创建EMR ClickHouse或数据库ClickHouse集群，且集群所在的安全组中入方向的安全策略包含以下策略。
- 所使用的DataWorks工作空间添加了ClickHouse计算引擎，详情请参见配置工作空间。

  > **说明** DataWorks工作空间添加了ClickHouse计算引擎后，**数据开发**页面才会显示ClickHouse目录。

- 已开通独享调度资源组，并且独享调度资源组需要绑定ClickHouse集群所在的VPC专有网络，详情请参见新增和使用独享调度资源组。

使用限制

仅支持使用独享调度资源组运行ClickHouse SQL节点任务。

操作步骤

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. 鼠标悬停至**+新建**图标，单击**ClickHouse > Click SQL**。

   您也可以找到相应的业务流程，右键单击**ClickHouse**，选择**新建 > Click SQL**。
3. 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。

4. 单击**提交**。
5. 在节点编辑页面进行数据开发。

   您可以根据业务需求，在节点编辑页面执行SQL任务。示例运行的任务代码如下。

   ```
   CREATE DATABASE if not EXISTS ck_test;
   CREATE TABLE if not EXISTS  ck_test.first_table  (
   `product_code` String,
   `package_name` String
   ) ENGINE = MergeTree ORDER BY package_name SETTINGS index_granularity = 8192;
   insert into ck_test.first_table (product_code, package_name) VALUES ('1', '1');
   select * from ck_test.first_table;
   ```

6. 保存并提交节点。

   > **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

i. 单击工具栏中的图标，保存节点。

ii. 单击工具栏中的图标。

iii. 在**提交新版本**对话框中，输入**变更描述**。

iv. 单击**确认**。

如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

7. 测试节点，详情请参见查看并管理周期任务。

6.11. 通用节点

6.11.1. OSS对象检查节点

您可以在下游任务需要依赖该OSS对象传入OSS时，使用OSS对象检查功能。例如，同步OSS数据至DataWorks，需要检测出已经产生OSS数据文件，才可以进行OSS同步任务。

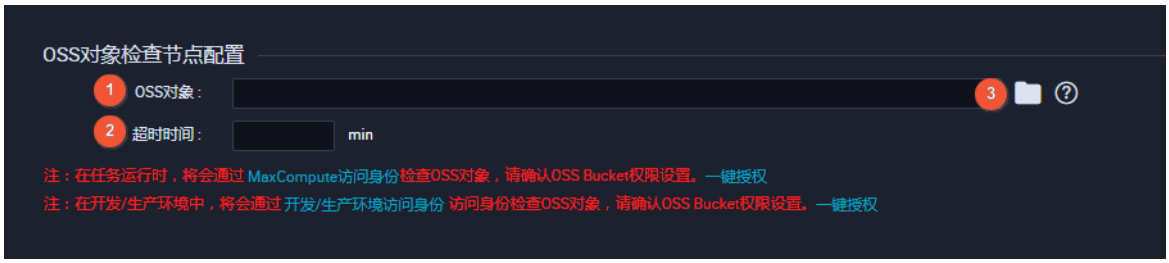
OSS对象检查可以检测所有租户下的OSS对象，具体操作步骤如下：

1. 登录DataWorks控制台，单击相应工作空间后的**进入数据开发**。
2. 鼠标悬停至+新建，单击**通用 > OSS对象检查**。

   您也可以打开相应的业务流程，右键单击**通用**，选择**新建 > OSS对象检查**。
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。

4. 新建成功后，进行**OSS对象检查节点配置**。



序号	参数	描述
1	OSS对象	此处可以手动填写OSS对象的存储路径，路径支持使用调度参数，详情请参见调度参数概述。 

序号	参数	描述
2	**超时时间**	在超时时间内，每5秒检测该OSS对象是否存在于OSS中。如果超出超时时间，仍未检测到OSS对象的存在，则OSS对象检查任务会失败。
3	**选择存储地址**	您可以选择以下两种存储地址： ○ **自己的存储**：检测当前租户下的OSS对象。 ○ **别人的存储**：检测非当前租户下的OSS对象。

> 说明
> - 任务在运行时，会通过MaxCompute访问身份检查OSS对象，请确认OSS Bucket的权限设置，详情请参见STS模式授权。
> - 在开发/生产环境中，任务会通过开发/生产环境访问身份检查OSS对象，请确认OSS Bucket的权限设置。
> - OSS对象检查不支持通配符，也不支持系统参数cyctime和bizdate，您可以使用自定义参数。

5. 在RAM中授权MaxCompute访问OSS的权限。

   MaxCompute结合了阿里云的访问控制服务（RAM）和令牌服务（STS），来解决账号的安全问题。

   - 当MaxCompute和OSS的owner是同一个账号时，可以直接在RAM控制台进行一键授权操作。
   - 当MaxCompute和OSS的owner不是同一账号时，可以通过以下操作进行授权：
     a. 在RAM中授权MaxCompute访问OSS的权限。

        创建如AliyunODPSDefaultRole或AliyunODPSRoleForOtherUser的角色，并设置如下策略内容。

```
--MaxCompute和OSS的Owner不是同一个账号。
{
"Statement": [
{
"Action": "sts:AssumeRole",
"Effect": "Allow",
"Principal": {
"Service": [
"MaxCompute的Owner云账号id@odps.aliyuncs.com"
]
}
}
],
"Version": "1"
}
```

b. 授予角色访问OSS必要的权限AliyunODPSRolePolicy。

```
{
"Version": "1",
"Statement": [
{
"Action": [
 "oss:ListBuckets",
 "oss:GetObject",
 "oss:ListObjects",
 "oss:PutObject",
 "oss:DeleteObject",
 "oss:AbortMultipartUpload",
 "oss:ListParts"
],
"Resource": "*",
"Effect": "Allow"
}
]
}
--您可以自定义其他权限。
```

c. 将权限AliyunODPSRolePolicy授权给该角色。

6. 进入**运维中心**页面，查看运行日志。

   如果出现如下所示的日志信息，说明未检测到OSS对象产生。

```
<Error>
 <Code>NoSuchKey</Code>
 <Message>The specified key does not exist.</Message>
 <RequestId></RequestId>
 <HostId>oss对象</HostId>
 <Key>xc/111.txt</Key>
</Error>
```

6.11.2. for-each节点

6.11.2.1. 逻辑原理介绍

DataWorks为您提供遍历节点（for-each节点），您可以通过for-each节点来循环遍历赋值节点传递的结果集。同时您也可以重新编排for-each节点内部的业务流程。本文为您介绍for-each节点的组成与应用逻辑。

应用场景

DataWorks的for-each节点主要用于有循环遍历的场景，且需要与赋值节点联合使用，将赋值节点作为for-each节点的上游节点，将赋值节点的输出结果赋值给for-each节点后，一次次循环来遍历赋值节点的输出结果。

使用for-each节点时，使用限制与注意事项请参见下文的使用限制与注意事项。

此外，for-each节点也是拥有内部节点的节点，内部节点可用来编译循环遍历的任务代码，详情请参见下文的节点组成。

使用限制与注意事项

- 上下游依赖

  for-each遍历节点需要遍历赋值节点传递的值，所以赋值节点需作为for-each节点的上游节点，for-each节点需要依赖赋值节点。

- 循环支持
  - 仅DataWorks标准版及以上版本支持使用for-each节点。
  - for-each节点最多支持循环128次，如果超过了128次，则运行会报错。实际循环遍历次数由上游赋值节点实际输出控制。
    - 一维数组类型的输出，循环遍历次数即为一维数组元素的个数。

      例如，赋值节点的赋值语言为Shell或Python（Python2）时，输出结果为一维数组：`2021-03-28,2021-03-29,2021-03-30,2021-03-31,2021-04-01`，则for-each节点会循环5次完成遍历。
    - 二维数组类型的输出，循环遍历次数即为二维数组元素的行数。

      例如，赋值节点的赋值语言为OdpsSQL时，输出结果为二维数组：

      ```
      +-----------------------------------------------+
      | uid             | region | age_range | zodiac |
      +-----------------------------------------------+
      | 0016359810821   | 湖北省 | 30~40岁   | 巨蟹座 |
      | 0016359814159   | 未知   | 30~40岁   | 巨蟹座 |
      +-----------------------------------------------+
      ```

      则for-each节点会循环2次完成遍历。

- 内部节点
  - 您可以删除for-each节点的内部节点间的依赖关系，重新编排内部业务流程，但需要分别将**start节点**、**end节点**分别作为for-each节点内部业务流程的首末节点。
  - 在for-each节点的内部节点使用分支节点进行逻辑判断或者结果遍历时，需要同时使用归并节点。

- 调测运行

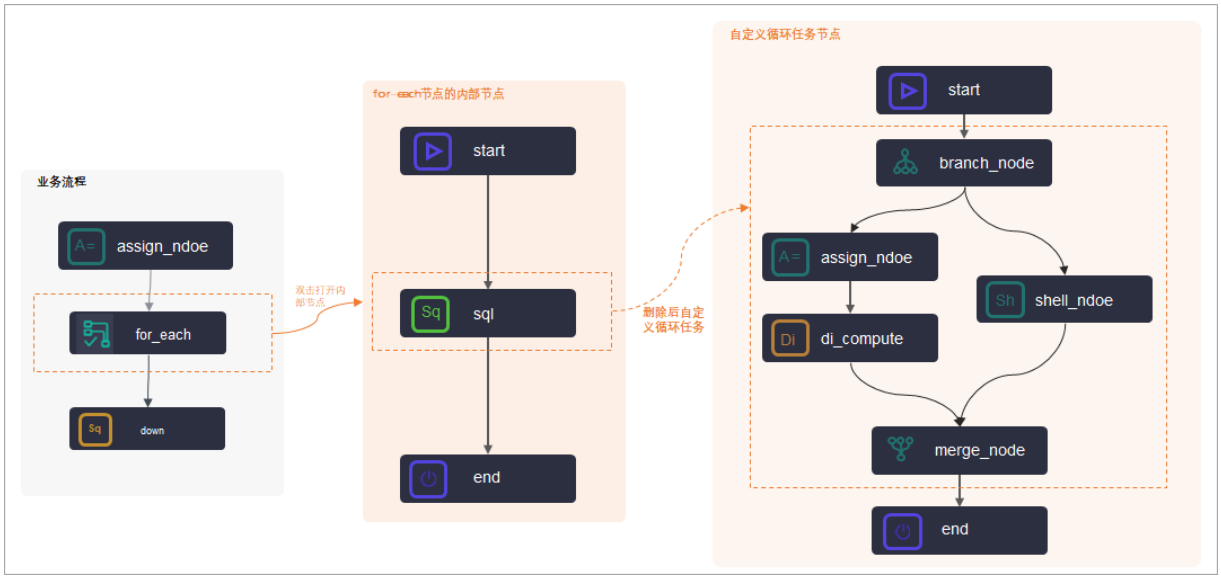
- DataWorks为标准模式时，不支持在DataStudio界面直接测试运行for-each节点。

  如果您想测试验证for-each节点的运行结果，您需要将包含for-each节点的任务发布提交到运维中心，在运维中心页面运行for-each节点任务。

- 在运维中心查看for-each节点的执行日志时，您需要右键实例，单击**查看内部节点**来查看内部节点的执行日志。

节点组成

DataWorks的for-each节点是包含内部节点的一种特殊节点，您在创建完成for-each节点时，同时也自动创建完成了三个内部节点：**start节点**（循环开始节点）、**sql节点**（循环任务节点）、**end节点**（循环结束判断节点），通过内部节点组织成内部节点流程，实现对上游赋值接节点输出结果的循环遍历。



如上图所示：

- **sql节点**

  DataWorks默认为您创建好了一个SQL类型的内部任务运行节点，您也可以删除默认的**sql节点**后，自定义内部循环遍历任务的运行节点。

  - 您的循环遍历任务是SQL类型的任务，则可以直接双击默认的**sql节点**，进入节点的代码开发页面开发任务代码。
  - 您的循环遍历任务比较复杂，您可以在内部节点流程中新建其他任务节点，并根据实际情况重新构建节点的运行流程。

  > **说明** 自定义循环任务节点时，您可以删除内部节点间的依赖关系，重新编排循环节点内部业务流程，但需要分别将**start节点**、**end节点**分别作为for-each节点内部业务流程的首末节点。

- **start节点与end节点**

  是内部节点业务流程每次循环遍历的开始节点与结束节点，不承载具体的任务代码。

  > **说明** for-each节点的end节点不控制循环遍历的次数，for-each节点的循环遍历次数由上游赋值节点实际输出控制。

内置变量

DataWorks的for-each节点每次循环遍历赋值节点的输出结果时，您可以通过一些内置的变量来获取当前已循环次数和偏移量。

内置变量	含义	与for循环对比
`${dag.loopDataArray}`	获取赋值节点的数据集	相当于for循环中的代码结果： `data=[]`
`${dag.foreach.current}`	获取当前遍历值	以下面的for循环代码为例： `for(int i=0;i<data.length;i++) { print(data[i]); }` • `data[i]` 相当于 `${dag.foreach.current}` 。 • `i` 相当于 `${dag.offset}` 。
`${dag.offset}`	当前偏移量（每一次遍历相对于第一次的偏移量）	
`${dag.loopTimes}`	获取当前遍历次数	-

在您了解自己输出的表结构的情况下，您可以使用如下变量方式，获取其他变量取值。

其他变量	含义
`${dag.foreach.current[n]}`	上游赋值节点的输出结果为二维数组时，每次遍历时获取当前数据行的某列的数据。
`${dag.loopDataArray[i][j]}`	上游赋值节点的输出结果为二维数组时，获取数据集中具体i行j列的数据。
`${dag.foreach.current[n]}`	上游赋值节点的输出结果为一维数组时，获取具体某列数据。

内置变量取值案例

- 案例1

  上游赋值节点为Shell节点，最后一条输出结果为 `2021-03-28,2021-03-29,2021-03-30,2021-03-31,2021-04-01` ，此时，各变量的取值如下：

  > **说明** 由于输出结果为一维数组，数组元素个数为5（逗号分隔每个元素），因此for-each总遍历次数为5。

内置变量	第1次循环遍历的取值	第2次循环遍历的取值
`${dag.loopDataArray}`	`2021-03-28,2021-03-29,2021-03-30,2021-03-31,2021-04-01`	
`${dag.foreach.current}`	`2021-03-28`	`2021-03-29`

内置变量	第1次循环遍历的取值	第2次循环遍历的取值
`${dag.offset}`	0	1
`${dag.loopTimes}`	1	2
`${dag.foreach.current[3]}`	`2021-03-30`	

- 案例2

  上游赋值节点为ODPS SQL节点，最后一条select语句查询出两条数据：

```
+-----------------------------------------------+
| uid             | region | age_range | zodiac |
+-----------------------------------------------+
| 0016359810821   | 湖北省 | 30~40岁   | 巨蟹座 |
| 0016359814159   | 未知   | 30~40岁   | 巨蟹座 |
+-----------------------------------------------+
```

  此时，各变量的取值如下：

  > **说明** 由于输出结果为二维数组，数组行数为2，因此for-each总遍历次数为2。

内置变量	第1次循环遍历的取值	第2次循环遍历的取值

`${dag.loopDataArray}`	`+-----------------------------------------------+`
	`| uid  | region | age_range | zodiac |`
	`+-----------------------------------------------+`
	`| 0016359810821  | 湖北省 | 30~40岁  | 巨蟹座 |`
	`| 0016359814159  | 未知  | 30~40岁  | 巨蟹座 |`
	`+-----------------------------------------------+`	

`${dag.foreach.current}`	`0016359810821,湖北省,30~40岁,巨蟹座`	`0016359814159,未知,30~40岁,巨蟹座`
`${dag.offset}`	0	1
`${dag.loopTimes}`	1	2
`${dag.foreach.current[0]}`	`0016359810821`	`0016359814159`
`${dag.loopDataArray[1][0]}`	`0016359814159`	

6.11.2.2. 配置for-each节点

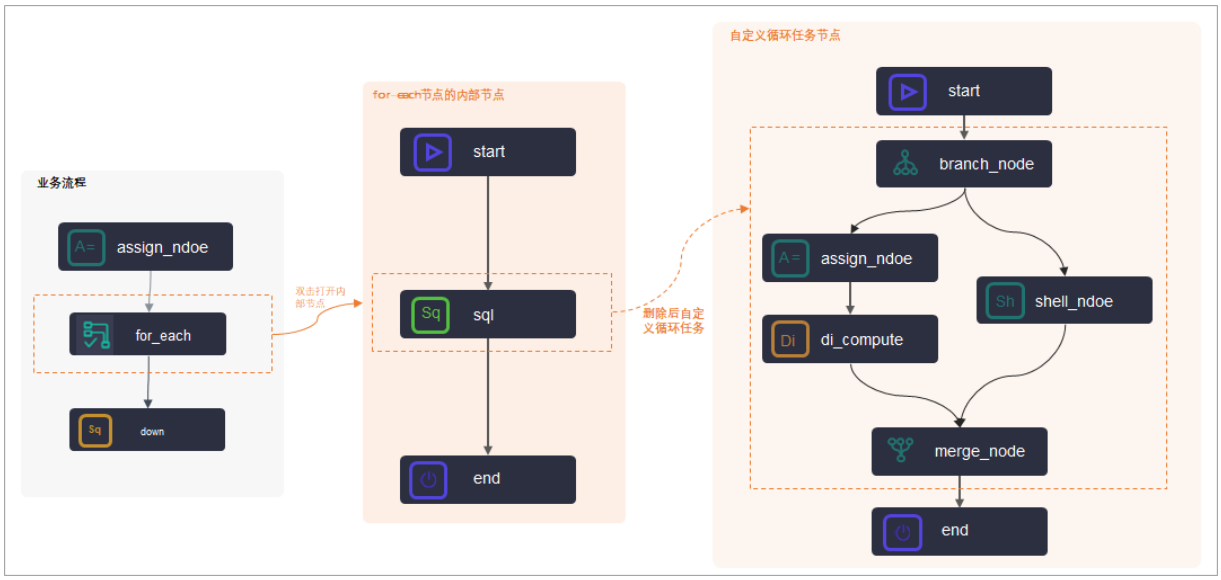
DataWorks为您提供遍历节点（for-each节点），您可以通过for-each节点来循环遍历赋值节点传递的结果集。同时您也可以重新编排for-each节点内部的业务流程。本文以一个具体示例，通过for-each节点2次循环遍历赋值节点输出结果，并在每次循环遍历时打印当前循环次数，为您介绍fo-each节点的逻辑原理与操作流程。

前提条件

您需要购买DataWorks标准版及以上版本，才可以使用for-each节点。

背景信息

DataWorks的for-each节点主要用于有循环遍历的场景，且需要与赋值节点联合使用，将赋值节点作为for-each节点的上游节点，将赋值节点的输出结果赋值给for-each节点后，一次次循环来遍历赋值节点的输出结果。此外，for-each节点也是拥有内部节点：**start节点**（循环开始节点）、**sql节点**、**end节点**。



您也可以自定义for-each遍历节点的内部业务流程，并通过for-each遍历节点提供的内置变量获取赋值节点传递的结果集。原理逻辑的详情请参见逻辑原理介绍。

使用限制与注意事项

- 上下游依赖

  for-each遍历节点需要遍历赋值节点传递的值，所以赋值节点需作为for-each节点的上游节点，for-each节点需要依赖赋值节点。

- 循环支持
  - 仅DataWorks标准版及以上版本支持使用for-each节点。

- for-each节点最多支持循环128次，如果超过了128次，则运行会报错。实际循环遍历次数由上游赋值节点实际输出控制。
  - 一维数组类型的输出，循环遍历次数即为一维数组元素的个数。

    例如，赋值节点的赋值语言为Shell或Python（Python2）时，输出结果为一维数组：`2021-03-28,2021-03-29,2021-03-30,2021-03-31,2021-04-01`，则for-each节点会循环5次完成遍历。
  - 二维数组类型的输出，循环遍历次数即为二维数组元素的行数。

    例如，赋值节点的赋值语言为OdpsSQL时，输出结果为二维数组：

    ```
    +------------------------------------------------+
    | uid             | region | age_range | zodiac |
    +------------------------------------------------+
    | 0016359810821   | 湖北省 | 30~40岁   | 巨蟹座 |
    | 0016359814159   | 未知   | 30~40岁   | 巨蟹座 |
    +------------------------------------------------+
    ```

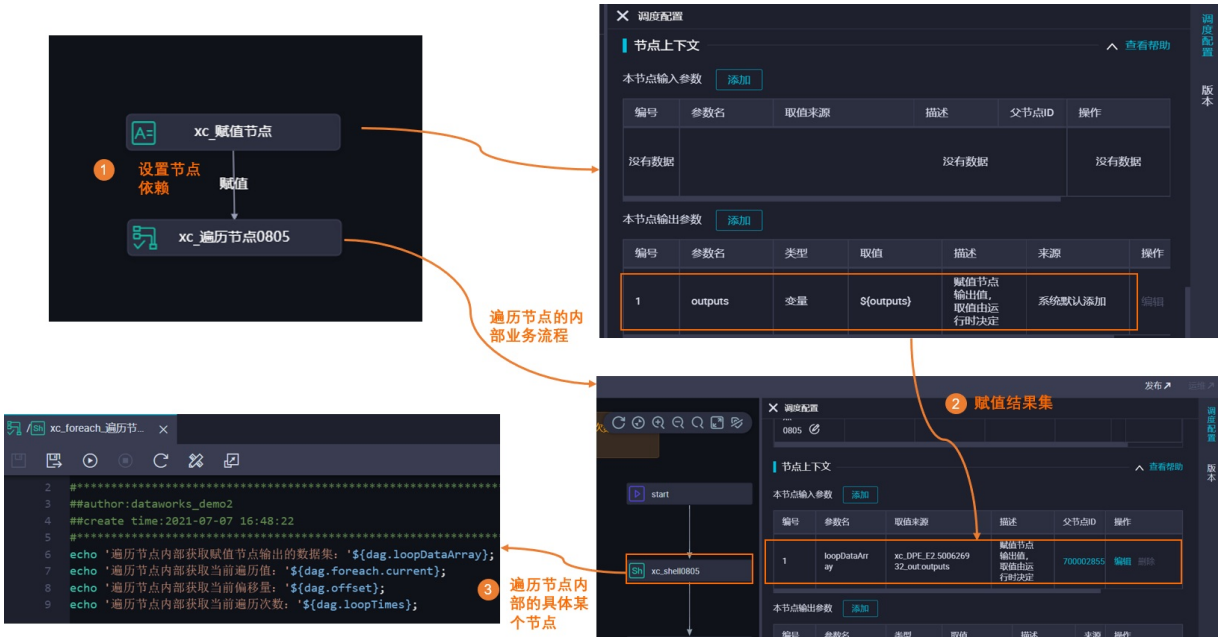
    则for-each节点会循环2次完成遍历。

- 内部节点
  - 您可以删除for-each节点的内部节点间的依赖关系，重新编排内部业务流程，但需要分别将**start节点**、**end节点**分别作为for-each节点内部业务流程的首末节点。
  - 在for-each节点的内部节点使用分支节点进行逻辑判断或者结果遍历时，需要同时使用归并节点。
- 调测运行
  - DataWorks为标准模式时，不支持在DataStudio界面直接测试运行for-each节点。

    如果您想测试验证for-each节点的运行结果，您需要将包含for-each节点的任务发布提交到运维中心，在运维中心页面运行for-each节点任务。
  - 在运维中心查看for-each节点的执行日志时，您需要右键实例，单击**查看内部节点**来查看内部节点的执行日志。

操作流程

使用遍历节点时，通常与赋值节点一起使用，操作流程如下所示。



1. 设置节点依赖关系

   for-each遍历节点需要依赖赋值节点。配置详情可参考文档：创建和配置业务流程。

2. 赋值结果集

   赋值节点自带的**节点上下文**输出参数**outputs**，需作为for-each遍历节点的**节点上下文**输入参数。配置详情可参考文档：配置赋值节点。

3. 遍历节点的内部节点获取参数

   根据业务需求自定义for-each遍历节点的内部业务流程，并在内部流程的节点中通过内置变量来获取所需参数值，运行循环遍历任务。内置变量的详情请参见内置变量，配置详情请参见配置for-each节点。

创建和配置业务流程

您需要创建一个上游为赋值节点，下游为for-each节点的业务流程：

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. 创建业务流程。
   i. 鼠标悬停至**+新建**图标，单击**业务流程**。
   ii. 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。

   iii. 单击**新建**。
3. 创建for-each节点。

i. 鼠标悬停至+新建图标，单击**通用** > **for-each**。

您也可以找到相应的业务流程，右键单击**通用**，选择**新建** > **for-each**。

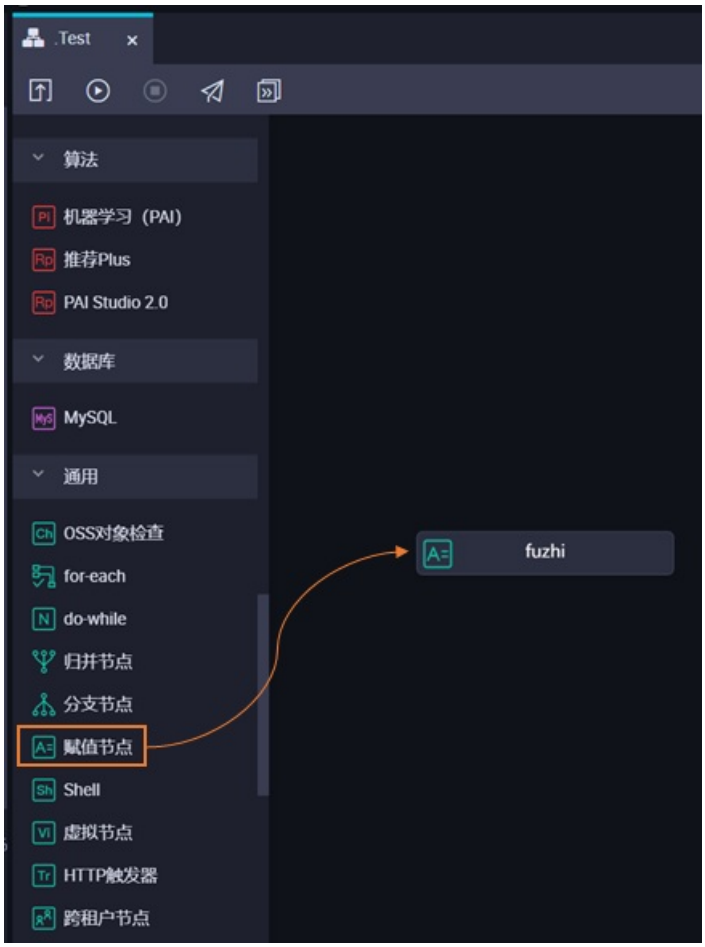
ii. 在**新建节点**对话框中，输入**节点名称**，并选择**目标文件夹**。

> 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。

iii. 单击**提交**。

4. 创建赋值节点，赋值节点的详情请参见赋值节点。

i. 在业务流程的编辑页面，鼠标单击**通用** > **赋值节点**并拖拽至右侧的编辑页面。



ii. 在**新建节点**对话框中，输入**节点名称**，并选择**目标文件夹**（默认在当前业务流程目录下）。

> 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。

iii. 单击**提交**。

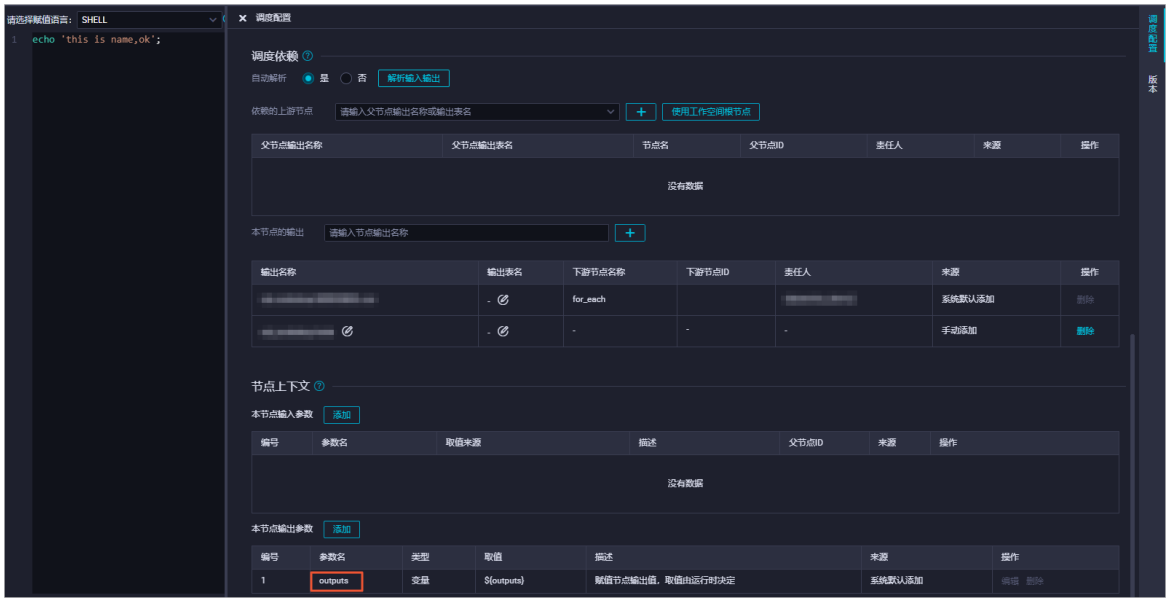
5. 通过拖拽连线，设置赋值节点为for-each节点的上游节点。

配置赋值节点

1. 双击赋值节点名称，打开节点的编辑页面。
2. 从**请选择赋值语言**列表中，选中**SHELL**。
3. 在节点的编辑页面，输入以下语句。

```
echo 'this is name,ok';
```

4. 单击节点编辑页面右侧的**调度配置**，在**节点上下文 > 本节点输出参数**区域查看默认输出的outputs参数。



5. 单击工具栏中的图标，保存赋值节点。
6. 提交赋值节点。

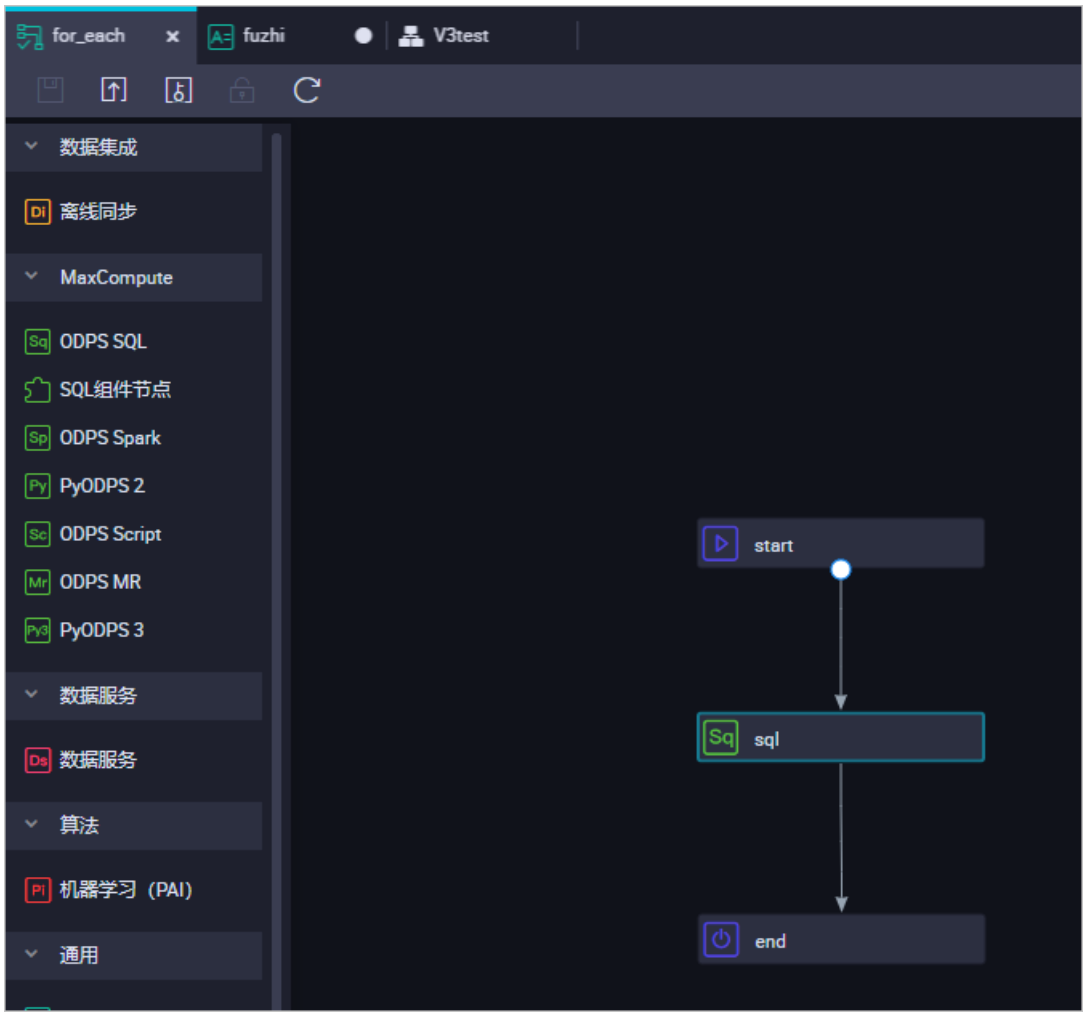
> **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   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，提交成功后，请单击右上角的**发布**。具体操作请参见发布任务。

配置for-each节点

1. 双击打开for-each节点的编辑页面，默认有start、SQL和end三个节点。

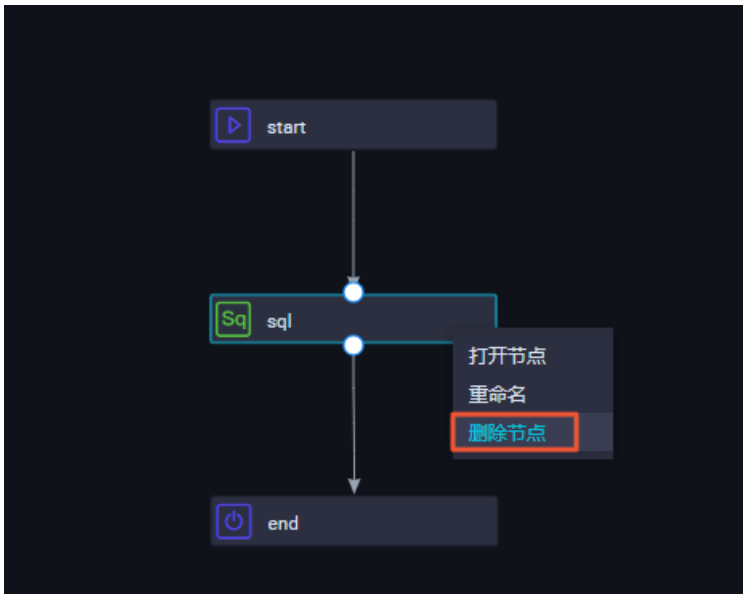


2. 删除SQL节点。

   您可以根据自身需求，选择第2个节点为不同的类型：

   - 如果您需要使用ODPS SQL节点，请跳过此步骤。
   - 如果您需要使用其它类型的节点（本文以使用Shell节点为例），请先删除默认产生的SQL节点。

i. 右键单击处于for-each节点中间的SQL节点，单击**删除节点**。

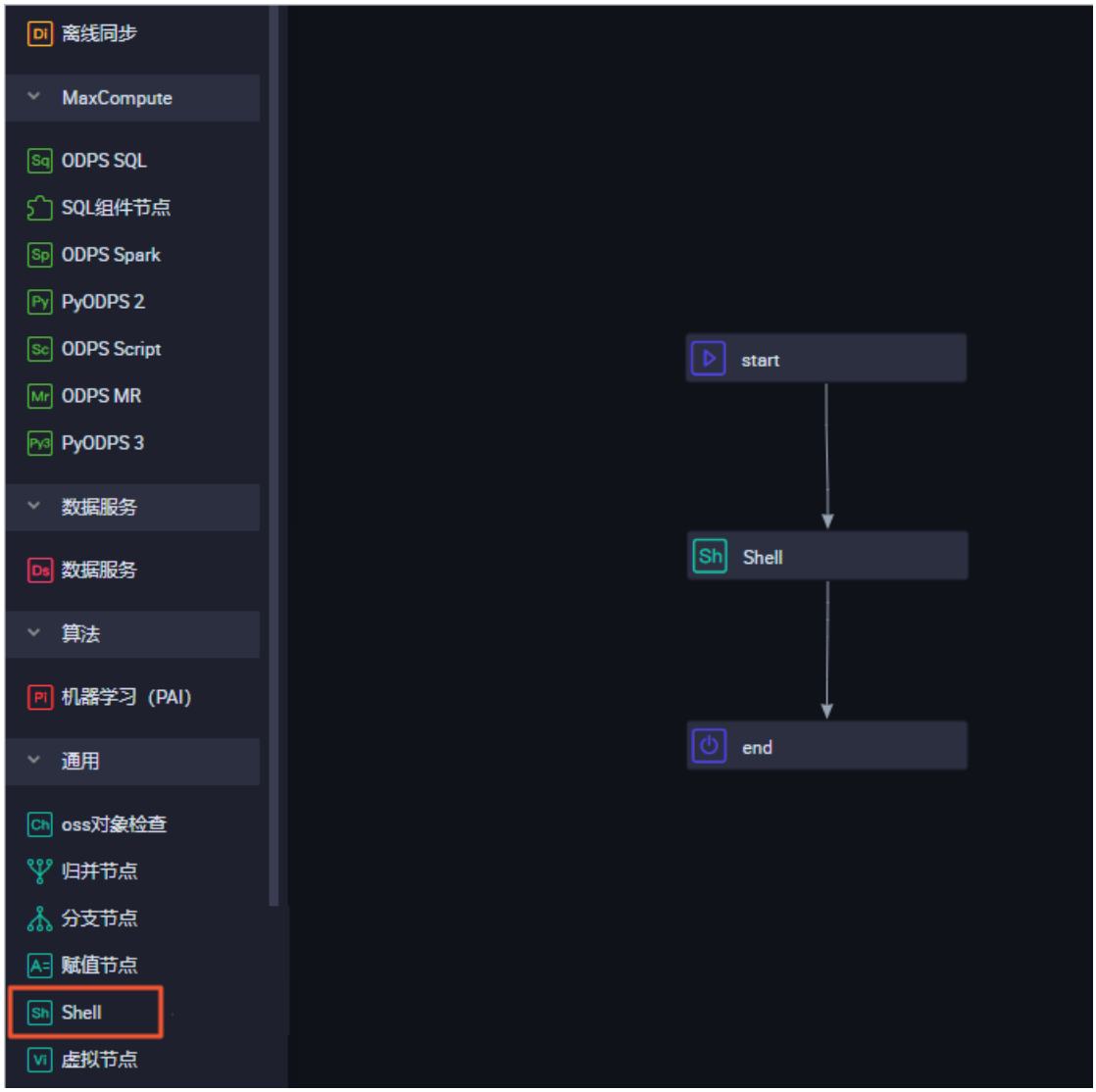


ii. 在**删除**对话框中，单击**确定**。

3. 创建并编辑Shell节点。

   您可以通过同样的方式，新建不同类型的节点。如果您使用的是默认的SQL节点，请跳过此步骤。

i. 鼠标单击**通用 > Shell**并拖拽至右侧的编辑页面。



ii. 在**新建节点**对话框中，输入**节点名称**。

> 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。

iii. 单击**提交**。

iv. 在for-each节点的编辑页面，通过拖拽连线，设置Shell节点的上游为start节点，下游为end节点。

v. 双击Shell节点，进入Shell节点的编辑页面。

vi. 输入以下代码。

```
echo ${dag.loopTimes} ----打印循环的次数。
```

> **说明**
> - for-each节点的start节点和end节点的逻辑是固定的，不可以进行编辑。
> - Shell节点中的代码修改后请务必保存，提交时不会进行提示。如果未保存，最新的代码不能及时更新。

for-each节点支持以下四种环境变量：

- ${dag.foreach.current}：当前遍历到的数据行。
- ${dag.loopDataArray}：输入的数据集。
- ${dag.offset}：偏移量。
- ${dag.loopTimes}：当前循环次数，值为${dag.offset}+1。

变量详情请参见内置变量和内置变量取值案例。

4. 配置for-each节点。
   i. 在for-each节点的编辑页面，单击右侧的**调度配置**。
   ii. 在**节点上下文 > 本节点输入参数**区域，单击默认参数名loopDataArray后的**编辑**。
   iii. 从**取值来源**列表中，选择上游赋值节点的outputs参数。

> **说明** 您在调度配置中添加上游赋值节点的依赖关系后，请手动添加取值来源。如果未添加取值来源，提交节点时会报错。

   iv. 单击**保存**。

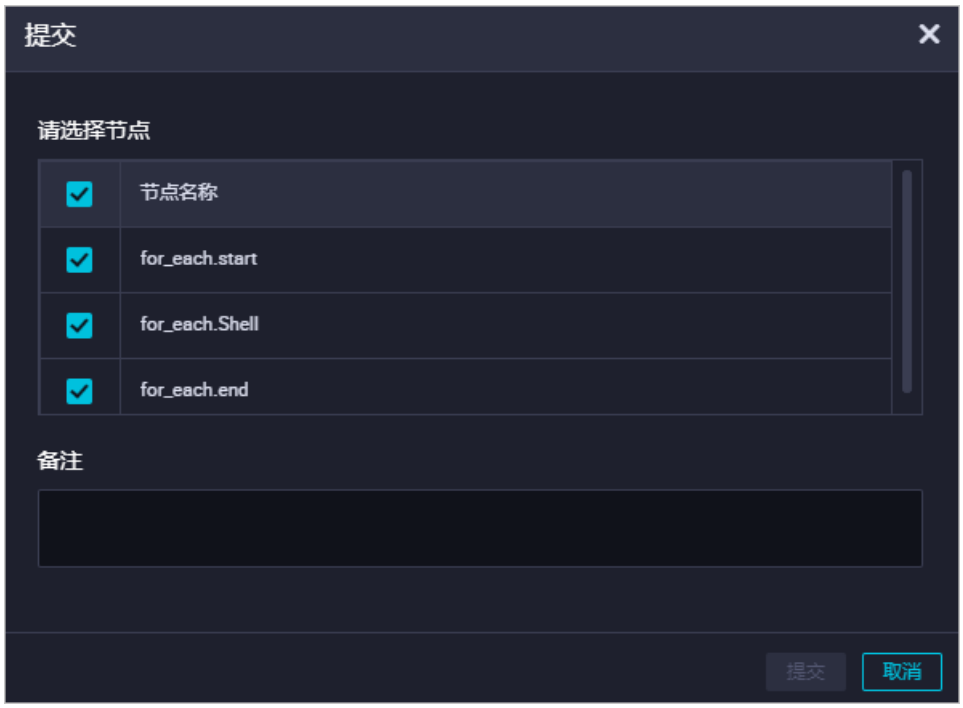
5. 单击工具栏中的图标，保存for-each节点。

6. 提交for-each节点。

> **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标。

ii. 在**提交**对话框中，选中需要提交的节点，输入**备注**。



iii. 单击**提交**。

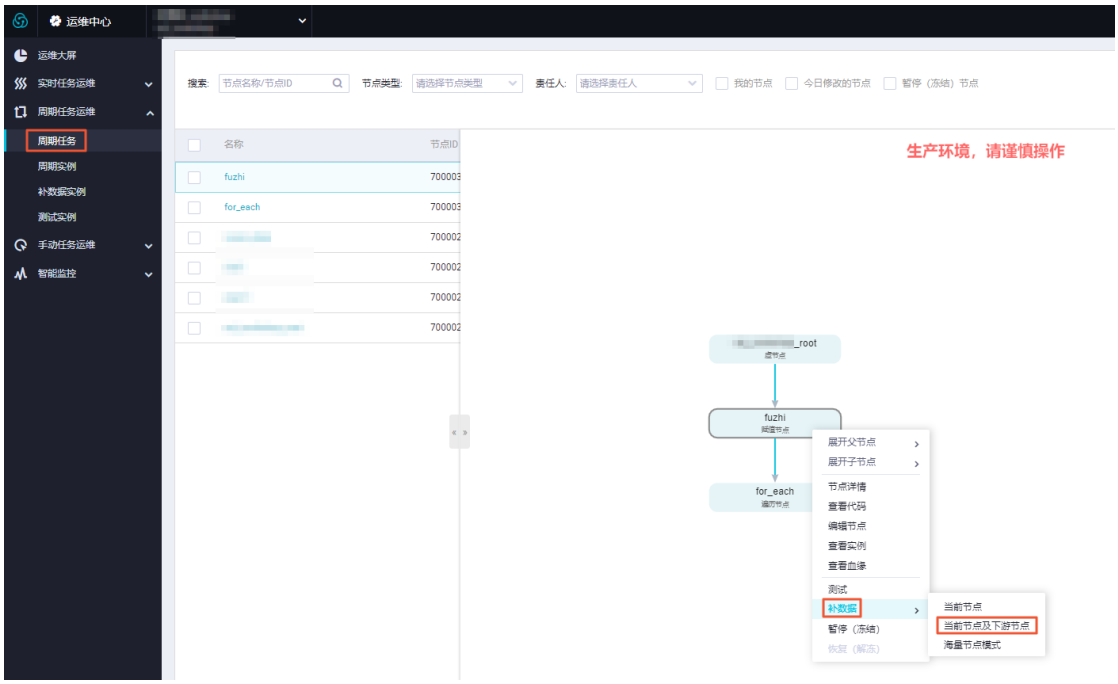
如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

7. 测试节点，并查看结果。

i. 单击页面右上方的**运维**，进入**运维中心**。

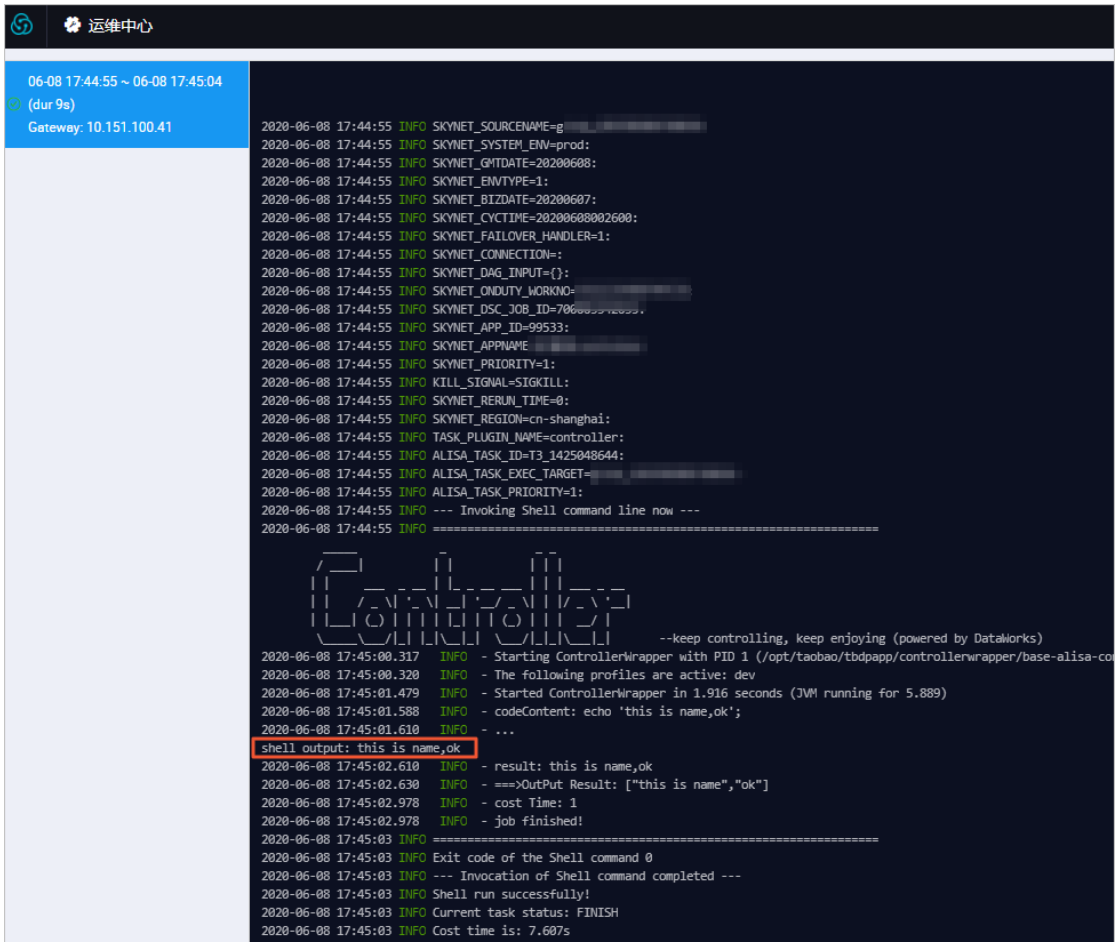
ii. 在左侧导航栏，单击**周期任务运维 > 周期任务**。

iii. 选中相应的节点，在右侧的DAG图中，右键单击赋值节点，选中**补数据 > 当前节点及下游节点**。



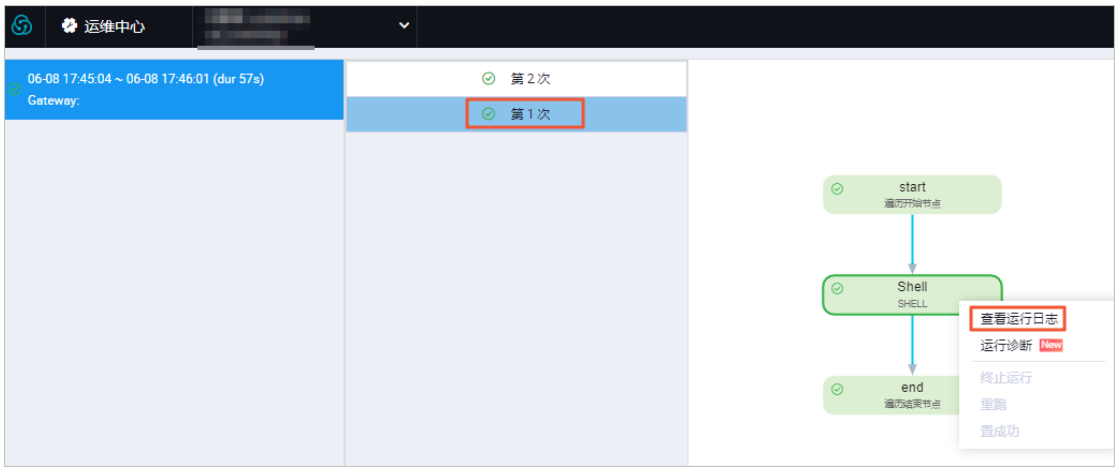
iv. 刷新**补数据实例**页面，待补数据实例运行成功后，单击实例后的**DAG图**。

v. 右键单击赋值节点，选中**查看运行日志**，确认赋值结果。

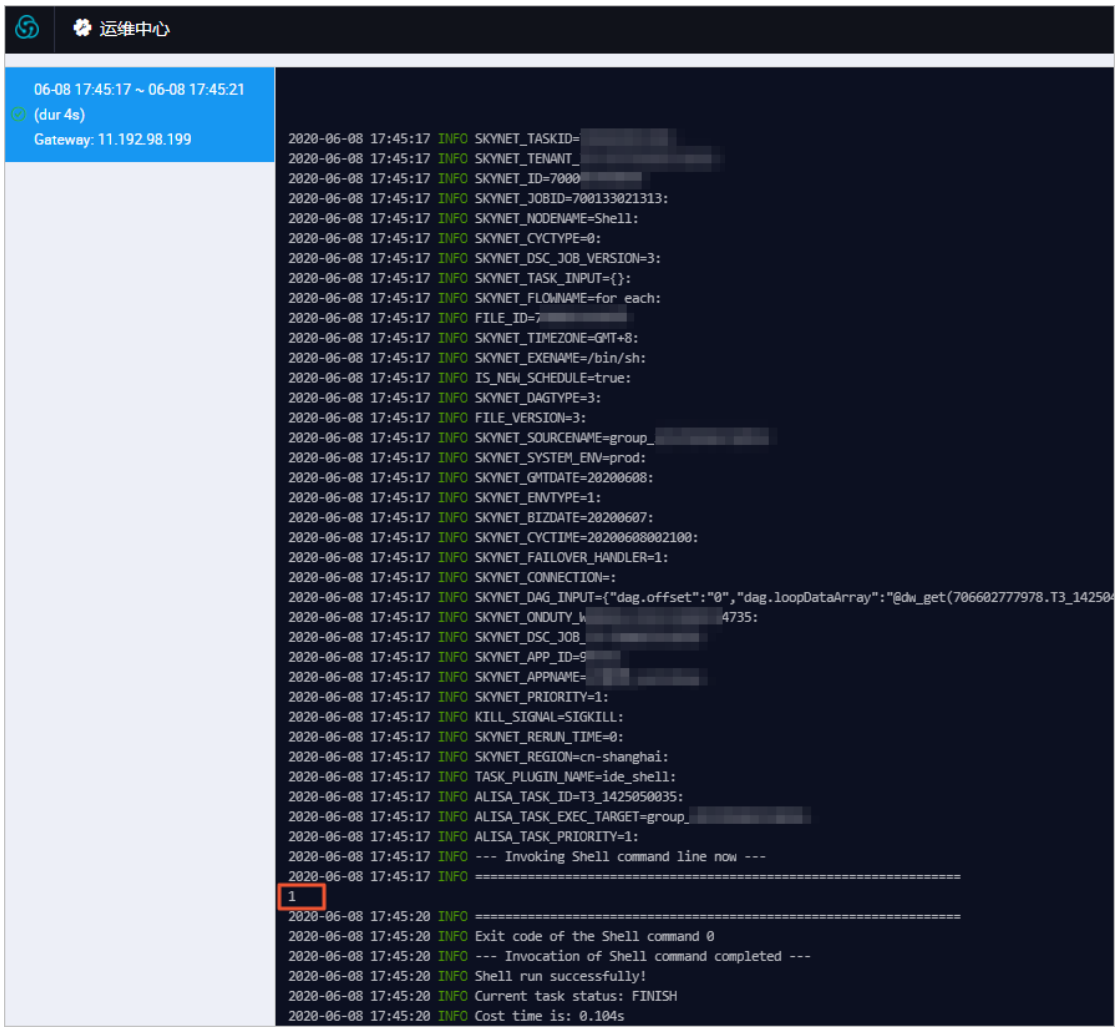


vi. 在**补数据实例**页面，右键单击遍历节点，选中**查看内部节点**。

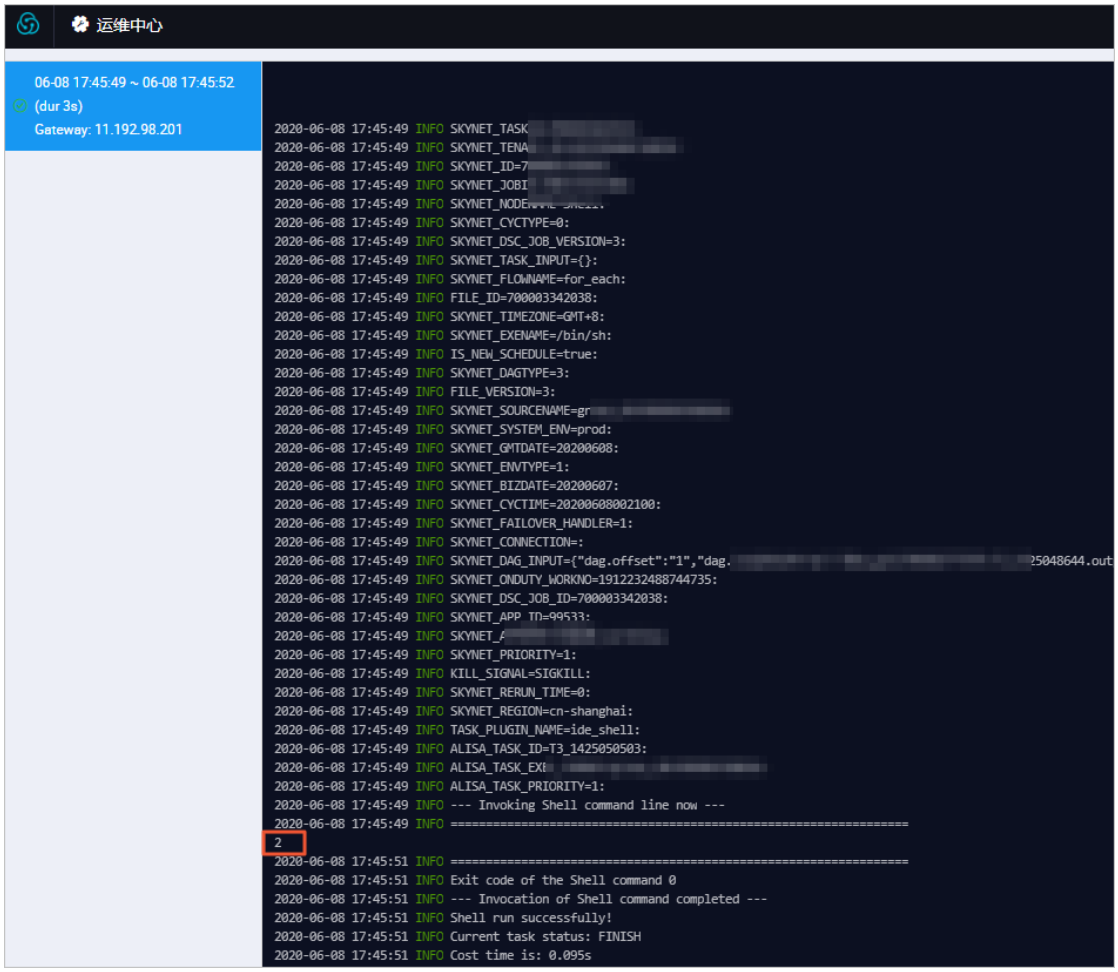
vii. 在内部节点页面，单击左侧的**第1次**，并右键单击Shell节点，选中**查看运行日志**。



在运行日志页面，查看第1次循环时，Shell节点的日志。



viii. 以同样的方式，查看第2次循环时，Shell节点的日志。



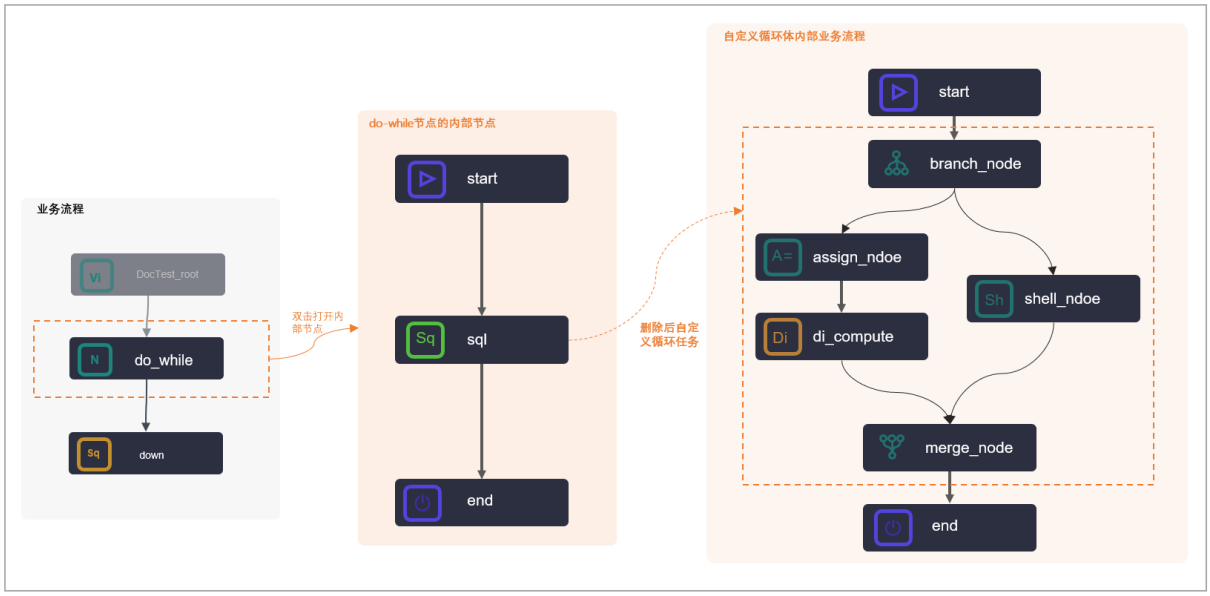
6.11.3. do-while节点

6.11.3.1. 逻辑原理介绍

DataWorks为您提供循环节点（do-while节点），您可以重新编排do-while节点内部的业务流程，将需要循环执行的逻辑写在节点内，再编辑end循环判断节点来控制是否退出循环。同时您也可以结合赋值节点来循环遍历赋值节点传递的结果集。本文为您介绍do-while节点的组成与应用逻辑。

节点组成

DataWorks的do-while节点是包含内部节点的一种特殊节点，您在创建完成do-while节点时，同时也自动创建完成了三个内部节点：**start节点**（循环开始节点）、**sql节点**（循环任务节点）、**end节点**（循环结束判断节点），通过内部节点组织成内部节点流程，实现任务的循环运行。



如上图所示：

- **start节点**

  是内部节点的开始节点，不承载具体的任务代码。

- **sql节点**

  DataWorks默认为您创建好了一个SQL类型的内部任务运行节点，您也可以删除默认的**sql节点**后，自定义内部循环任务的运行节点。

  - 您的循环任务是SQL类型的任务，则可以直接双击默认的**sql节点**，进入节点的代码开发页面开发循环任务代码。
  - 您的循环任务比较复杂，您可以在内部节点流程中新建其他任务节点，并根据实际情况重新构建节点的运行流程。

    通常循环任务的业务流程会与赋值节点、分支节点、归并节点联合使用，典型应用场景说明请参见典型应用：与赋值节点联合使用。

    > **说明** 自定义循环任务节点时，您可以删除内部节点间的依赖关系，重新编排循环节点内部业务流程，但需要分别将**start节点**、**end节点**分别作为do-while节点内部业务流程的首末节点。

- **end节点**

  - **end节点**是do-while节点的循环判断节点，来控制do-while节点循环次数，其本质上是一个赋值节点，输出 `true` 和 `false` 两种字符串，分别代表继续下一个循环和不再继续循环。
  - **end节点**支持使用ODPS SQL、SHELL和Python（Python2）三种语言进行循环判断代码开发，同时do-while节点为您提供了便利的内置变量，便于您进行end代码开发。内置变量的介绍请参见内置变量和变量取值案例，不同语言开发的样例代码请参见案例1：end节点代码样例。

使用限制与注意事项

- 循环支持
  - 仅DataWorks标准版及以上版本支持使用do-while节点。

- do-while节点最多支持循环128次，**end节点**控制循环次数时，如果超过了128次，则运行会报错。

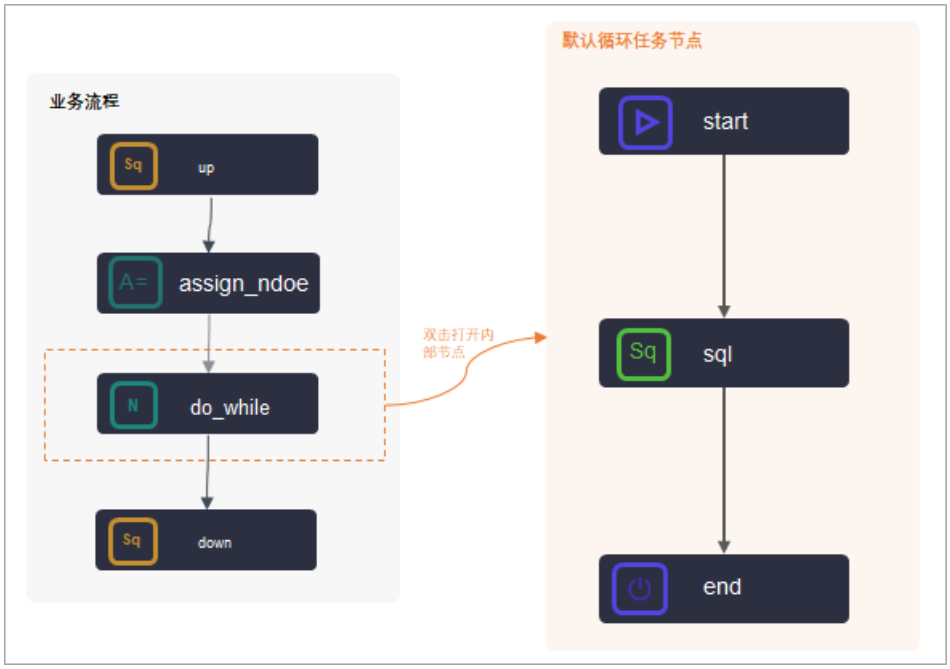
- 内部节点
  - 自定义循环任务节点时，您可以删除内部节点间的依赖关系，重新编排循环节点内部业务流程，但需要分别将**start节点**、**end节点**分别作为do-while节点内部业务流程的首末节点。
  - 在do-while节点的内部节点使用分支节点进行逻辑判断或者结果遍历时，需要同时使用归并节点。
  - do-while节点的内部节点**end节点**在代码开发时，不支持添加注释。

- 调测运行
  - DataWorks为标准模式时，不支持在DataStudio界面直接测试运行do-while节点。

    如果您想测试验证do-while节点的运行结果，您需要将包含do-while节点的任务发布提交到运维中心，在运维中心页面运行do-while节点任务。如果您在do-while节点内使用了赋值节点传递的值，请在运维中心测试时，同时运行赋值节点和循环节点。
  - 在运维中心查看do-while节点的执行日志时，您需要右键实例，单击**查看内部节点**来查看内部节点的执行日志。

典型应用：与赋值节点联合使用

do-while节点常常与赋值节点联合使用，如下图所示。



与赋值节点联合使用时：

- 您需要将赋值节点的输出作为赋值节点的本节点输入，且与赋值节点做好上下游依赖关系的配置，其他配置注意事项请参见案例2：与赋值节点联合使用。
- 与赋值节点联合使用时，可以使用一些内置变量来获取当前已循环次数、赋值参数值等循环变量值，详情请参见内置变量。

内置变量

DataWorks的do-while节点，通过内部节点来实现循环运行任务，每次任务循环运行时，您可以通过一些内置的变量来获取当前已循环次数和偏移量。

内置变量	含义	取值
`${dag.loopTimes}`	当前已循环次数	第一次循环为1、第二次为2、第三次为3...第n次为n。
`${dag.offset}`	偏移量	第一次循环为0、第二次为1、第三次为2...第n次为n-1。

如果您联合使用了赋值节点，则还可以通过以下方式来获取赋值参数值和循环变量参数。

> **说明** 以下以变量示例中，*input*是do-while节点中自定义的**本节点输入参数**名称，实际使用时，需替换为您真实的名称。

内置变量	含义
`${dag.input}`	上游赋值节点传递的数据集。
`${dag.input[${dag.offset}]}`	循环节点内部获取当前循环的数据行。
`${dag.input.length}`	循环节点内部获取数据集长度。

变量取值案例

- 案例1

  上游赋值节点为Shell节点，最后一条输出结果为`2021-03-28,2021-03-29,2021-03-30,2021-03-31,2021-04-01`，此时，各变量的取值如下：

内置变量	第1次循环时取值	第2次循环时取值
`${dag.input}`	`2021-03-28,2021-03-29,2021-03-30,2021-03-31,2021-04-01`	
`${dag.input[${dag.offset}]}`	`2021-03-28`	`2021-03-29`
`${dag.input.length}`	5	
`${dag.loopTimes}`	1	2
`${dag.offset}`	0	1

- 案例2

  上游赋值节点为ODPS SQL节点，最后一条select语句查询出两条数据：

```
+-----------------------------------------------+
| uid             | region | age_range | zodiac |
+-----------------------------------------------+
| 0016359810821   | 湖北省 | 30~40岁   | 巨蟹座 |
| 0016359814159   | 未知   | 30~40岁   | 巨蟹座 |
+-----------------------------------------------+
```

此时，各变量的取值如下：

内置变量	第1次循环时取值	第2次循环时取值
`${dag.input}`	+----------+----------+----------+----------+ | uid | region | age_range | zodiac | +----------+----------+----------+----------+ | 0016359810821 | 湖北省 | 30~40岁 | 巨蟹座 | | 0016359814159 | 未知 | 30~40岁 | 巨蟹座 | +----------+----------+----------+----------+	
`${dag.input[${dag.offset}]}`	`0016359810821,湖北省,30~40岁,巨蟹座`	`0016359814159,未知,30~40岁,巨蟹座`
`${dag.input.length}`	2 **说明** 二维数组的行数为数据集长度，当前赋值节点输出的二维数组行数为2。	
`${dag.input[0][1]}` **说明** 二维数组的第一行第一列的取值。	`0016359810821`	
`${dag.loopTimes}`	1	2
`${dag.offset}`	0	1

案例1：end节点代码样例

end节点支持使用ODPS SQL、SHELL和Python（Python2）三种语言进行循环判断代码开发，以下为您示例三种不同语言下，典型的代码样例。

- 使用ODPS SQL语言时：

```
SELECT  CASE
 WHEN COUNT(1) > 0 AND ${dag.offset}<= 9
  THEN true
  ELSE false
 END
FROM  xc_dpe_e2.xc_rpt_user_info_d  where dt='20200101';
```

end节点示例代码中将表行数和迁移量与固定值比较，来限制do-while节点整体的循环次数。

- 使用Shell语言时：

```
if [ ${dag.loopTimes} -lt 5 ];
then
     echo "True"
else
     echo "False"
fi
```

将循环次数 `${dag.loopTimes}` 和5进行比较，来限制do-while节点整体的循环次数。

例如：第一次循环 `${dag.loopTimes}` 的值为1、第二次为2，以此类推，第五次为5。至此end节点的输出结果为false，do-while节点退出循环。

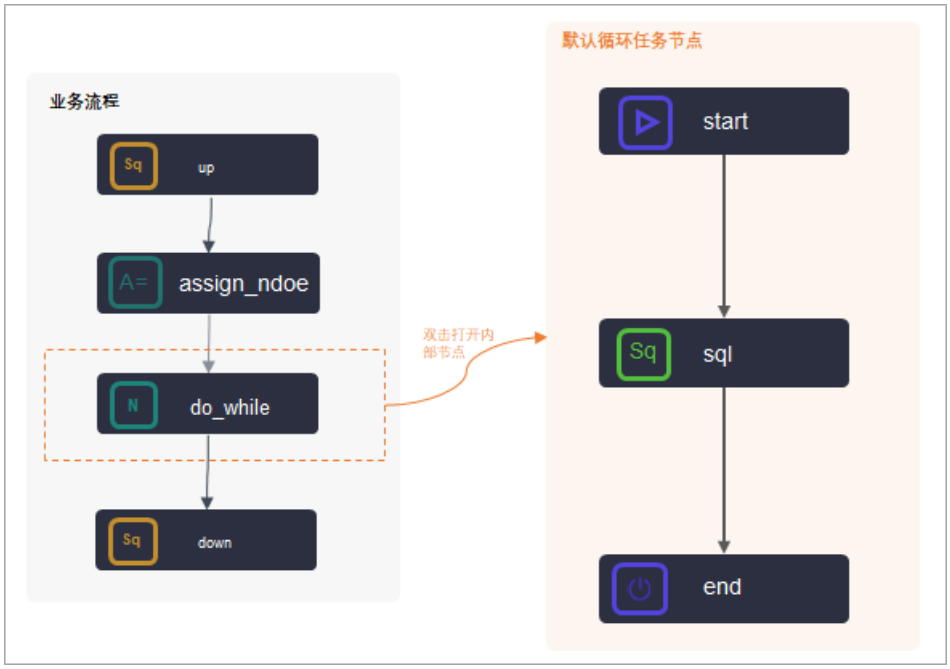
- 使用Python（Python2）语言时：

```
if ${dag.loopTimes}<${dag.input.length}:
   print True;
else
   print False;
# 如果end节点输出True，则继续下一个循环。
# 如果end节点输出False，则终止循环。
```

代码中把循环次数 `${dag.loopTimes}` 和赋值节点传递的数据集行数进行比较，来限制do-while节点整体的循环次数。

案例2：与赋值节点联合使用

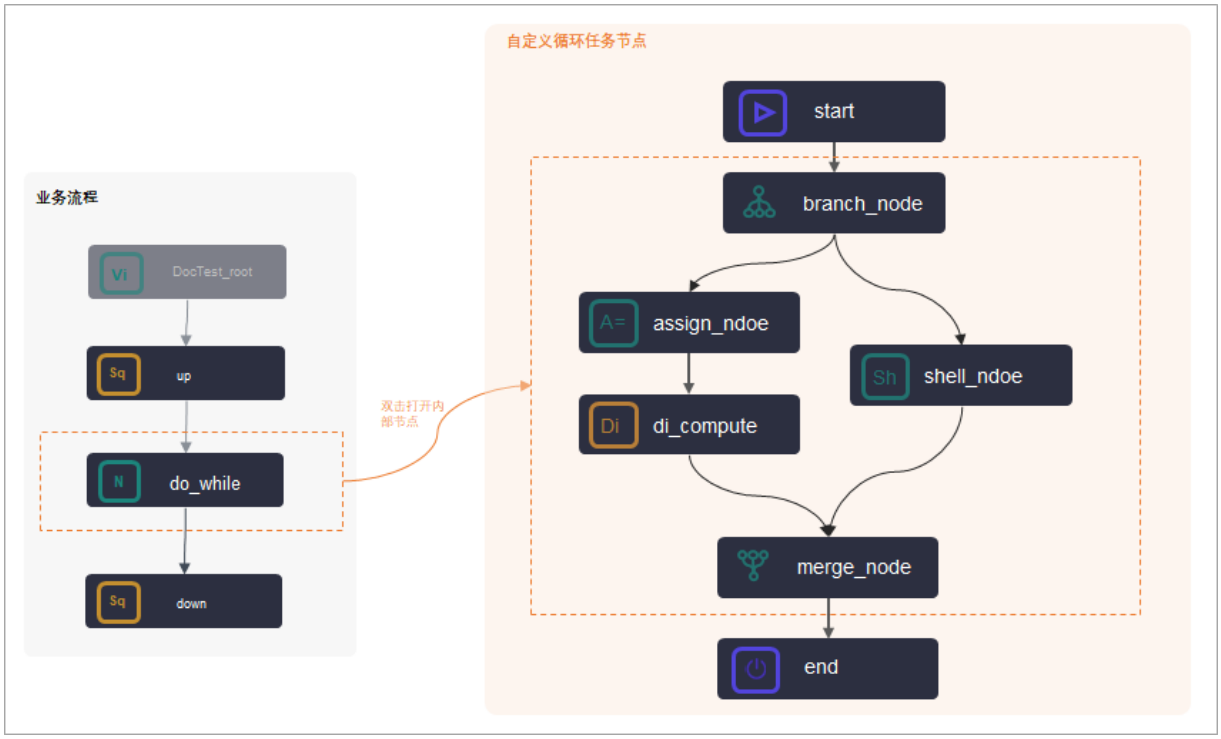
do-while节点与赋值节点联合使用时，典型的应用场景和注意事项如下所示。



应用场景	注意事项	配置案例
使用 `do-while节点` 进行循环任务时，内部节点的循环任务在每一次循环时，需要获取使用上游节点（ `up节点` ）的输出参数时，此时可以使用赋值节点（ `assign_node` ）。	• 依赖关系 do-while节点需要依赖上游赋值节点（assign_node）。 **说明** 如上图所示，依赖关系是do-while节点依赖赋值节点，而非do-while的内部循环任务节点（ `sql节点` ）依赖赋值节点。 • 上下文参数 ◦ 赋值节点需将输出参数作为赋值节点（ `assign_node` ）的**本节点输出参数**。 ◦ do-while的内部循环任务节点（ `sql节点` ）需将赋值节点的输出参数添加为**本节点输入参数**。 **说明** 上下文关系设置，需设置内部的循环任务节点，而非do-while节点。	

案例3：与分支节点、归并节点联合应用

与分支节点和归并节点联合应用时，典型的应用场景和注意事项如下所示。



应用场景	注意事项
do-while节点内部需要进行逻辑判断或者结果遍历时，此时可以在do-while节点的内部节点中自定义循环任务节点，并使用分支节点（ `branch_node` 和归并节点（ `merge_node` ））。	在do-while节点内部，分支节点（ `branch_node` ）需要和归并节点（ `merge_node` ）同时使用。

6.11.3.2. 配置do-while节点

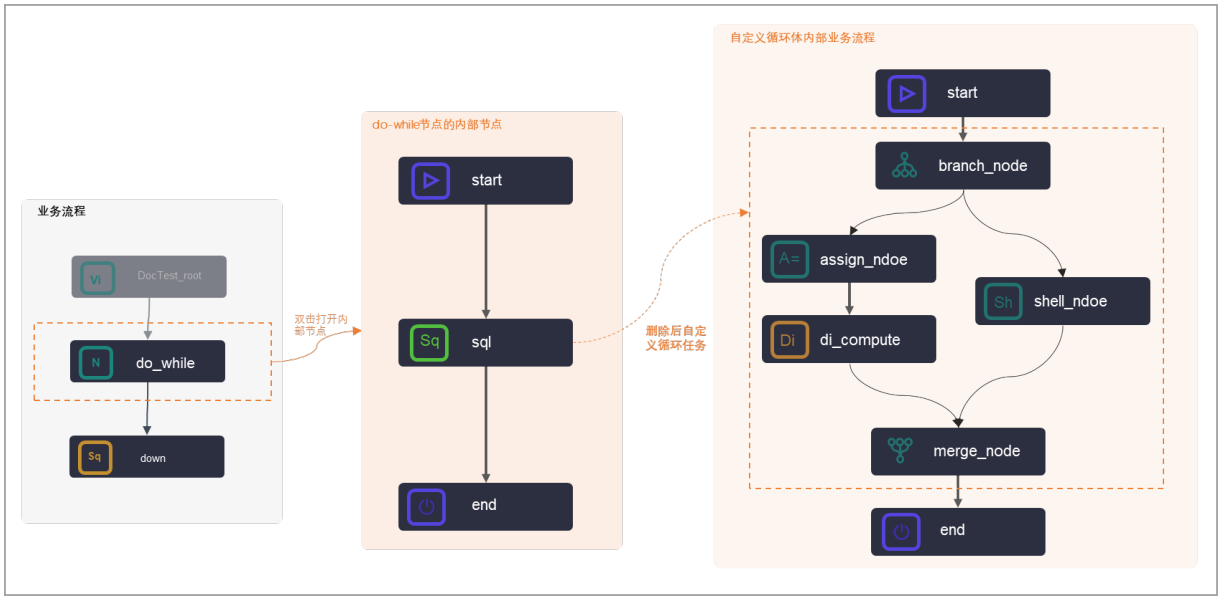
DataWorks为您提供循环节点（do-while节点），您可以重新编排do-while节点内部的业务流程，将需要循环执行的逻辑写在节点内，再编辑end循环判断节点来控制是否退出循环。同时您也可以结合赋值节点来循环遍历赋值节点传递的结果集。本文通过实现简单和复杂场景的示例，为您介绍如何配置do-while节点。

前提条件

您需要购买DataWorks标准版及以上版本，才可以使用do-while节点。

背景信息

DataWorks的do-while（循环）节点是包含内部节点的一种特殊节点，您在创建完成do-while节点时，同时也自动创建完成了三个内部节点：**start节点**（循环开始节点）、**sql节点**（循环任务节点）、**end节点**（循环结束判断节点），通过内部节点组织成内部节点流程，实现任务的循环运行。



您也可以自定义循环任务节点，并通过do-while节点提供的内置变量来编写控制循环次数的end节点代码。原理逻辑的详情请参见逻辑原理介绍，您可根据实际情况规划业务流程，do-while节点的配置操作请参见下文的操作步骤。

使用限制与注意事项

- 循环支持
  - 仅DataWorks标准版及以上版本支持使用do-while节点。
  - do-while节点最多支持循环128次，**end节点**控制循环次数时，如果超过了128次，则运行会报错。
- 内部节点
  - 自定义循环任务节点时，您可以删除内部节点间的依赖关系，重新编排循环节点内部业务流程，但需要分别将**start节点**、**end节点**分别作为do-while节点内部业务流程的首末节点。
  - 在do-while节点的内部节点使用分支节点进行逻辑判断或者结果遍历时，需要同时使用归并节点。
  - do-while节点的内部节点**end节点**在代码开发时，不支持添加注释。
- 调测运行
  - DataWorks为标准模式时，不支持在DataStudio界面直接测试运行do-while节点。

    如果您想测试验证do-while节点的运行结果，您需要将包含do-while节点的任务发布提交到运维中心，在运维中心页面运行do-while节点任务。如果您在do-while节点内使用了赋值节点传递的值，请在运维中心测试时，同时运行赋值节点和循环节点。
  - 在运维中心查看do-while节点的执行日志时，您需要右键实例，单击**查看内部节点**来查看内部节点的执行日志。

配置流程

1. 配置节点依赖

   do-while节点需要依赖赋值节点。

2. 赋值结果集

   赋值节点自带的**节点上下文**输出参数**outputs**，需作为do-while循环节点的**节点上下文**输入参数。

3. do-while循环节点的内部节点获取参数

   根据业务需求自定义do-while循环节点的内部业务流程，并在内部流程的节点中通过变量来获取所需参数值。

创建do-while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. 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，鼠标悬停至**+新建**图标，单击**通用 > do-while**。

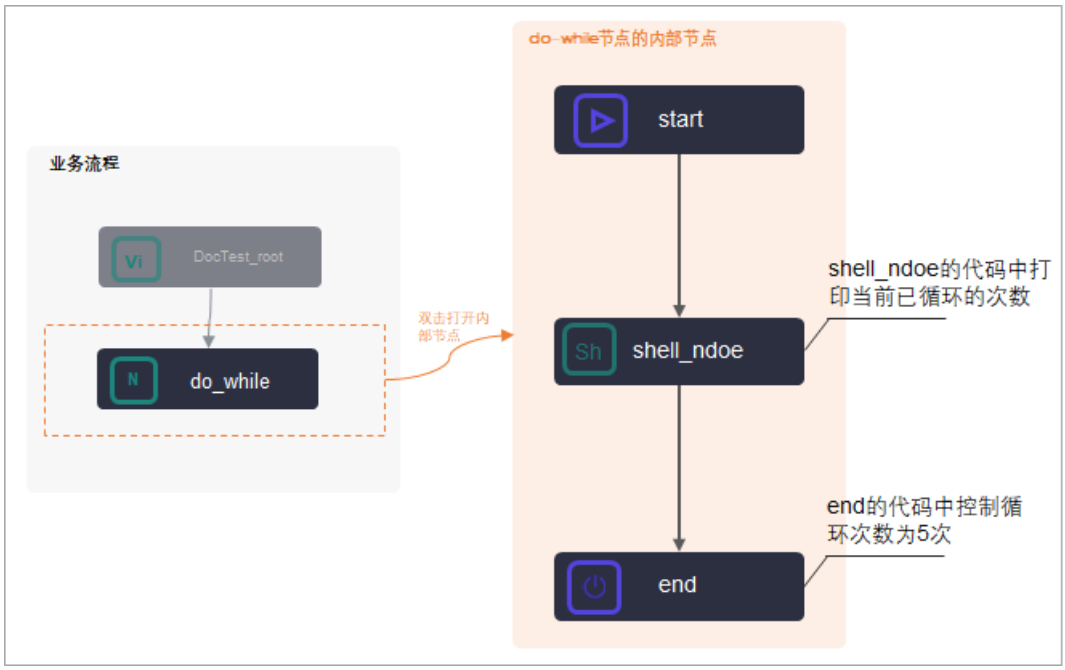
   您也可以打开相应的业务流程，右键单击**通用**，选择**新建 > do-while**。
3. 在**新建节点**对话框中，输入**节点名称**，并选择**目标文件夹**。

   > **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。

4. 单击**提交**。

do-while节点的简单示例

本节为您介绍如何使用循环节点循环5次，并在每次循环中打印出当前的循环次数的端到端操作步骤。



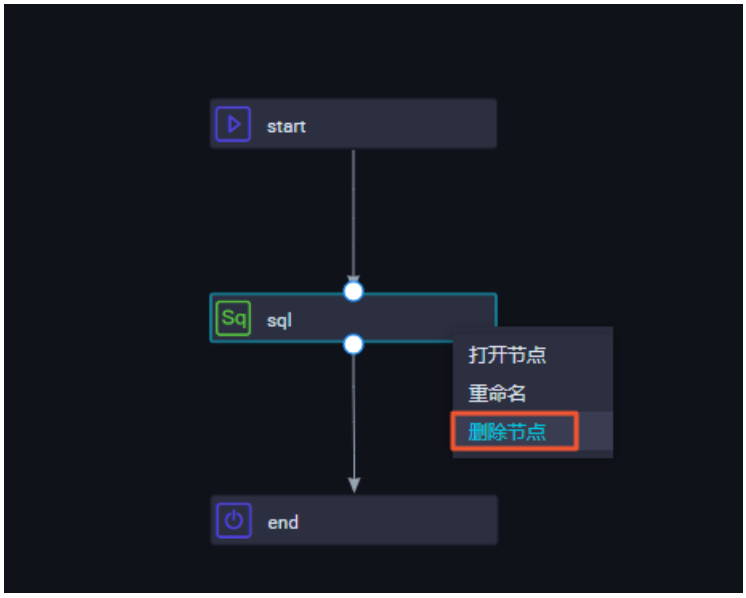
1. 双击do-while节点名称，进入内部节点页面。

   do-while节点默认有start、sql和end三个节点：

   - start节点是一个循环开始的标记节点，并无业务作用。
   - sql节点是DataWorks提供的一个业务处理节点示例，此处需要将其删除，替换为自己的业务处理Shell节点（打印当前循环次数）。
   - end节点具有标记循环结束和判断是否开启下一次循环的功能，此处用于定义do-while节点的结束条件。

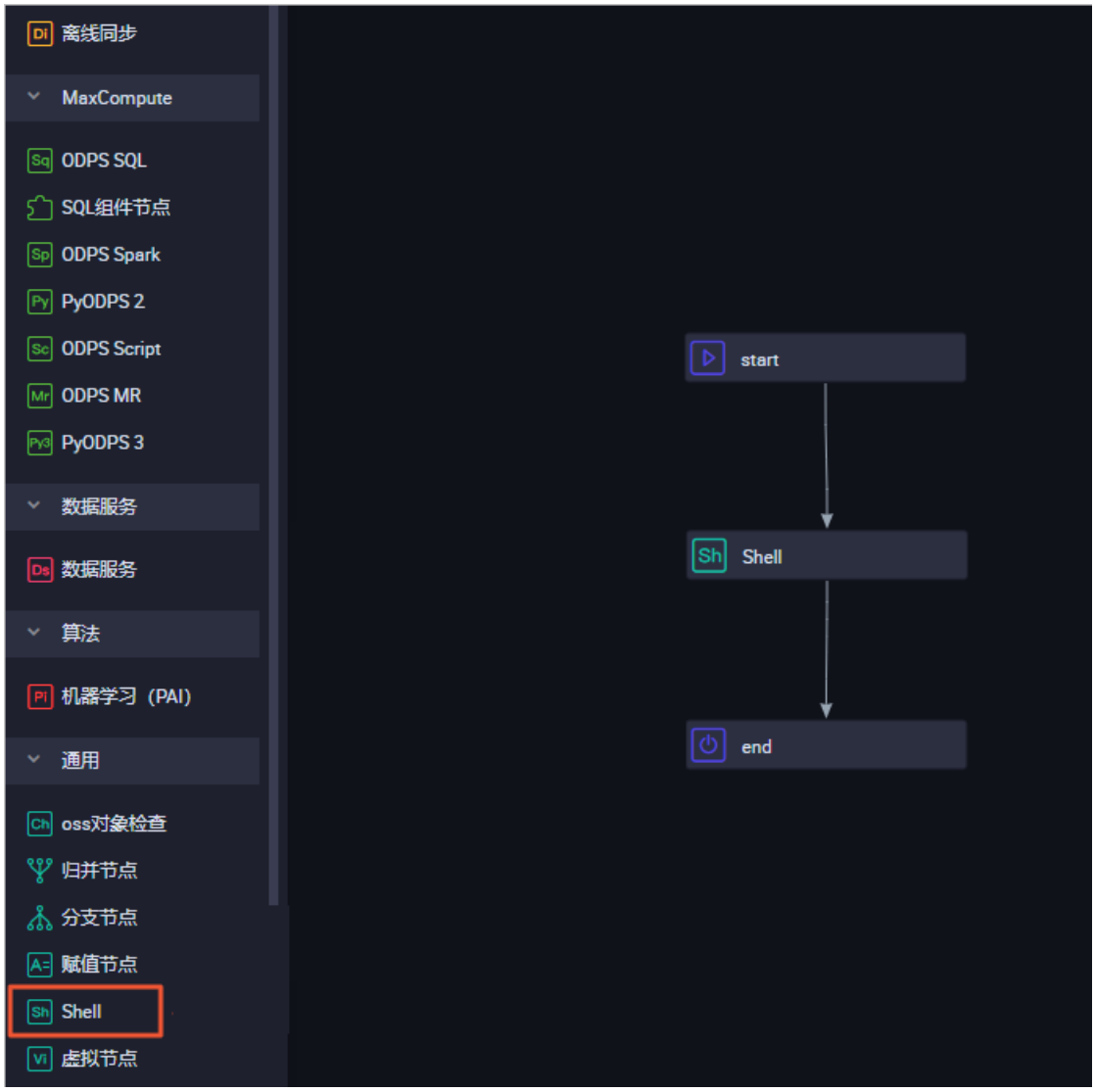
2. 删除sql节点。

   i. 右键单击处于do-while节点中间的sql节点，单击**删除节点**。

   ii. 在**删除**对话框中，单击**确定**。

3. 创建并编辑循环任务节点，本示例使用Shell节点。
   i. 鼠标单击**通用 > Shell**并拖拽至右侧的编辑页面。



   ii. 在**新建节点**对话框中，输入**节点名称**。

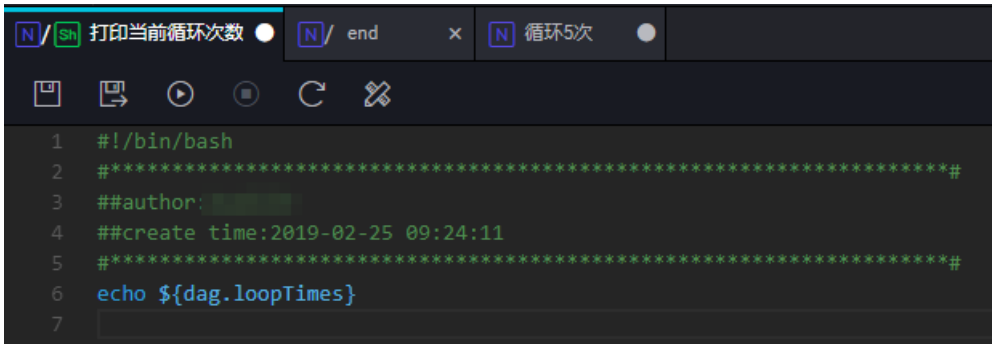
> 注意　节点名称必须是大小写字母、中文、数字、下划线（_）以及小数点（.），且不能超过128个字符。

   iii. 单击**提交**。
   iv. 在do-while节点的编辑页面，通过拖拽连线，设置Shell节点的上游为start节点，下游为end节点。
   v. 双击Shell节点，进入Shell节点的编辑页面。

vi. 输入以下代码。

```
echo ${dag.loopTimes} ----打印循环的次数。
```

- ${dag.loopTimes}变量是系统的保留变量，代表当前的循环次数，从1开始，do-while的内部节点可以直接引用该变量。更多内置变量请参见内置变量和变量取值案例。
- Shell节点中的代码修改后请务必保存，提交时不会进行提示。如果未保存，最新的代码不能及时更新。



4. 配置end节点，控制循环次数。
   i. 双击打开end节点的编辑页面。
   ii. 在**请选择赋值语言**下拉列表中，选中**Python**。
   iii. 输入以下代码，定义do-while节点的结束条件。

```
if ${dag.loopTimes}<5:
 print True;
else:
 print False;
```

   - ${dag.loopTimes}变量是系统的保留变量，代表当前的循环次数，从1开始，do-while的内部节点可以直接引用该变量。更多内置变量请参见内置变量和变量取值案例。
   - 代码中把 `dag.loopTimes` 和5进行比较，可以限制整体的循环次数。第一次循环dag.loopTimes为1、第二次为2，以此类推，第五次为5。至此表达式${dag.loopTimes}<5结果为false，退出循环。
5. 单击节点编辑页面右侧的**调度配置**，配置节点的调度属性，详情请参见配置基础属性。
6. 单击工具栏中的图标。
7. 提交do-while节点。

> **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标。
   ii. 在**提交**对话框中，选中需要提交的节点，输入**备注**。
   iii. 单击**提交**。

如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

8. 测试节点，并查看结果。

> **说明** DataWorks为标准模式时，不支持在DataStudio界面直接测试运行do-while节点。
>
> 如果您想测试验证do-while节点的运行结果，您需要将包含do-while节点的任务发布提交到运维中心，在运维中心页面运行do-while节点任务。如果您在do-while节点内使用了赋值节点传递的值，请在运维中心测试时，同时运行赋值节点和循环节点。

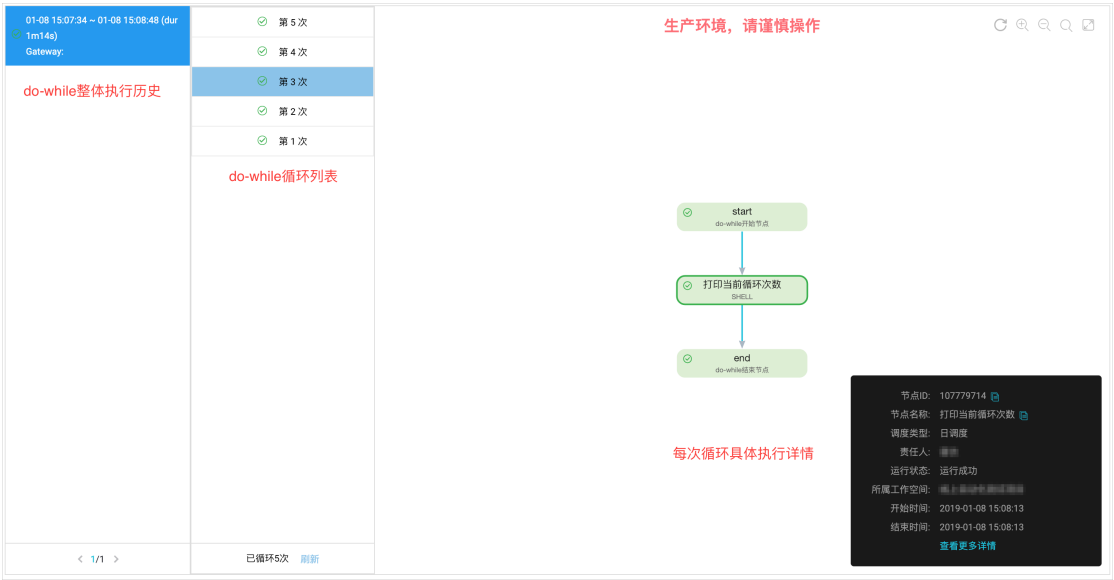
i. 单击页面右上方的**运维**，进入**运维中心**。

ii. 在左侧导航栏，单击**周期任务运维 > 周期任务**。

iii. 选中相应的节点，在右侧的DAG图中，右键单击赋值节点，选中**补数据 > 当前节点及下游节点**。

iv. 刷新**补数据实例**页面，待补数据实例运行成功后，单击实例后的**DAG图**。

v. 右键单击do-while节点，选中**查看内部节点**。

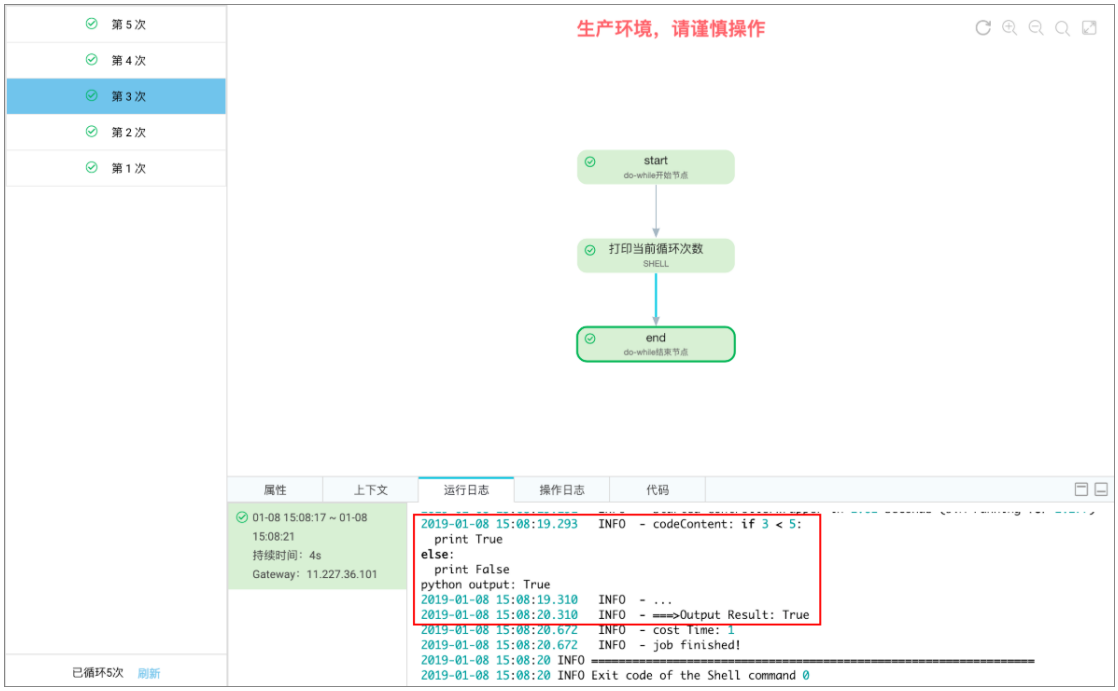


do-while节点的内部循环体分以下三部分：

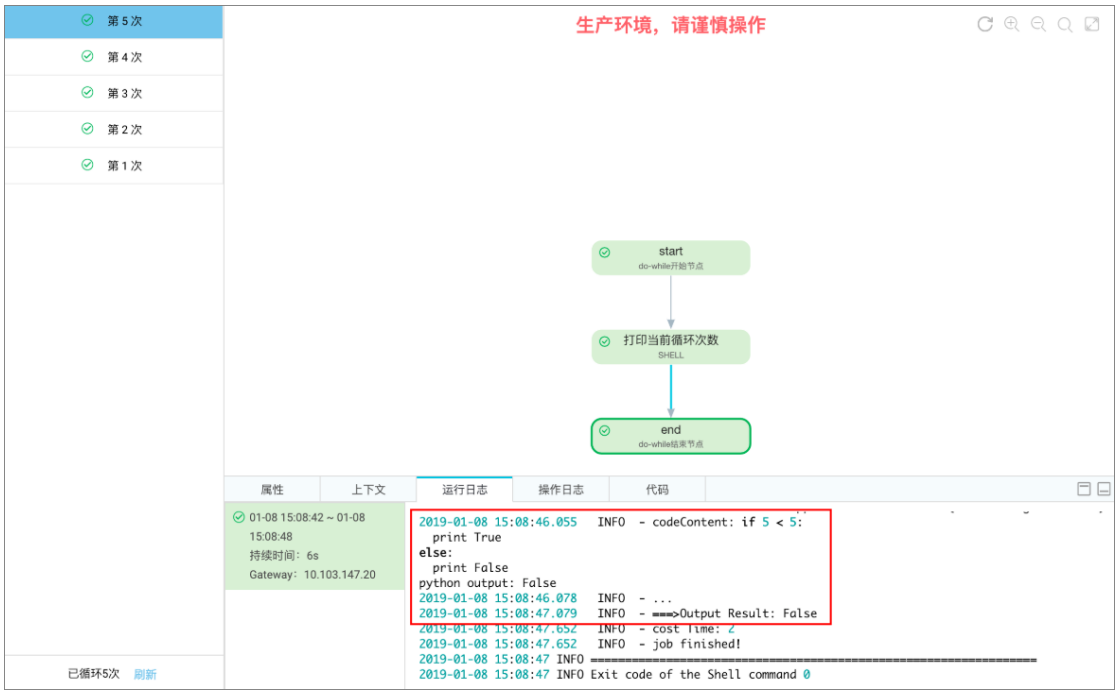
- 视图左侧为do-while节点的重跑历史列表，只要do-while实例整体运行一次，历史列表便会产生一条相应的记录。
- 视图中部为循环记录列表，会列出当前do-while节点共运行多少次循环，以及每次循环的状态。
- 视图右侧为每次循环的具体信息，单击循环记录列表中的某次循环，即可展示出该循环每个实例的运行情况。

vi. 在内部节点页面，单击左侧的**第3次**，并右键单击Shell节点，选中**查看运行日志**。

在运行日志页面，查看第3次循环end节点的日志。



查看第5次循环end节点的日志。



第5次循环运行结束，退出循环。

由该示例可见，do-while节点的工作流程如下：

i. 从start节点开始运行。
ii. 按照定义的任务依赖关系依次运行每个任务。
iii. 在end节点中定义循环的结束条件。
iv. 一组任务运行完毕之后，运行end的结束条件语句。
v. 如果end的判断语句在日志中打印True，则从1开始继续下一个循环。

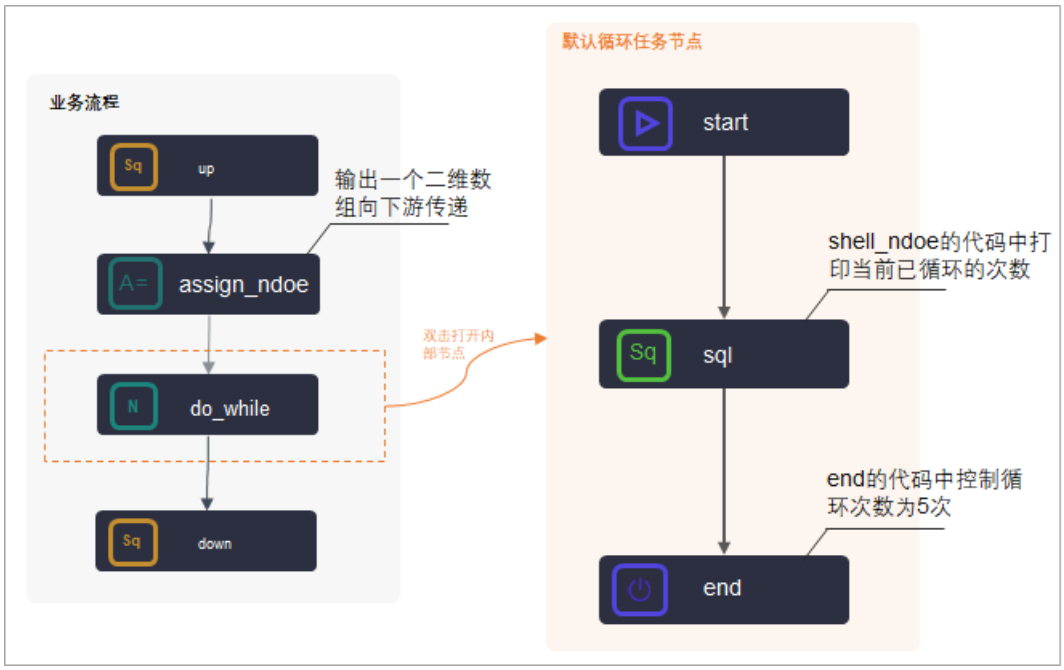
vi. 如果end的判断语句在日志中打印False，则退出整个循环，do-while节点整体结束。

循环节点的复杂示例

除上述简单场景外，您还会遇到通过循环的方式依次处理一组数据的每一行的复杂场景。实现该场景前，您需要满足以下条件：

- 需要部署一个上游节点，能够把查询出的数据输出给下游节点使用，您可以使用赋值节点实现该条件。
- 循环节点需要能够获取上游赋值节点的输出，您可以通过配置上下文依赖来实现该条件。
- 循环节点的内部节点需要能够引用到每一行的数据，增强已有的节点上下文，并额外下发了系统变量${dag.offset}，可以帮您快速引用循环节点的上下文。

以下以一个具体的案例，为您示例复杂场景的配置步骤。



如上图所示：

- 赋值节点输出一个二维数组，将此二维数组传递给do-while循环节点。

  二维数组的示例值为：

```
+-----------------------------------------------+
| uid            | region | age_range | zodiac |
+-----------------------------------------------+
| 0016359810821  | 湖北省 | 30~40岁   | 巨蟹座 |
| 0016359814159  | 未知   | 30~40岁   | 巨蟹座 |
+-----------------------------------------------+
```

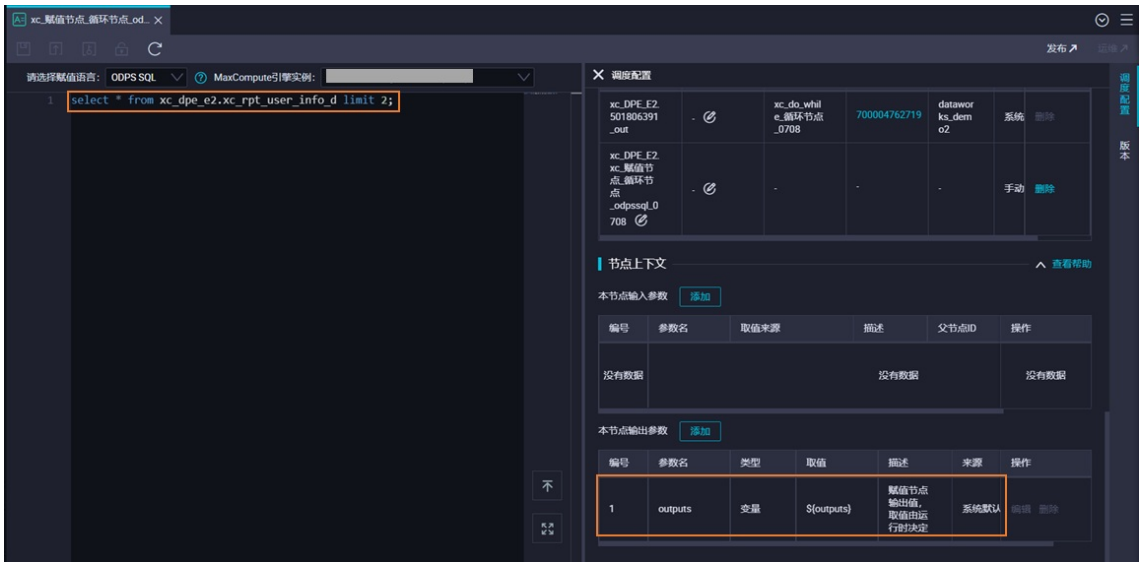
- do-while节点的内部节点通过变量来获取并打印当前循环参数、偏移量、上游赋值节点输入的参数值等。

1. 创建并配置赋值节点。

   核心操作要点为：

   - 赋值代码与上下文参数：选择赋值节点的赋值语言，并编译赋值参数的代码，后续赋值节点的本节点输出会根据规则生成输出参数。

> **说明** 赋值节点的输出后续需要作为do-while节点本节点输入。



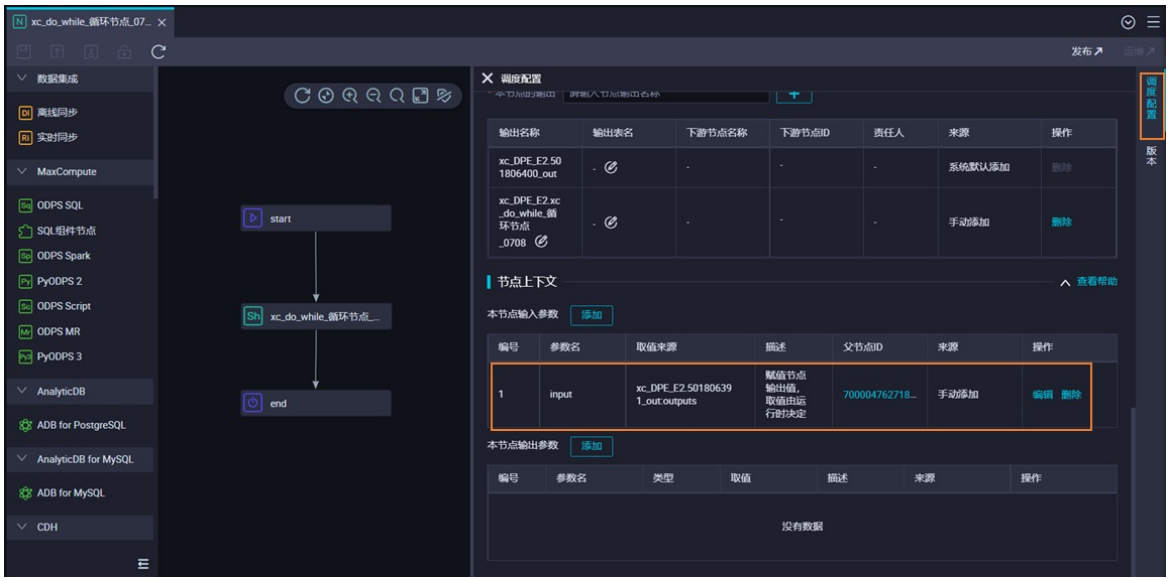
- 上下游依赖关系：您可以在业务流程中新建一个赋值节点，并通过连线，配置赋值节点为do-while节点的上游节点。

详细操作步骤请参见赋值节点。

2. 将赋值节点的输出添加为do-while节点的本节点输入。

   单击do-while节点编辑页面右侧的**调度配置**，在do-while节点的**节点上下文**区域，单击**添加**。设置**参数名**为input，**取值来源**为上游赋值节点的输出。

> **说明** 这里的上下文关系为赋值节点与do-while节点的上下文参数配置，不是内部节点的上下文参数配置。



3. 配置do-whiel节点的内部循环任务节点。

   双击do-while节点名称，打开节点的编辑页面，定义循环体。

   do-while节点默认有start、sql和end三个节点，您需要删除sql节点并创建一个Shell节点，通过编译

Shell节点的代码，打印循环参数。操作核心要点如下。

- 上下游依赖：删除sql新建Shell后，需要通过连线将内部节点的上下游关系建立好。

- 循环任务代码：内部的Shell节点的代码编译时，可以联合内置变量来打印各种循环参数。do-while节点可用的内置变量可参见内置变量，Shell节点的参考代码如下。

```
echo '${dag.input}';
echo '获取当前循环的行数据:'${dag.input[${dag.offset}]};
echo '获取偏移量:'${dag.offset};
echo '获取循环次数:'${dag.loopTimes};
echo '获取上游赋值节点_odpssql传递的数据集长度:'${dag.input.length};
echo '如果您要取赋值节点传递的数据集中某行某列数据,需要按照二维数组方式取值:'${dag.input[0][1]
};
```

4. 定义end节点的循环结束条件。

   可使用do-while节点支持的内置变量来进行循环控制。例如，比较变量dag.loopTimes（循环次数）和dag.input.length（取值长度）。如果dag.loopTimes小于dag.input.length，输出True并继续循环。如果不小于，则输出False并退出循环。样例代码如下。

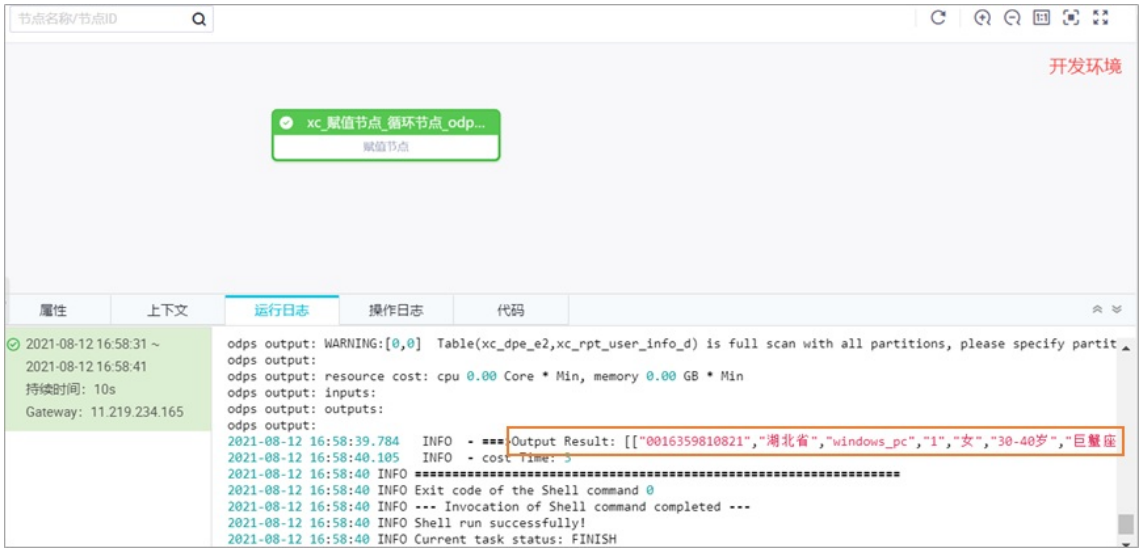
```
if ${dag.loopTimes}<${dag.input.length}:
    print True;
else:
    print False;
```

5. 运行节点并查看结果。

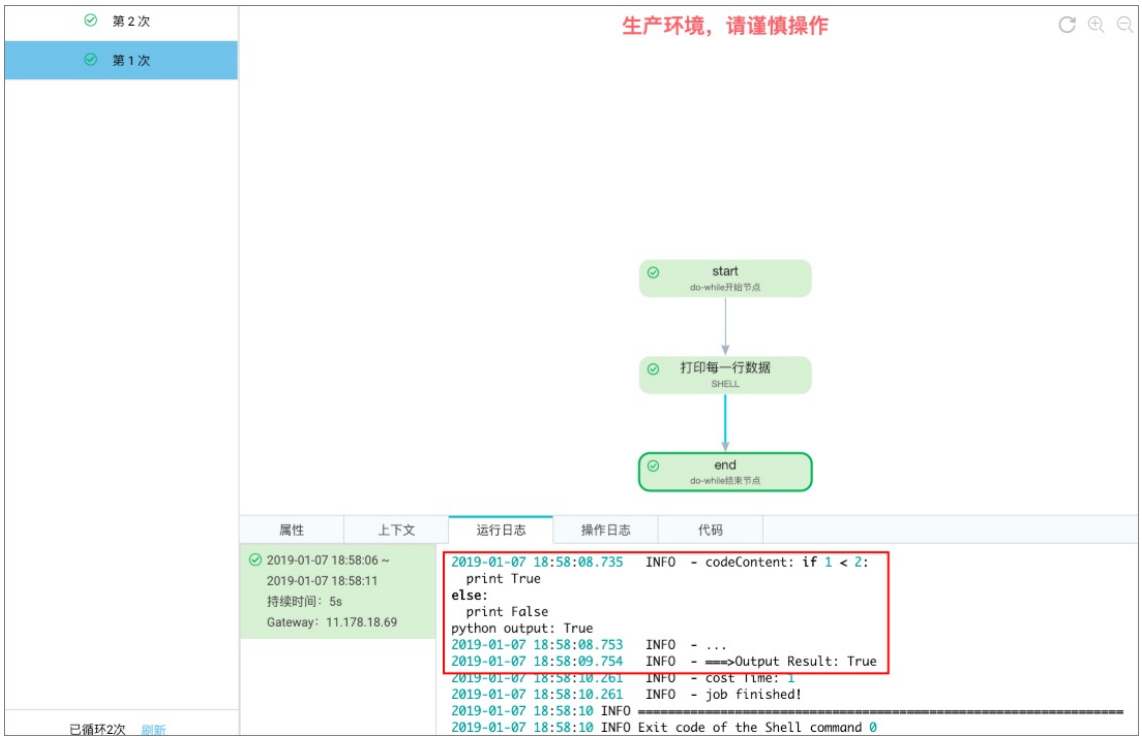
   进入运维中心后，右键节点选择**补数据 > 当前节点及下游节点**，选择赋值节点和循环节点，运行完成后在运行日志中查看运行结果。

   > **说明**
   > - 如果您在do-while节点内使用了赋值节点传递的值，请在运维中心测试时，同时运行赋值节点和循环节点。
   > - 在运维中心查看do-while节点的执行日志时，您需要右键实例，单击**查看内部节点**来查看内部节点的执行日志。

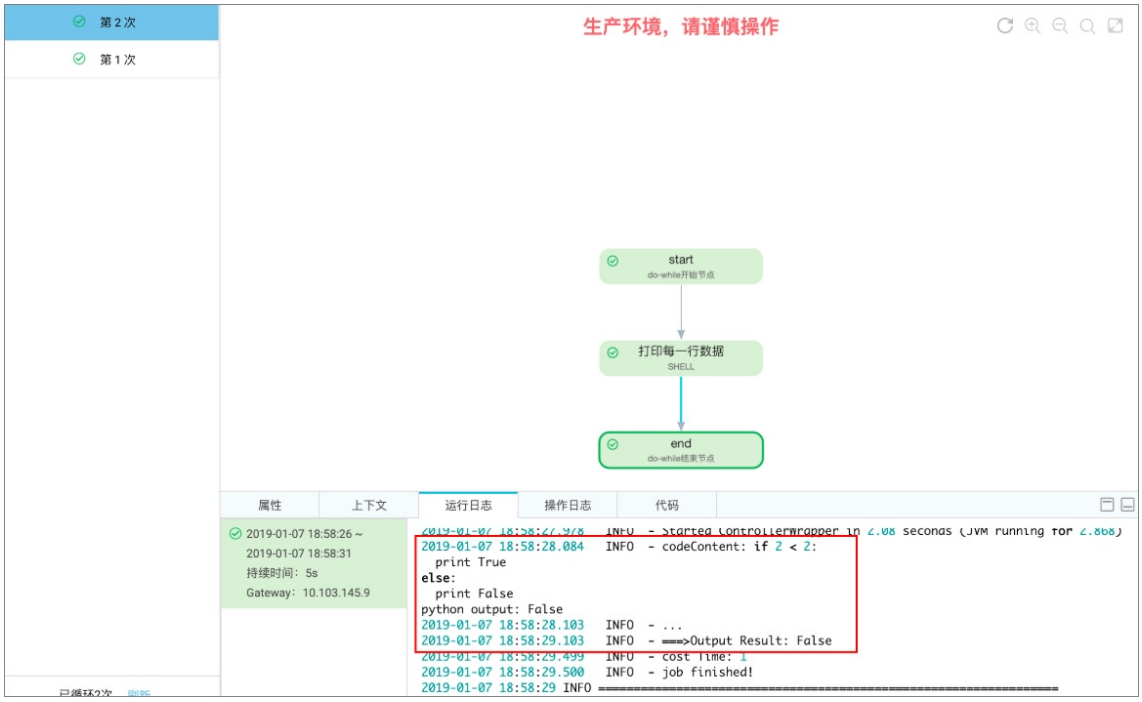
   - 赋值节点的输出结果。



- 第1次end节点的运行结果。



- 第2次end节点的运行结果。



总结

- do-while与while、for-each和do-while三种循环类型对比如下：
  - do-while能够实现先循环再判断的循环体，即do...while语句，能够通过系统的变量dag.offset结合节点上下文间接实现foreach语句。
  - do-while不能实现先判断再循环的方式，即while语句。
- do-while运行流程：
  i. 从start开始按任务依赖关系依次运行循环体中的任务。
  ii. 运行用户在end节点中定义的代码。
    - 如果end节点输出True，则继续下一个循环。
    - 如果end节点输出False，则终止循环。
- 如何使用上下文依赖：do-while的内部节点可以通过${dag.上下文变量名}的方式引用到do-while节点定义的节点上下文。
- 系统参数：DataWorks会为do-while内部节点自动下发两个系统变量。
  - dag.loopTimes：从1开始标识这一次循环的次数。
  - dag.offset：从0开始标识该次循环相对于第一次循环的次数偏移量。

6.11.4. 归并节点

本文为您介绍归并节点的概念，以及如何新建归并节点、定义归并逻辑，并通过实践案例为您展示归并节点的调度配置和运行详情。

背景信息

归并节点是DataStudio中提供的逻辑控制系列节点中的一类，可以对上游节点的运行状态进行归并，用于解决分支节点下游节点的依赖挂载和运行触发问题。

目前归并节点的逻辑定义不支持选择节点运行状态，仅支持将分支节点的多个下游节点归并为运行成功的状态，以便下游节点能够直接挂载归并节点作为依赖。

例如，分支节点C定义了两个逻辑互斥的分支走向C1和C2，不同分支使用不同的逻辑写入同一张MaxCompute表，如果下游节点B依赖此MaxCompute表的产出，则必须使用归并节点J先将分支归并后，再把归并节点J作为B的上游依赖。如果直接把B挂载在C1、C2下，任何时刻，C1和C2总有一个会因分支条件不满足，而显示实例状态为**分支未被选中**，而B也会因为上游有未被选中跳过运行的节点，实际也会是**分支未被选中**，空跑跳过的状态，节点并没有实际运行，所有下游节点均会如此。

使用限制

您需要购买DataWorks标准版及以上版本，才可以使用归并节点功能。

创建归并节点

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. 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. 鼠标悬停至**+新建**图标，单击**通用 > 归并节点**。
3. 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。

4. 单击**提交**。

定义归并逻辑

新建归并节点后，进入节点编辑页面定义归并逻辑。

1. 添加需要归并的分支节点，该节点作为归并节点的父节点。

   在**添加归并分支**后的下拉框，输入父节点的输出名称或输出的表名称，单击**+**图标。

   > **说明** 当需要归并多个分支节点时，您需要多次重复执行添加操作。

2. 在**归并条件设置**区域，配置分支节点的归并条件。

   您需要配置归并逻辑及分支节点的运行状态。

   - 归并逻辑条件包括：
     - **且**：上游所有分支节点运行完成后均满足其设置的运行状态时，**执行结果设置**区域设置的本节点运行状态才会生效。
     - **或**：上游任意分支节点运行完成后满足其设置的运行状态时，**执行结果设置**区域设置的本节点运行状态便会生效。
   - 节点运行完成后的状态包括：
     - **成功**：节点运行成功。
     - **失败**：节点运行失败。
     - **分支未运行**：节点未运行。

3. 在**执行结果设置**区域，设置本节点的运行状态。

   > **说明** 当前仅支持设置本节点的运行状态为**成功**。

如上图示例：

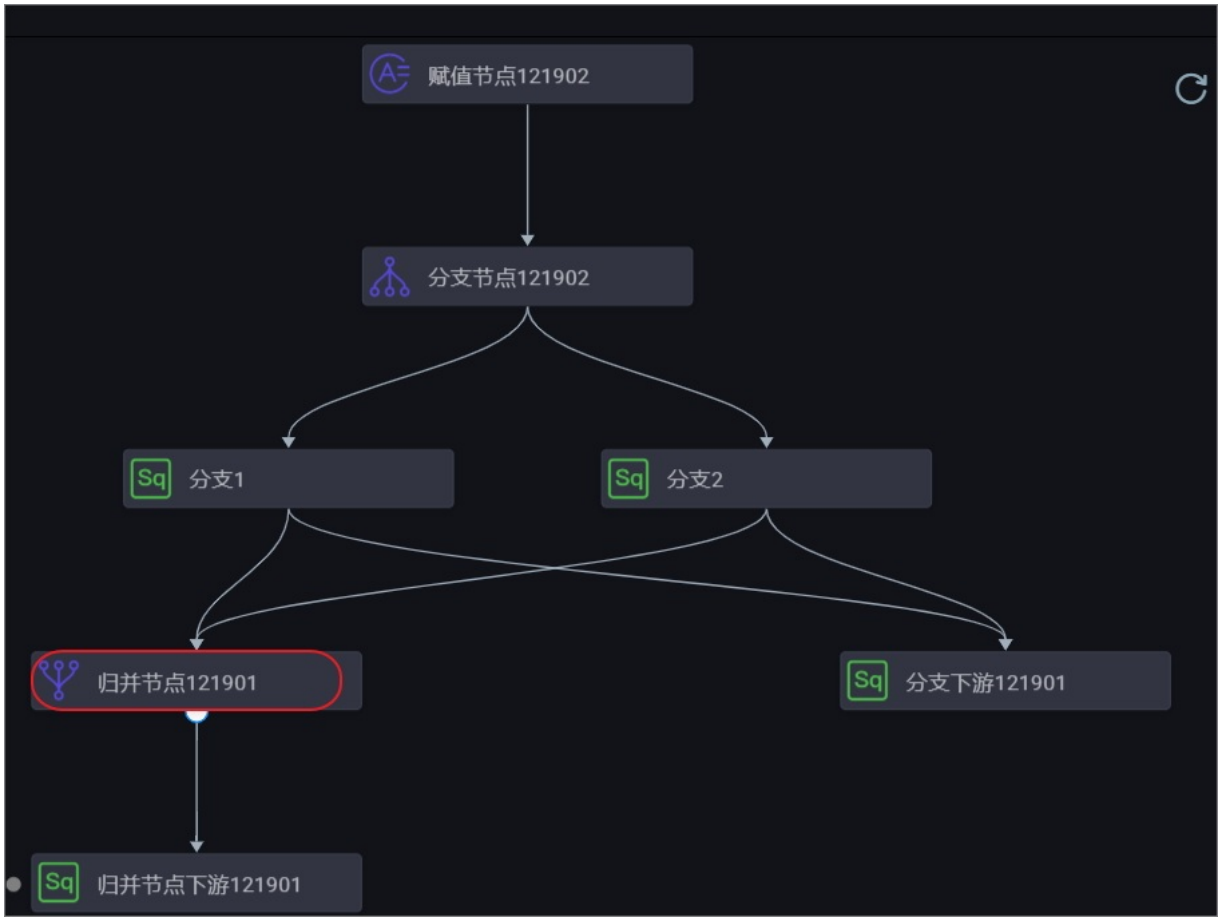
- 添加分支节点A、B为当前归并节点的上游节点。
- 节点A的运行状态设置为**成功**、**分支未运行**、**失败**，即节点A运行完成即可。
- 节点B的运行状态设置为**成功**、**分支未运行**，即节点B运行完成且节点B运行未失败。
- 归并逻辑条件设置为**且**。

因此，当节点A运行完成，并且节点B运行完成且不失败，当前归并节点的**成功**运行状态才会生效。

单击节点编辑页面右侧的**调度配置**，即可设置归并节点的调度属性。详情请参见配置基础属性。

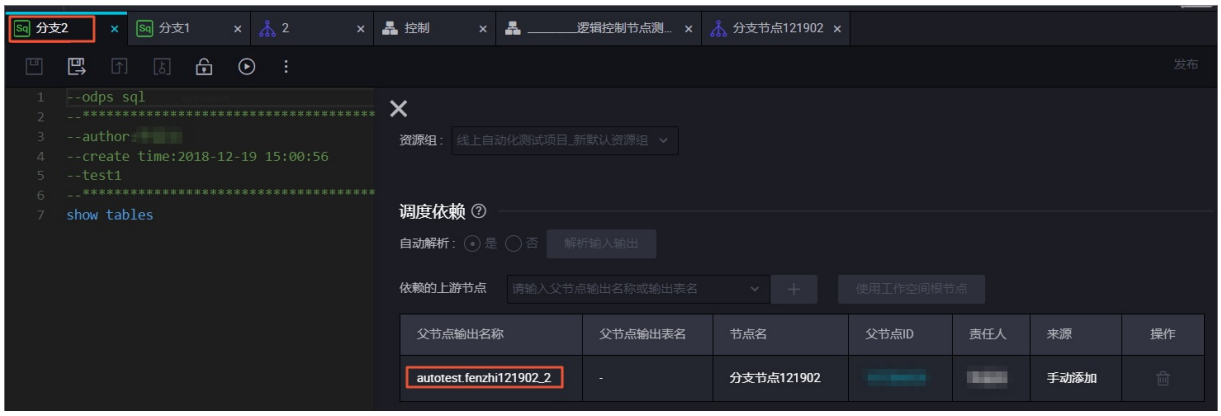
归并节点示例

在下游节点中，添加分支节点作为上游节点后，通过选择对应的分支节点输出来定义不同条件下的分支走向。例如在下图所示的业务流程中，**分支1**和**分支2**均为分支节点的两个下游节点。



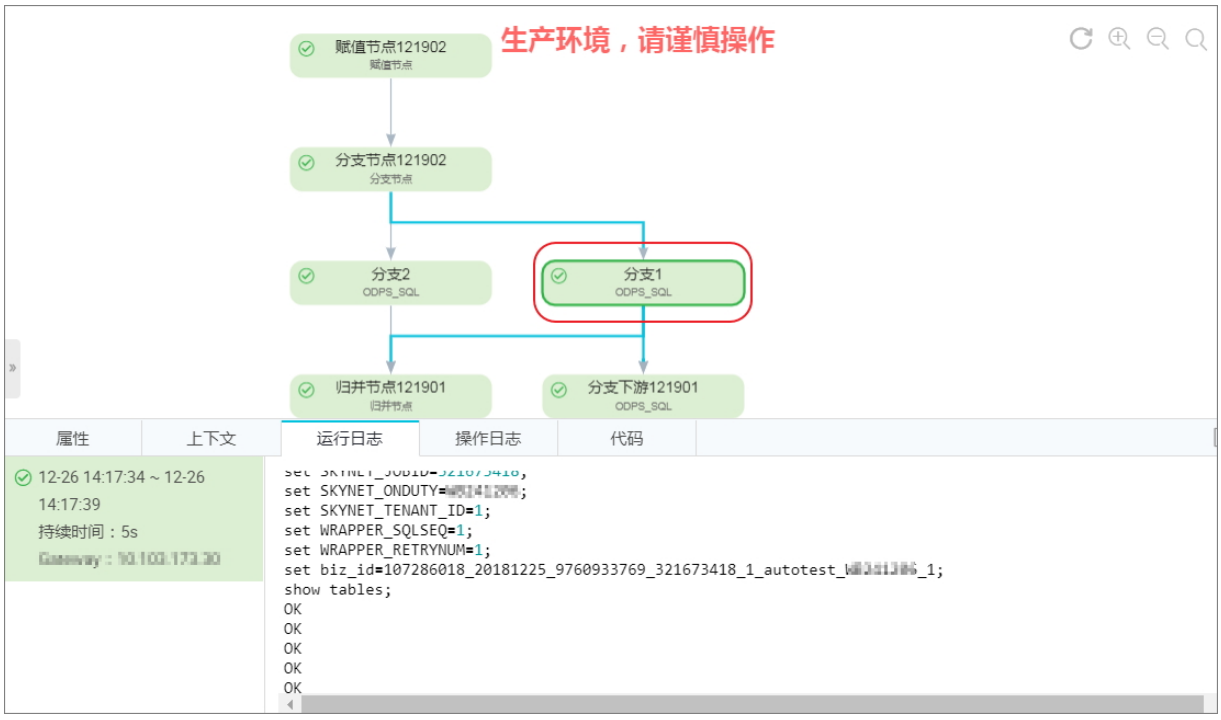
分支1依赖于autotest.fenzhi121902_1输出。

分支2依赖于autotest.fenzhi121902_2输出。

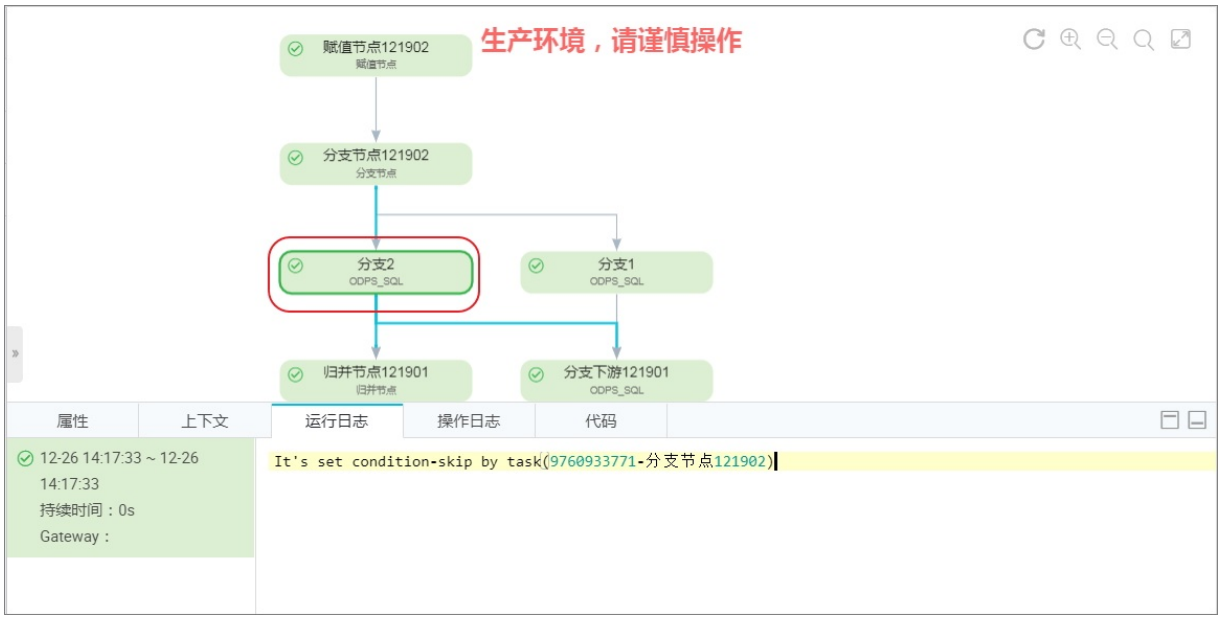


运行任务

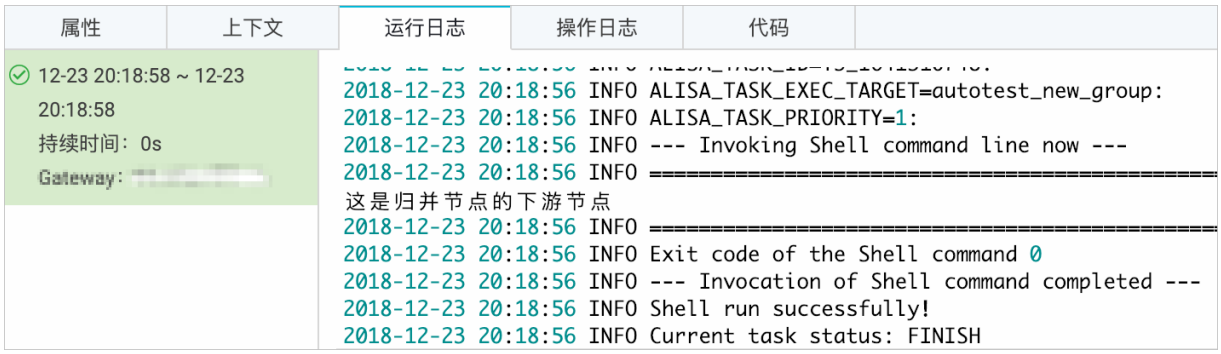
您可以在**运行日志**中查看满足分支条件、被选中运行的分支下游节点的运行情况。



您可以在**运行日志**中查看到不满足分支条件、未被选中运行的分支下游节点，被置为跳过。



归并节点的下游节点正常运行。



6.11.5. 分支节点

分支节点是DataStudio中提供的逻辑控制系列节点中的一类。分支节点可以定义分支逻辑和不同逻辑条件时下游分支走向。

前提条件

- 您需要购买DataWorks标准版及以上版本，才可以使用分支节点功能。
- 通常分支节点需要配合赋值节点使用，详情请参见赋值节点。

创建分支节点

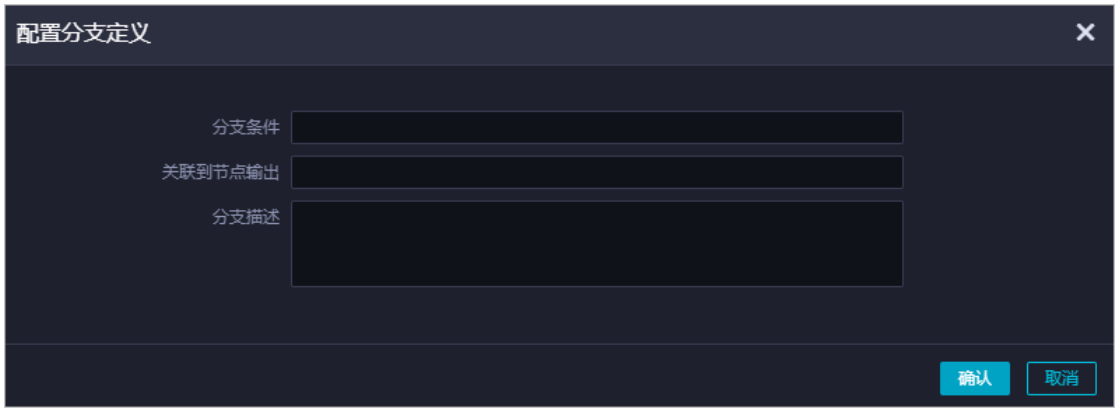
1. 进入**数据开发**页面。
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**工作空间列表**。
   iii. 选择工作空间所在地域后，单击相应工作空间后的**进入数据开发**。
2. 鼠标悬停至**+新建**图标，单击**通用 > 分支节点**。

   您也可以打开相应的业务流程，右键单击**通用**，选择**新建 > 分支节点**。

3. 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。

4. 单击**提交**。
5. 定义分支逻辑。
   i. 在**分支逻辑定义**页面，单击**添加分支**。
   ii. 在**配置分支定义**对话框中，配置各项参数。



参数	描述
分支条件	分支条件的说明如下： ■ 分支条件仅支持按照Python比较运算符定义逻辑判断条件。 ■ 如果运行态表达式取值为*true*，表示满足对应的分支条件。 ■ 如果运行态表达式解析报错，会将整个分支节点运行状态置为失败。 ■ 分支条件中支持使用全局变量和节点上下文定义的参数。例如，${input}可以是定义在分支节点的节点输入参数。
关联到节点输出	关联到节点输出的说明如下： ■ 节点输出供分支节点下游节点挂载依赖关系使用。 ■ 满足分支条件时，对应的关联的节点输出上挂载的下游节点被选中运行（同时需要参考该节点依赖的其它上游节点的状态）。 ■ 不满足分支条件时，对应的关联的节点输出上挂载的下游节点不会被选中执行，该下游节点会被置成 **因为分支条件不满足而未运行** 的状态。
分支描述	对分支的定义进行简要说明。例如，定义${input}==1和${input}>2两个分支。

关联到节点输出的示例如下。

例如，分支节点下游关联两个节点，节点名称分别为qqqq和wwww。当分支条件为1时，以qqqq节点逻辑运行。当分支条件为2时，以wwww节点逻辑运行。则配置分支节点时，关联到节点输出可以随便配置。例如，分支1关联节点输出为1234，分支2关联节点输出为2324，则均会作为本分支节点的输出名称。下游qqqq节点需要挂载分支节点的输出名称1234，wwww节点需要挂载分支节点的输出名称2324。

iii. 单击**确认**。

添加分支后，您可以进行**编辑**和**删除**操作：

- 单击**编辑**，可以修改设置的分支，并且相关的依赖关系也会改动。
- 单击**删除**，可以删除设置的分支，并且相关的依赖关系也会改动。

6. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性。

定义好分支条件后，会在**调度配置 > 调度依赖**区域的**本节点的输出**中，自动添加输出名称。下游节点可以通过输出名称进行依赖挂载。

> **说明**
> - 由于空跑属性会向下传递，不建议放置自依赖的任务在分支链路上。
> - 如果连线建立上下文的依赖，在调度配置中没有输出记录，请手动输入。

7. 提交节点。

> **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

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**。

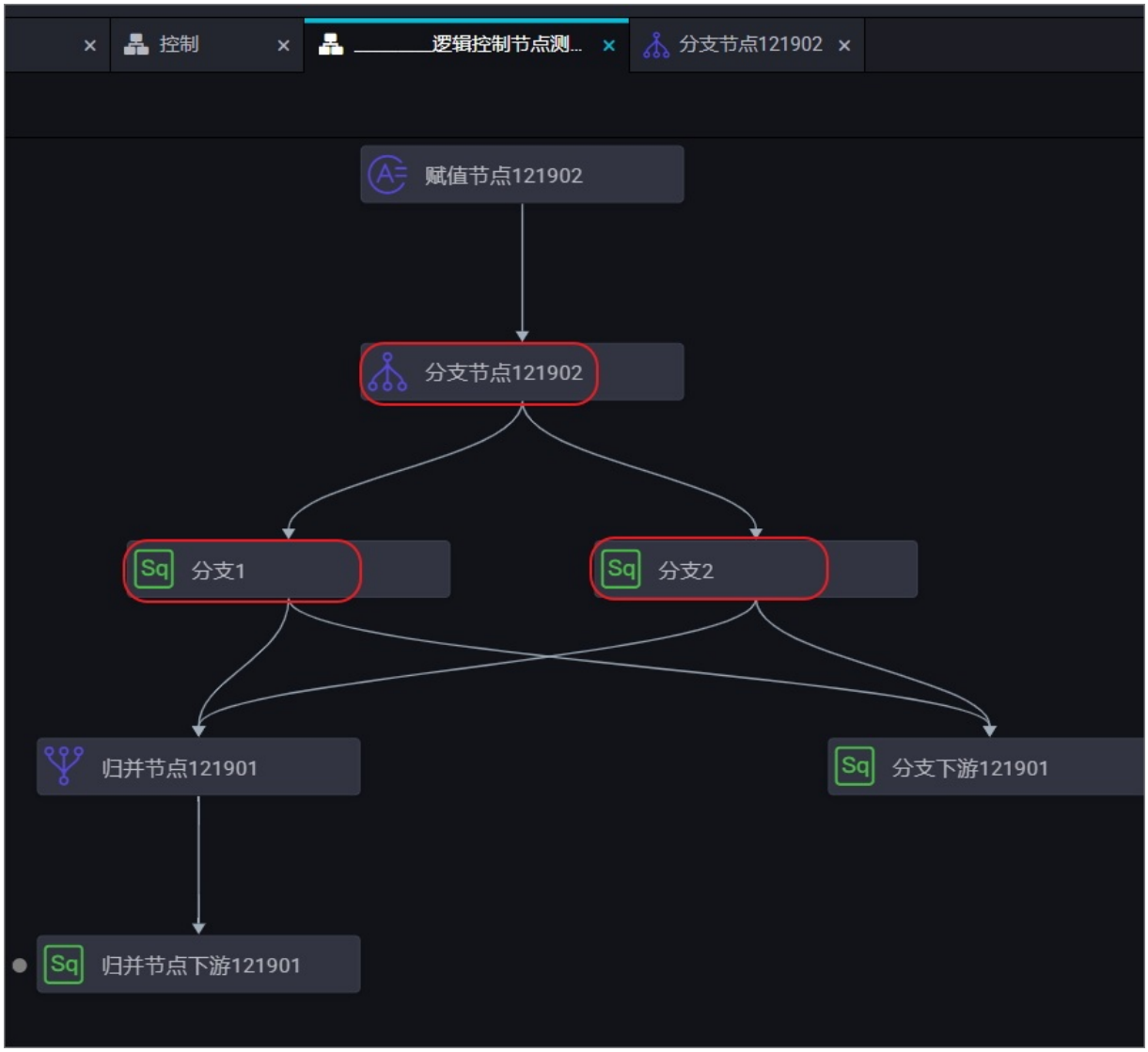
iii. 单击**确认**。

如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

8. 测试节点，详情请参见查看并管理周期任务。

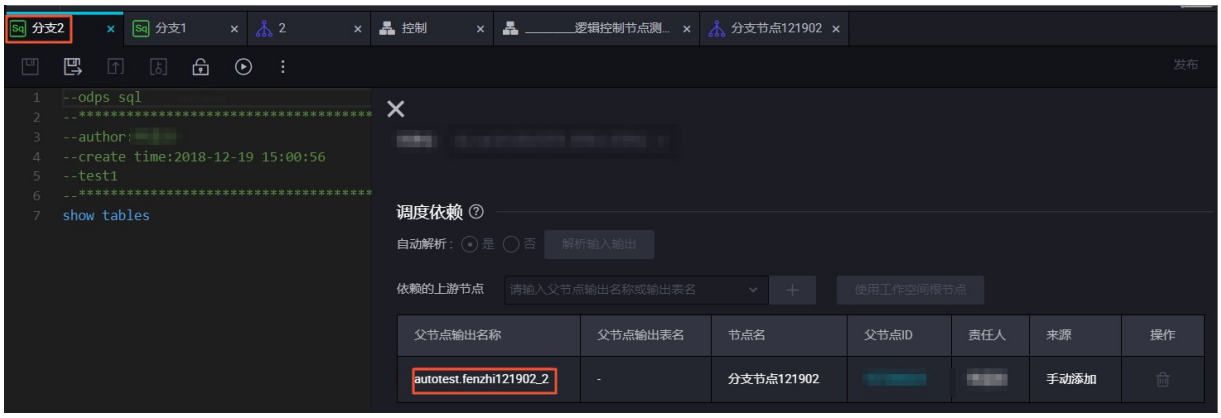
输出示例：下游节点挂载分支节点

在下游节点中，添加分支节点做为上游节点后，通过选择对应的分支节点输出来定义不同条件下的分支走向。例如在下图所示的业务流程中，**分支1**和**分支2**均为分支节点的两个下游节点。



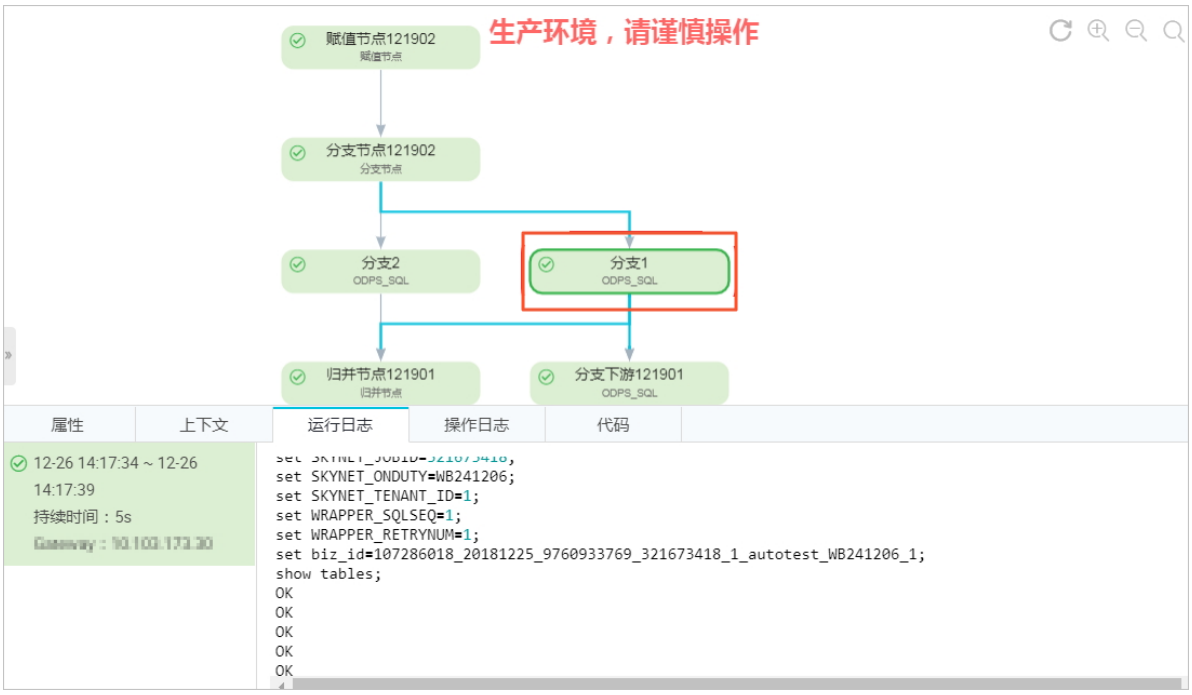
分支1：依赖于autotest.fenzhi121902_1输出。

分支2：依赖于autotest.fenzhi121902_2输出。

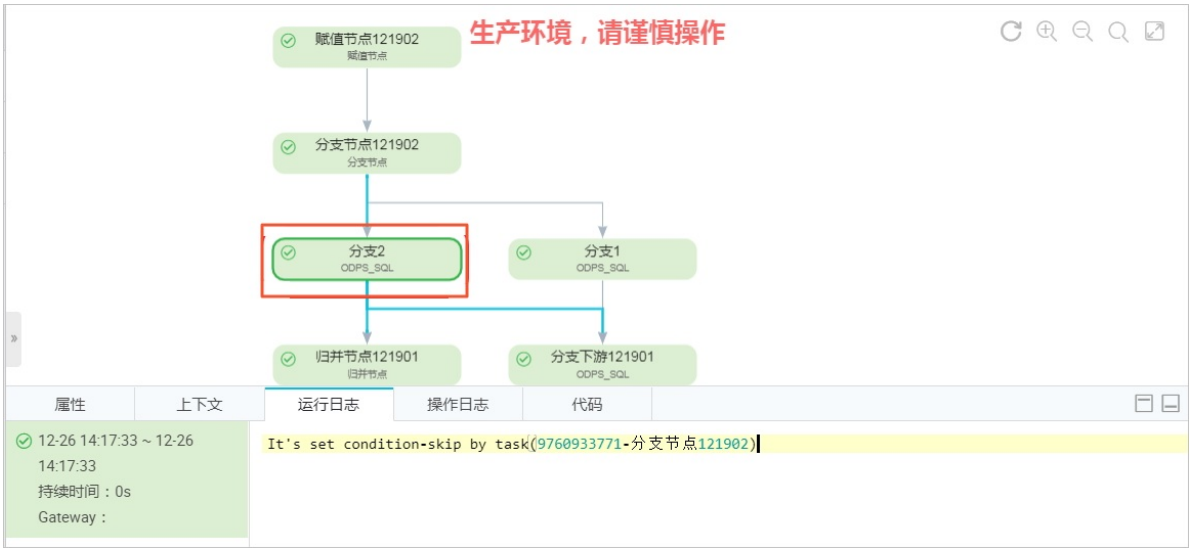


提交调度至运维中心运行，分支节点满足条件一（依赖于autotest.fenzhi121902_1），则日志的打印结果如下：

- 您可以在**运行日志**中查看满足分支条件、被选中运行的分支下游节点的运行情况。



- 您可以在**运行日志**中查看到不满足分支条件、未被选中运行的分支下游节点，被置为跳过。



支持的Python比较运算符

以下假设变量a为10，变量b为20。

运算符	描述	示例
==	等于：比较对象是否相等。	（a==b）返回false。
!=	不等于：比较两个对象是否不相等。	（a!=b）返回true。
<>	不等于：比较两个对象是否不相等。	（a<>b）返回true。这个运算符类似!=。
>	大于：返回x是否大于y。	（a>b）返回false。
<	小于：返回x是否小于y。所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。	（a<b）返回true。
>=	大于等于：返回x是否大于等于y。	（a>=b）返回false。
<=	小于等于：返回x是否小于等于y。	（a<=b）返回true。

6.11.6. 赋值节点

当您需要将上游节点任务的结果提供给下游节点使用时，您可使用赋值节点，实现任务结果在节点间传递。赋值节点支持ODPS SQL、SHELL和Python2三种赋值语言，且根据赋值规则，自动为您添加赋值参数（outputs参数），便于其他节点引用。您可以结合节点上下文配置，参考本文使用赋值节点。

前提条件

您已购买标准版及以上版本的DataWorks。

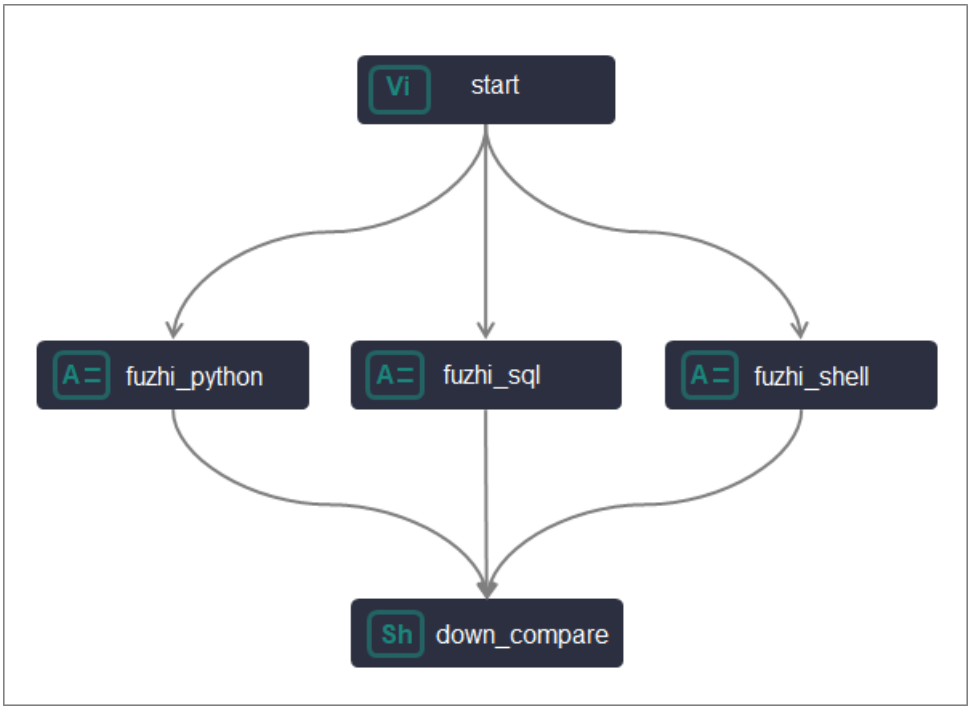
标准版及以上版本的DataWorks才支持使用赋值节点。

背景信息

使用赋值节点进行透传参数时，需关注以下三个要点：

- 业务流程中，赋值节点与上下游节点间的依赖关系

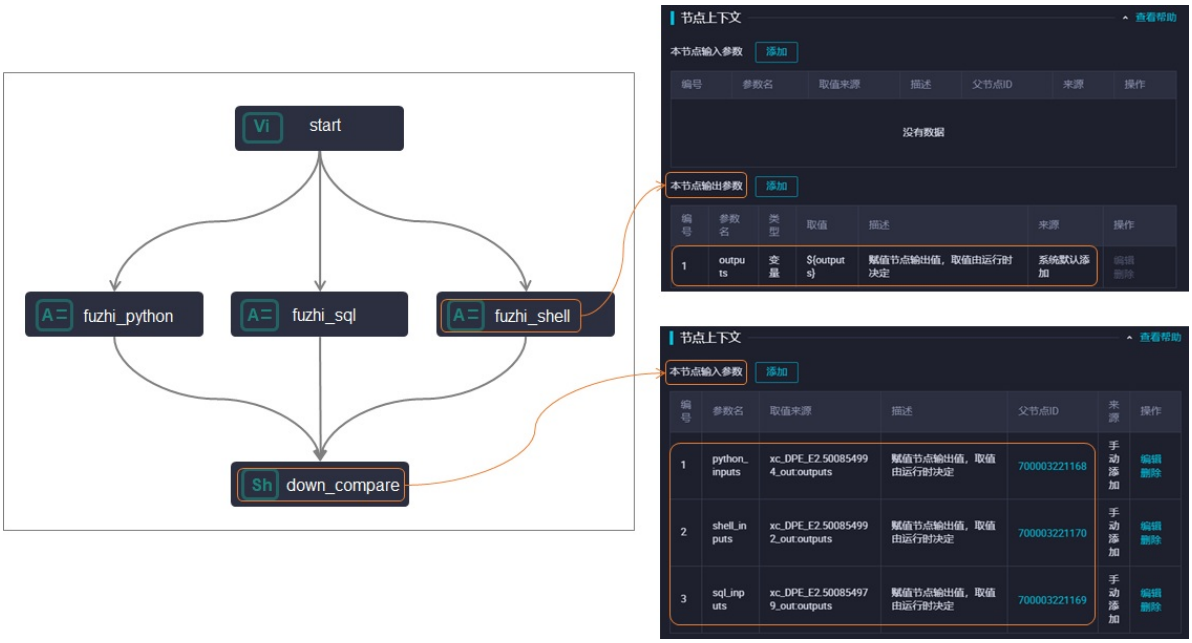
- 业务流程中，赋值节点与上下游节点间的依赖关系。



如上图所示，使用赋值节点透传参数时：

- 赋值节点（fuzhi_python、fuzhi_sql、fuzhi_shell）需作为引用赋值节点参数节点（down_compare）的上游节点，下游节点需要与赋值节点设置直接依赖关系（赋值节点为下游节点的一层父节点）。
- 赋值节点作为上游与其它节点配合使用时，请务必先提交赋值节点，以便下游节点在配置时可以解析出参数。

- 参数传递时，赋值节点与下游节点的上下文参数透传关系。



如上图所示，通过赋值节点与引用节点的上下文参数的配置，形成参数透传引用关系：

- 赋值节点（fuzhi_python、fuzhi_sql、fuzhi_shell）需将待赋值给下游的参数添加为**节点上下文**中的**本节点输出参数**。
- 下游引用赋值参数的节点需将待引用的赋值参数添加为**节点上下文**中的**本节点输入参数**。

> 说明
> - 部分数据开发节点，可直接在节点的上下游参数中手动添加赋值参数（outputs参数），无需通过赋值节点即可将参数透传给下游节点引用。例如，EMR Hive、EMR Spark SQL、ODPS Script、Hologres SQL、AnalyticDB for PostgreSQL和MySql节点，此类节点支持手动添加赋值节点，赋值参数的使用与赋值节点一致，添加赋值参数的操作详情请参见配置节点上下文。
> - 其他节点无法直接在本节点中直接添加赋值参数，需要使用赋值节点进行参数透传。
> - 赋值节点参数传递只支持传递给一层子节点，不支持跨节点传递。

- 如果下游需要取赋值节点传递结果，下游节点连同赋值节点一块执行，您可以业务流程面板运行或者在运维中心执行验证上下游参数传递情况。
- 参数引用时，赋值节点的参数输出格式与下游节点引用参数方式的关系。

  不同语言的赋值参数（outputs参数）赋值说明如下。

赋值语言	outputs参数取值	outputs参数格式	outputs参数大小限制
ODPS SQL	最后一行SELECT语句的输出作为赋值参数，添加为赋值节点的本节点输出参数，供其他节点引用。	将输出结果作为一个二维数组传递至下游。	传递值最大为2 MB。如果赋值语句的输出结果超过该限制，赋值节点会运行失败。
SHELL	最后一行ECHO语句的数据，添加为赋值节点的本节点输出参数，供其他节点引用。	将输出结果基于逗号（,）分割为一维数组。	
Python2	最后一行PRINT语句的输出，添加为赋值节点的本节点输出参数，供其他节点引用。	将输出结果基于逗号（,）分割为一维数组。	

使用限制

- 标准版及以上版本的DataWorks才支持使用赋值节点。
- 赋值节点使用的Python2语言为普通的Python语法，不支持直接访问MaxCompute数据。若要访问MaxCompute数据，需在独享调度资源组引入MaxCompute的Python资源包，并在节点中按照外部应用访问MaxCompute的方式配置AccessKey、Endpoint、ProjectName等信息来访问MaxCompute，详情请参见在PyODPS节点中调用第三方包。Python资源的参数引用方式请参见PyODPS节点调度参数配置示例。

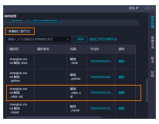
操作流程

本文以在down_compare节点中，分别输出赋值节点使用Python2、ODPS SQL和SHELL语言编辑的最后一行代码输出结果为例，为您介绍赋值节点如何结合节点上下文实现上下游参数传递，操作流程如下。

1. 创建赋值节点及其他节点。
2. 配置上下游依赖。
3. 配置上下文参数并引用赋值参数（ODPS SQL）。

4. 配置上下文参数并引用赋值参数（Python）。
5. 配置上下文参数并引用赋值参数（SHELL）。

不同语言的赋值参数（outputs参数）使用案例如下。

赋值语言	outputs取值示例	赋值节点调度配置	下游节点调度配置	下游节点取值方式	下游节点返回结果
ODPS SQL	示例查看*fuzhi_tb*表。 • 查询代码：`SELECT * FROM fuzhi_tb;`。 • 显示结果。 	1. 赋值节点的**调度配置 > 节点上下文**默认生成一个本节点输出参数**ouputs**。  2. 在节点编辑页面，单击  图标提交节点。 配置节点上下文，详情请参见配置节点上下文。	以上游赋值节点使用的赋值语言为**ODPS SQL**示例。 1. 配置下游节点依赖上游赋值节点。  配置节点调度依赖，详情请参见配置同周期调度依赖。 2. 节点上下文添加本节点输入参数，参数命名为`inputs_odps_sql`。  配置节点上下文，详情请参见配置节点上下文。	不同类型的下游节点取值如下： • ODPS SQL：`select '${inputs_odps_sql[0][0]}';`。 • SHELL：`echo '${inputs_shell[0]}';`。 • Pyodps3：`print ('${inputs_python[0]}');`。	Hello
SHELL	示例语句为：`echo 'Data','我是赋值节点2赋值语言shell';`。				Data
Python2	示例语句为：`print "Works!,我是赋值节点3赋值语言是python";`。				Works!

创建赋值节点及其他节点

根据本文的示例场景，需使用3个赋值节点，分别示例3种赋值语言场景的使用，因此您首先需要创建3个赋值节点。

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**进入数据开发**。
4. 鼠标悬停至+新建图标，单击**通用 > 赋值节点**。

   您也可以找到相应的业务流程，右键单击**通用**，选择**新建 > 赋值节点**。
5. 在**新建节点**对话框中，输入**节点名称**，并选择**目标文件夹**。

   本示例中新建3个不同语言的赋值节点（Python2、ODPS SQL和SHELL），节点名称分别为fuzhi_python、fuzhi_sql、fuzhi_shell。

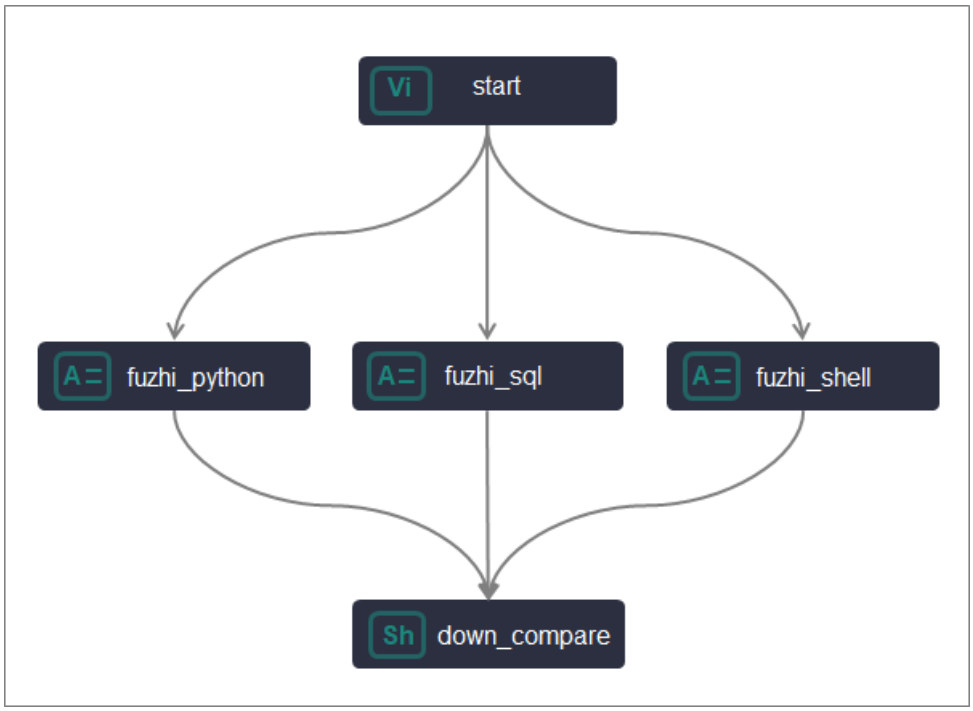
> 注意　节点名称必须是大小写字母、中文、数字、下划线（_）以及小数点（.），且不能超过128个字符。

6. 单击**提交**。

重复上述步骤，创建完成所有的赋值节点，并根据实际情况，创建完成业务流程里的其他节点，例如本示例中，您还需创建上游start节点（虚拟节点）、下游down_compare节点（Shell节点），操作详情可参见虚拟节点和Shell节点。

配置上下游依赖

创建完成赋值节点（Python、ODPS SQL和SHELL）和其他节点后，您需要根据实际的业务关系，设置节点的上下游依赖关系。



本示例中，您可以直接通过拉线，将start节点作为所有赋值节点的上游节点，down_compare节点作为所有赋值节点的下游节点，操作详情可参见调度依赖配置指导：鼠标拖拽。

此外，您可根据实际需要配置各节点调度配置中的基础属性、时间属性、资源属性，详情可参见配置基础属性、时间属性配置说明、配置资源属性。

配置上下文参数并引用赋值参数（ODPS SQL）

以下以配置赋值语言为ODPS_SQL的赋值节点，并在down_compare节点中引用赋值参数为例，为您示意如何操作。

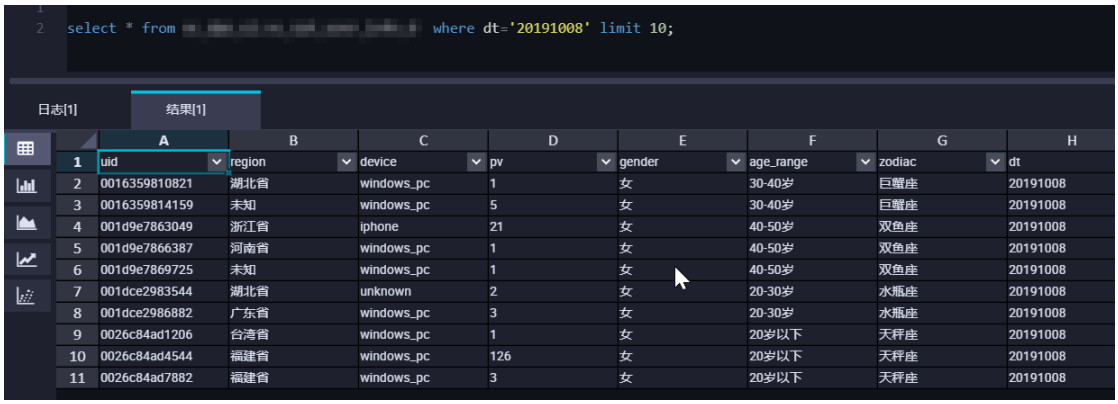
1. 配置赋值节点。
   i. 在相应的业务流程下，双击打开赋值语言为ODPS SQL的赋值节点fuzhi_sql。
   ii. 在代码编辑页面，选择赋值语言为ODPS_SQL，写入赋值代码。
   例如：

```
select * from xc_dpe_e2.xc_rpt_user_info_d  where dt='20191008' limit 10;
```

iii. 单击页面右侧的**调度配置**，查看**节点上下文**中的**本节点输出参数**。

赋值节点将代码的查询结果作为节点输出，赋值给赋值节点自带的输出参数outputs。

本示例赋值节点的查询结果如下。

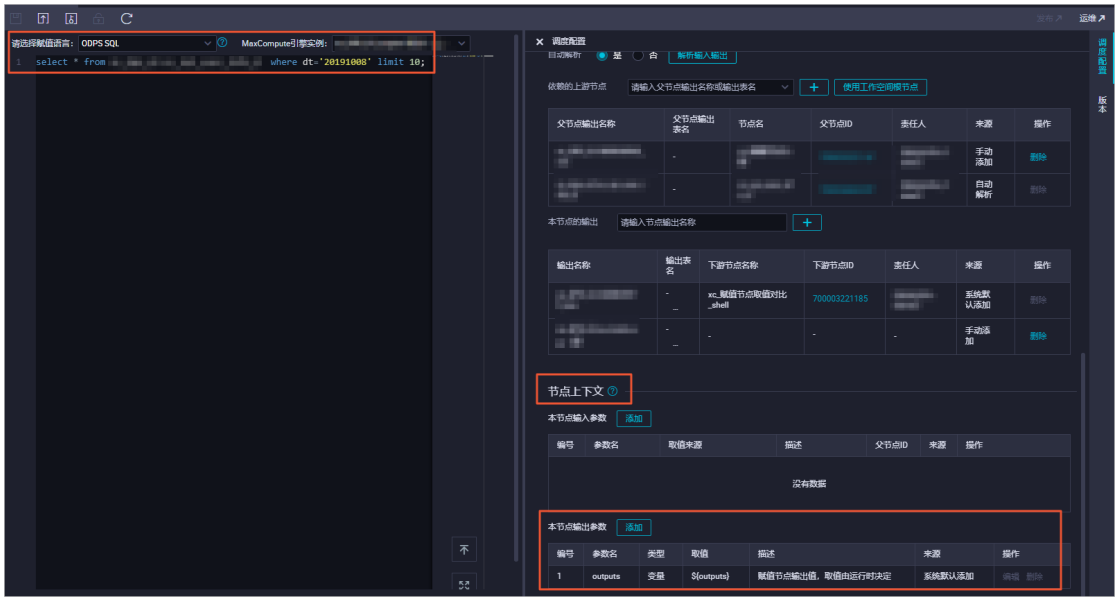


select * from [redacted] where dt='20191008' limit 10;

日志[1] 结果[1]

	A	B	C	D	E	F	G	H
1	uid	region	device	pv	gender	age_range	zodiac	dt
2	0016359810821	湖北省	windows_pc	1	女	30-40岁	巨蟹座	20191008
3	0016359814159	未知	windows_pc	5	女	30-40岁	巨蟹座	20191008
4	001d9e7863049	浙江省	iphone	21	女	40-50岁	双鱼座	20191008
5	001d9e7866387	河南省	windows_pc	1	女	40-50岁	双鱼座	20191008
6	001d9e7869725	未知	windows_pc	1	女	40-50岁	双鱼座	20191008
7	001dce2983544	湖北省	unknown	2	女	20-30岁	水瓶座	20191008
8	001dce2986882	广东省	windows_pc	3	女	20-30岁	水瓶座	20191008
9	0026c84ad1206	台湾省	windows_pc	1	女	20岁以下	天秤座	20191008
10	0026c84ad4544	福建省	windows_pc	126	女	20岁以下	天秤座	20191008
11	0026c84ad7882	福建省	windows_pc	3	女	20岁以下	天秤座	20191008

则此查询结果作为一个二维数组，赋值给**本节点输出参数**中的outputs参数。



2. 配置引用节点。

i. 双击打开下游Shell节点down_compare节点。

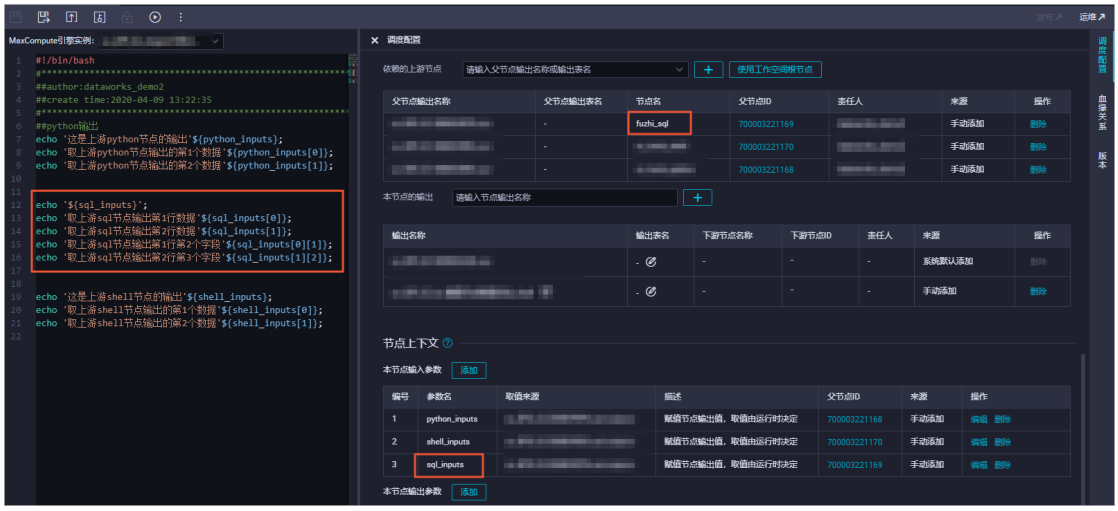
ii. 在代码开发页面编写代码。

例如：

```
echo '${sql_inputs}';
echo '取上游sql节点输出第1行数据'${sql_inputs[0]};
echo '取上游sql节点输出第2行数据'${sql_inputs[1]};
echo '取上游sql节点输出第1行第2个字段'${sql_inputs[0][1]};
echo '取上游sql节点输出第2行第3个字段'${sql_inputs[1][2]};
```

iii. 单击页面右侧的**调度配置**，配置**节点上下文**中的**本节点输入参数**。

将fuzhi_sql节点的outputs参数添加为本节点输入参数，并命名为sql_inputs。



3. 执行引用，查看引用结果。

i. 单击工具栏中的图标。

ii. 在**警告**对话框中，单击**继续运行**。

iii. 查看引用结果。

```
0016359810821,湖北省,windows_pc,1,女,30-40岁,巨蟹座,20191008
0016359814159,未知,windows_pc,5,女,30-40岁,巨蟹座,20191008
001d9e7863049,浙江省,iphone,21,女,40-50岁,双鱼座,20191008
001d9e7866387,河南省,windows_pc,1,女,40-50岁,双鱼座,20191008
001d9e7869725,未知,windows_pc,1,女,40-50岁,双鱼座,20191008
001dce2983544,湖北省,unknown,2,女,20-30岁,水瓶座,20191008
001dce2986882,广东省,windows_pc,3,女,20-30岁,水瓶座,20191008
0026c84ad1206,台湾省,windows_pc,1,女,20岁以下,天秤座,20191008
0026c84ad4544,福建省,windows_pc,126,女,20岁以下,天秤座,20191008
0026c84ad7882,福建省,windows_pc,3,女,20岁以下,天秤座,20191008
取上游sql节点输出第1行数据 0016359810821,湖北省,windows_pc,1,女,30-40岁,巨蟹座,20191008
取上游sql节点输出第2行数据 0016359814159,未知,windows_pc,5,女,30-40岁,巨蟹座,20191008
取上游sql节点输出第1行第2个字段 湖北省
取上游sql节点输出第2行第3个字段windows_pc
```

配置上下文参数并引用赋值参数（Python）

以下以配置赋值语言为Python2的赋值节点，并在down_compare节点中引用赋值参数为例，为您示意如何操作。

1. 配置赋值节点。

i. 在相应的业务流程下，双击打开赋值语言为Python2的上游节点fuzhi_python。

ii. 在代码编辑页面，选择赋值语言为Python2，写入赋值代码。

例如：

```
print "a,b,c";
```

> **说明** 赋值节点使用的Python2语言为普通的Python语法，不支持直接访问MaxCompute数据。若要访问MaxCompute数据，需在独享调度资源组引入MaxCompute的Python资源包，并在节点中按照外部应用访问MaxCompute的方式配置AccessKey、Endpoint、ProjectName等信息来访问MaxCompute，详情请参见在PyODPS节点中调用第三方包。Python资源的参数引用方式请参见PyODPS节点调度参数配置示例。

iii. 单击页面右侧的**调度配置**，查看**节点上下文**中的**本节点输出参数**。

赋值节点将代码的查询结果作为节点输出，赋值给赋值节点自带的输出参数outputs。本示例的查询结果为a,b,c。

赋值语言为Python2时，查询结果将基于逗号（,）分割为一维数组，赋值给**本节点输出参数**中的**outputs**参数。

2. 配置引用节点。

i. 双击打开下游Shell节点down_compare节点。

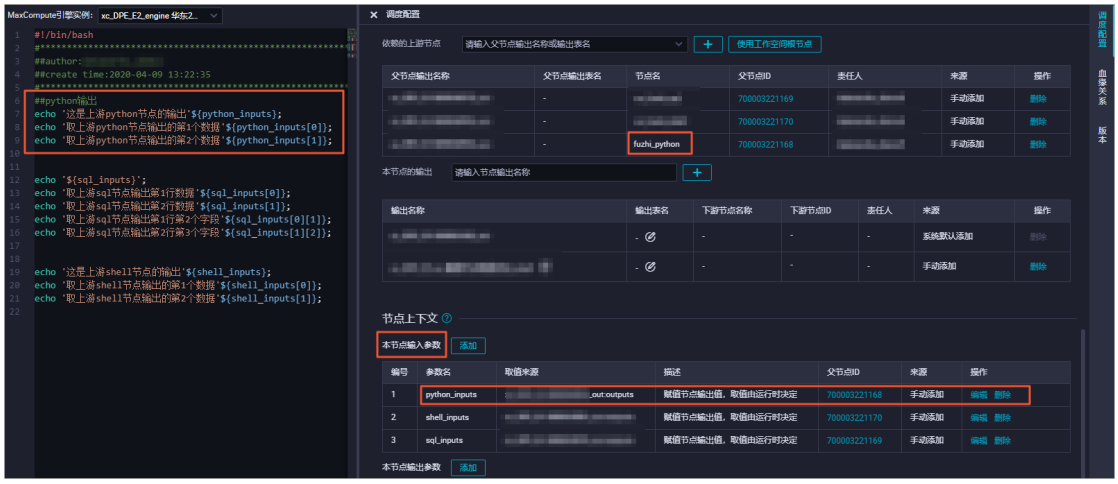
ii. 在代码开发页面编写代码。

例如：

```
echo '这是上游python节点的输出'${python_inputs};
echo '取上游python节点输出的第1个数据'${python_inputs[0]};
echo '取上游python节点输出的第2个数据'${python_inputs[1]};
```

iii. 单击页面右侧的**调度配置**，配置**节点上下文**中的**本节点输入参数**。

将fuzhi_python节点的outputs参数添加为本节点输入参数，并命名为python_inputs。

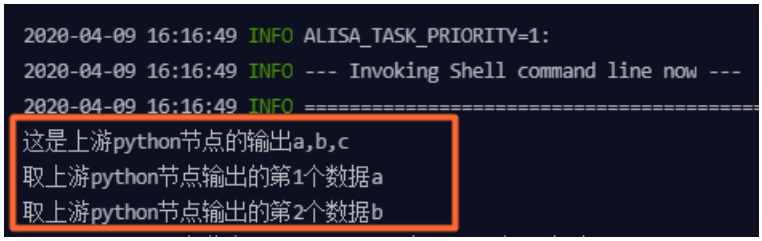


3. 执行引用，查看引用结果。

i. 单击工具栏中的图标。

ii. 在**警告**对话框中，单击**继续运行**。

iii. 查看引用结果。



配置上下文参数并引用赋值参数（SHELL）

以下以配置赋值语言为SHELL的赋值节点，并在down_compare节点中引用赋值参数为例，为您示意如何操作。

1. 配置赋值节点。

i. 在相应的业务流程下，双击打开赋值语言为SHELL的上游节点fuzhi_shell。

ii. 在代码编辑页面，选择赋值语言为SHELL，写入赋值代码。

例如：

```
echo "hello,world";
```

iii. 单击页面右侧的**调度配置**，查看**节点上下文**中的**本节点输出参数**。

赋值节点将代码的查询结果作为节点输出，赋值给赋值节点自带的输出参数outputs。本示例的查询结果为hello,world。

赋值语言为SHELL时，查询结果将基于逗号（,）分割为一维数组，赋值给**本节点输出参数**中的**outputs**参数。

2. 配置引用节点。

i. 双击打开下游Shell节点down_compare节点。

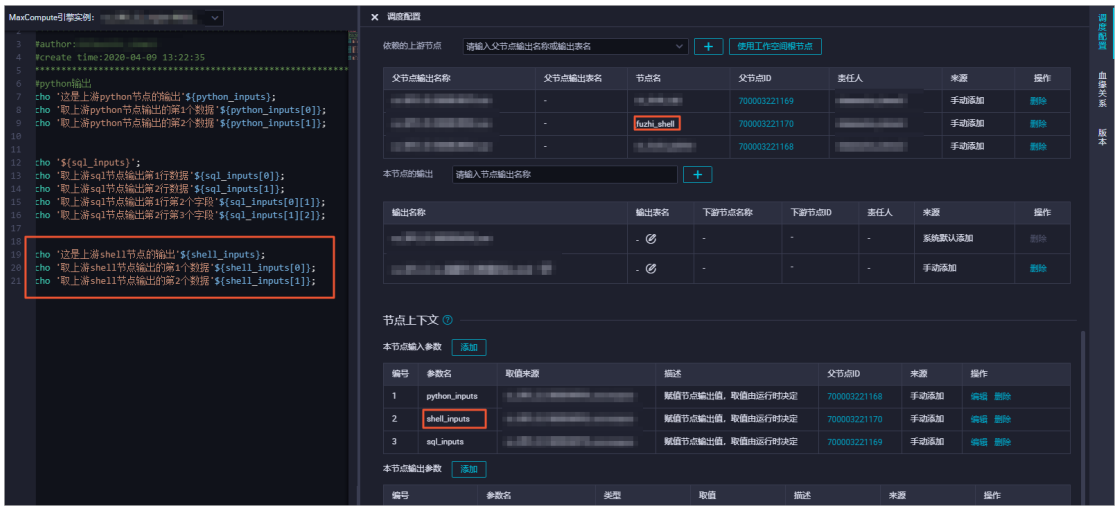
ii. 在代码开发页面编写代码。

例如：

```
echo '这是上游shell节点的输出'${shell_inputs};
echo '取上游shell节点输出的第1个数据'${shell_inputs[0]};
echo '取上游shell节点输出的第2个数据'${shell_inputs[1]};
```

iii. 单击页面右侧的**调度配置**，配置**节点上下文**中的**本节点输入参数**。

将fuzhi_shell节点的outputs参数添加为本节点输入参数，并命名为shell_inputs。



3. 执行引用，查看引用结果。

i. 单击工具栏中的图标。

ii. 在**警告**对话框中，单击**继续运行**。

iii. 查看引用结果。

```
这是上游shell节点的输出hello,world
取上游shell节点输出的第1个数据hello
取上游shell节点输出的第2个数据 world
```

6.11.7. Shell节点

Shell节点支持标准Shell语法，不支持交互性语法。

背景信息

Shell节点仅支持使用**独享调度资源组**。详情请参见新增和使用独享调度资源组。

独享调度资源组上运行Shell节点时，如果您需要访问的目标端有白名单限制，请添加独享调度资源组的白名单至目标端应用，详情请参见新增和使用独享调度资源组。

操作步骤

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. 鼠标悬停至**+新建**图标，单击**通用 > Shell**。

   您也可以打开相应的业务流程，右键单击**通用**，选择**新建 > Shell**。
3. 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。
4. 单击**提交**。
5. 编辑Shell节点。
   i. 编辑Shell节点代码。

   如果您需要在Shell中调用系统调度参数，Shell语句如下所示。

   ```
   echo "$1 $2 $3"
   ```

   > **说明** 参数1 参数2...多个参数之间用空格分隔。更多系统调度参数的使用，请参见调度参数概述。

   i. 单击工具栏中的图标，将其保存至服务器。

ii. 单击工具栏中的图标，执行编辑的Shell语句。

如果您需要修改在**数据开发**页面测试时使用的任务执行资源，请单击工具栏中的图标，选择相应的独享调度资源组。

6. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性。详情请参见配置基础属性。
7. 保存并提交节点。

> **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

i. 单击工具栏中的图标，保存节点。

ii. 单击工具栏中的图标。

iii. 在**提交新版本**对话框中，输入**变更描述**。

iv. 单击**确认**。

如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

8. 测试节点，详情请参见查看并管理周期任务。

6.11.8. 虚拟节点

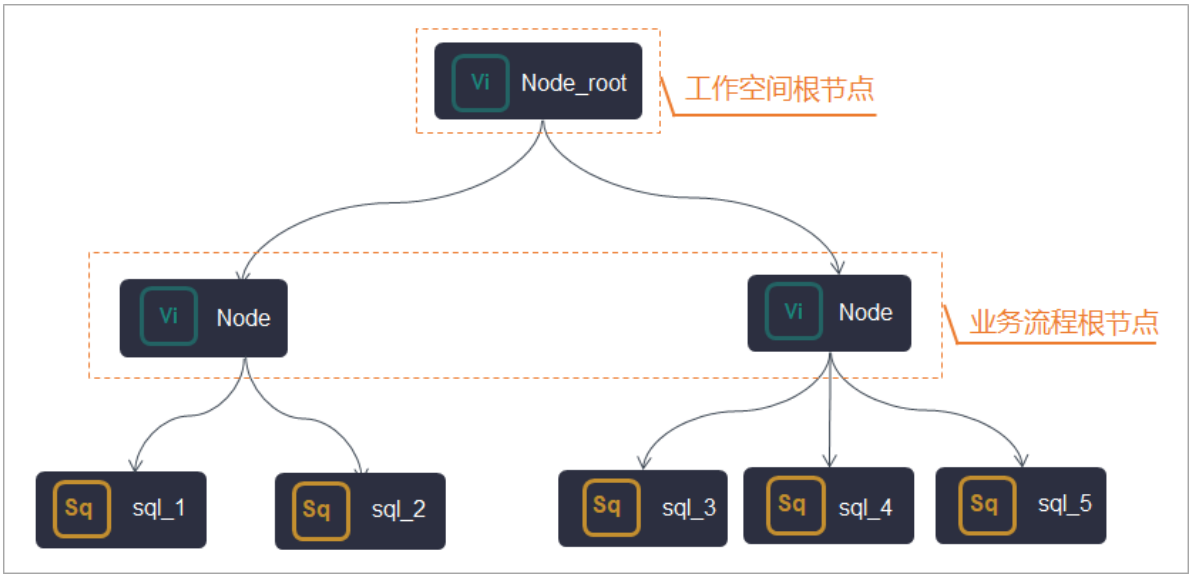
虚拟节点属于控制类节点，它是不产生任何数据的空跑节点（即调度到该节点时，系统直接返回成功，不会真正执行、不会占用资源或阻塞下游节点运行），通常作为业务流程的统筹起始节点，或业务流程中多个分支节点的汇总输出节点使用。本文为您介绍虚拟节点的应用场景及创建使用。

应用场景

虚拟节点通常用于如下场景：

- **复杂依赖场景下的业务管理**

  当您的实际业务包含多个业务流程时，为了业务流程与业务流程之间便于管理，建议每个业务流程都使用虚拟节点设置一个空跑的统筹起始节点，使数据流转路径更加清晰。

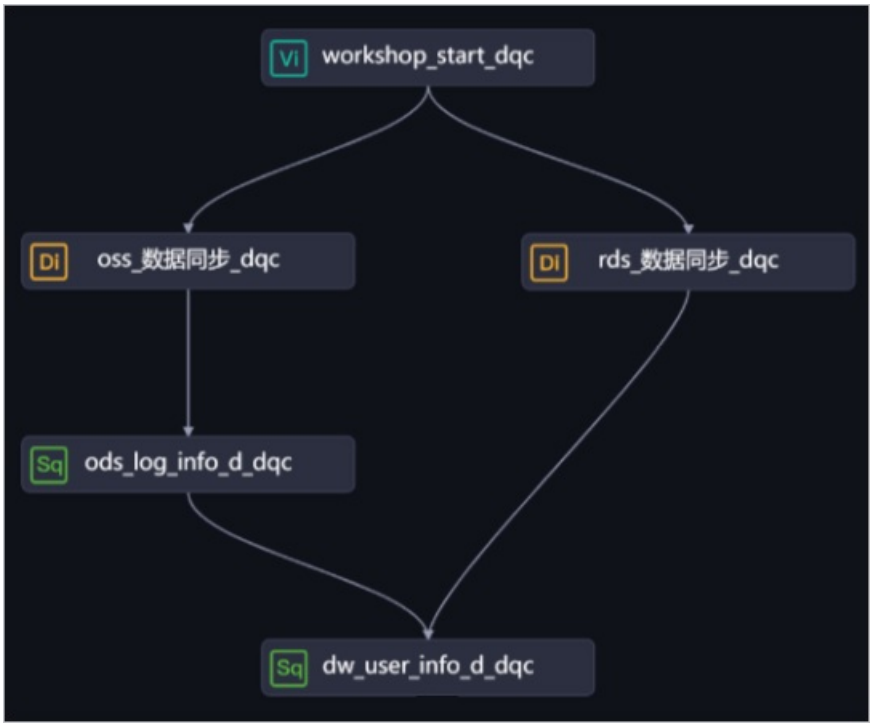


- **调度无血缘关系的节点**

当业务流程中的最终输出节点有多个分支输入节点，且输入节点没有依赖关系时，您需要将虚拟节点作为多个输入节点的上游，将工作空间根节点作为虚拟节点的上游，实现工作空间根节点调度该虚拟节点，虚拟节点调度下游业务节点。当整个业务流程需要统一调度时间时，您也可以使用该方式指定虚拟节点的定时时间，来控制各分支节点的最早调度运行时间。

> **说明** 工作空间根节点作为上游依赖时，不会呈现在业务流程面板中。您可以在任务提交发布后，进入运维中心查看。运维中心详情请参见周期任务运维概述。

示例如下。



`oss_数据同步_dqc` 、 `rds_数据同步_dqc` 节点不存在血缘关系，不能根据血缘关系来设置节点的调度依赖。此时，您可使用虚拟节点（ `workshop_start_dqc` ）作为统筹起始节点，统一调度下游无血缘关系的分支节点，当下游分支节点满足运行条件时便会启动运行。

> **说明** 通过离线同步将其他数据源中的数据同步至DataWorks，对于DataWorks上接收同步数据的表来说，在DataWorks侧不存在上游血缘关系。

- **管理多分支结果的业务流程，实现跨业务流程的调度依赖**

  包含多个分支结果的业务流程如果要实现跨业务流程依赖，您需要使用虚拟节点对多个分支节点进行汇总，再手动将该汇总节点的输出作为下游业务流程统筹起始节点的输入，以此方式实现跨业务流程依赖。详情请参见跨业务流程配置调度依赖。

  > **说明** 一个业务流程存在多个分支结果时，您需要新建一个虚拟节点（例如，业务流程_end_虚拟节点），业务流程_end_虚拟节点依赖上游多个分支结果，当业务流程_end_虚拟节点执行成功，则表示该业务流程执行完成。

创建并使用虚拟节点

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. 选择工作空间所在地域后，单击相应工作空间后的**数据开发**。

2. 创建**业务流程**。

如果您已有**业务流程**，则可以忽略该步骤。

i. 鼠标悬停至+新建图标，选择**新建业务流程**。
ii. 在**新建业务流程**对话框，输入**业务名称**。
iii. 单击**新建**。

3. 创建虚拟节点。

i. 鼠标悬停至+新建图标，选择**新建节点 > 通用 > 虚拟节点**。

您也可以找到相应的业务流程，右键单击业务流程，选择**新建节点 > 通用 > 虚拟节点**。

ii. 在**新建节点**对话框中，输入**名称**，并选择**节点类型**及**路径**。

> **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**，进入虚拟节点编辑页面。

4. 任务调度配置。

如果您需要周期性执行创建的节点任务，可以单击节点编辑页面右侧的**调度配置**，根据业务需求配置该节点任务的调度信息：

- 配置任务调度的基本信息，详情请参见配置基础属性。
- 配置时间调度周期、重跑属性和上下游依赖关系，详情请参见时间属性配置说明及配置同周期调度依赖。

> **说明** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

- 配置资源属性，详情请参见配置资源属性。

5. 提交并发布节点任务。

i. 单击工具栏中的图标，保存节点。
ii. 单击工具栏中的图标，提交节点任务。
iii. 在**提交新版本**对话框中，输入**变更描述**。
iv. 单击**确定**。

如果您使用的是标准模式的工作空间，任务提交成功后，需要将任务发布至生产环境进行发布。请单击顶部菜单栏左侧的**任务发布**。具体操作请参见发布任务。

6. 查看周期调度任务。

i. 单击编辑界面右上角的**运维**，进入生产环境运维中心。
ii. 查看运行的周期调度任务，详情请参见查看并管理周期任务。

如果您需要查看更多周期调度任务详情，可单击顶部菜单栏的**运维中心**，详情请参见运维中心概述。

6.11.9. HTTP触发器节点

如果您有其他调度系统，希望在调度系统的任务完成后触发DataWorks上的任务运行，您可以使用DataWorks的HTTP触发器节点功能。本文为您介绍外部调度系统触发场景下，使用DataWorks的HTTP触发器节点的流程和注意事项。

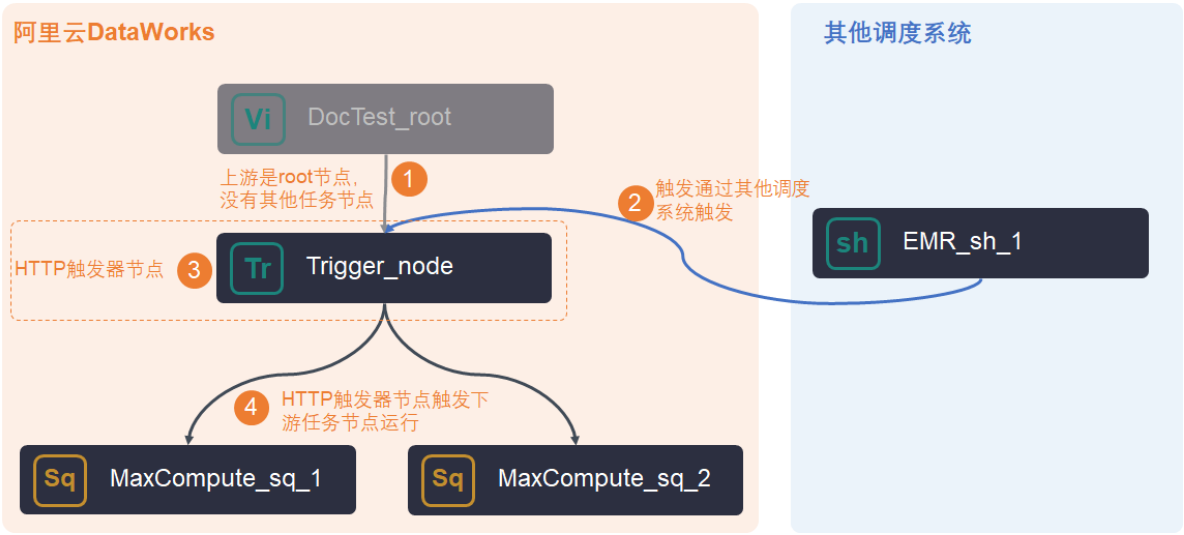
前提条件

- 已开通DataWorks企业版及以上版本。
- 已创建好业务流程和需要通过HTTP触发器节点触发的计算任务节点。以使用MaxCompute的SQL进行任务计算为例，您可参见创建ODPS SQL节点，创建完成MaxCompute的SQL计算节点。

背景信息

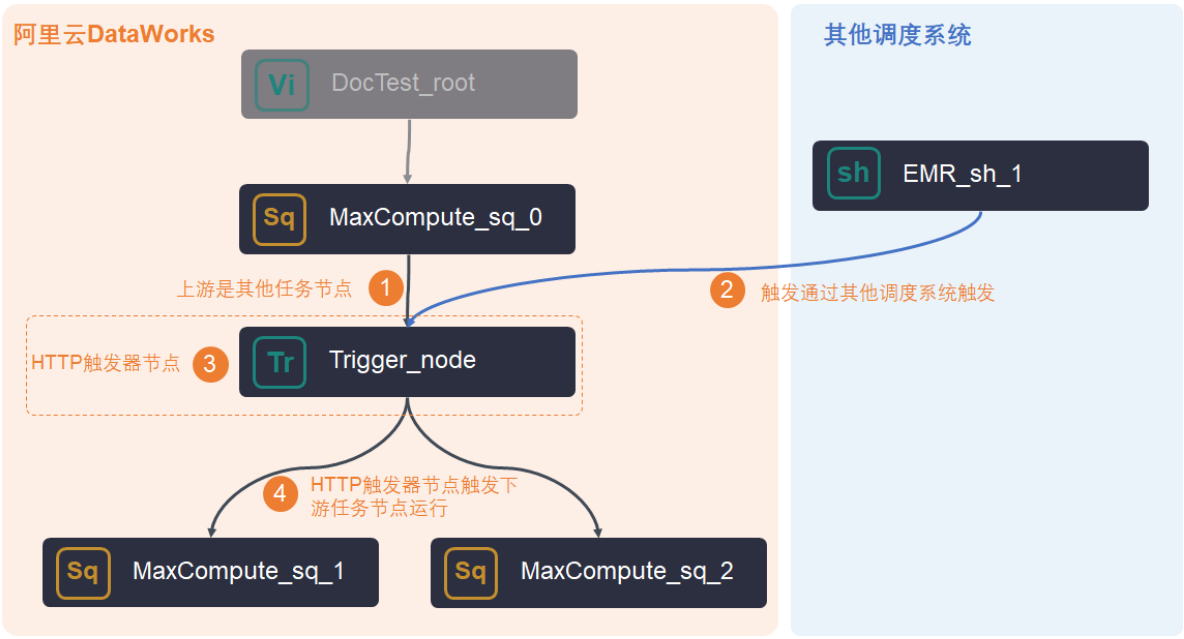
外部调度系统触发任务运行有以下两种典型场景：

- HTTP触发器节点无其他上游任务节点



此种场景下，您需要创建HTTP触发节点后，在其他调度系统配置好调度触发，并在DataWorks上配置好各节点的调度和上下游依赖关系，详情可参见创建HTTP触发器节点和其他调度系统的触发配置。

- HTTP触发器节点有上游任务节点



此种场景下：

- 您需要创建HTTP触发节点后，在其他调度系统配置好调度触发，并在DataWorks上配置好各节点的调度和上下游依赖关系，详情可参见创建HTTP触发器节点和其他调度系统的触发配置。
- HTTP触发器节点默认上游节点为业务流程的根节点，当上游有其他任务节点时，您需要手动修改为对应的上游任务节点。
- 当上游任务节点运行完成，且外部调度系统发出调度指令后，HTTP触发节点才会触发下游任务节点运行。

  如果外部调度系统提前发出调度指令，但是上游任务节点没有运行完成，HTTP触发节点不会触发下游任务节点。系统会保留外部调度系统的调度指令，待上游任务运行完成后，再通过HTTP触发节点触发下游任务节点运行。

> **注意** 外部调度系统的触发指令仅保留24小时。如果24小时内上游任务节点没有运行完成，则触发指令会丢失，外部调度系统本次发出的调度指令失效。

使用限制

- HTTP触发器节点功能仅适用DataWorks企业版及以上版本，当前HTTP触发器节点功能已在国内地域、新加坡地域和德国（法兰克福）地域部署上线。
- HTTP触发器节点仅作为触发节点，不可以直接写计算运行任务，您需要将待运行的任务节点作为HTTP触发器节点的下游节点。
- 业务流程创建完成正常运行后，如果您想重跑触发器节点，您同时需要在外部调度系统中重新下发触发指令。
- 业务流程创建完成正常运行后，如果您想获取触发器节点的下游任务节点的历史时间段的运行结果，您可参见执行补数据并查看补数据实例进行补数据操作。补数据操作时无需外部调度系统下发调度指令，HTTP触发器节点会直接触发下游节点运行。

触发说明

触发HTTP触发器节点需要满足以下条件：

- HTTP触发器节点已经生成周期实例（在运维中心**周期实例**面板可以搜到该实例）。

- HTTP触发器节点所依赖的所有父节点都已经执行成功（实例为成功状态）。
- HTTP触发器节点生成的周期实例定时时间已到。
- HTTP触发器节点使用的调度资源组，在触发时间点资源充足。
- HTTP触发器节点处于非冻结状态。
- 仅等待触发状态下的HTTP触发器节点才可被触发（已触发成功过的再次触发将不会执行）。

创建HTTP触发器节点

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. 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，鼠标悬停至+新建图标，单击**通用 > HTTP触发器**。

   您也可以打开相应的业务流程，右键单击**通用**，选择**新建 > HTTP触发器**。
3. 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。

4. 单击**提交**。
5. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性，详情请参见配置基础属性。

   > **说明** HTTP触发器节点默认上游节点为业务流程的根节点，当上游有其他任务节点时，您需要手动修改为对应的上游任务节点。

6. 保存并提交节点。

   > **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标，保存节点。
   ii. 单击工具栏中的图标。
   iii. 在**提交新版本**对话框中，输入**变更描述**。
   iv. 单击**确认**。

   如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。
7. 测试节点，详情请参见查看并管理周期任务。

其他调度系统的触发配置

在外部调度系统中进行触发配置时，您可以通过以下三种方式：Java方式、Python方式或API调用方式。

- Java方式
  i. 安装Java SDK，详情可参见开始使用。

  其中，DataWorks的SDK请用下面的pom配置。

```
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-dataworks-public</artifactId>
  <version>3.4.2</version>
</dependency>
```

ii. 代码示例

```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.exceptions.ServerException;
import com.aliyuncs.profile.DefaultProfile;
import com.google.gson.Gson;
import java.util.*;
import com.aliyuncs.dataworks_public.model.v20200518.*;
public class RunTriggerNode {
public static void main(String[] args) {
// 设置RegionId、访问密钥和访问密码。
// 其中：cn-hangzhou表示任务所在的地域，即RegionId。
// <accessKeyId>表示访问密钥。
// <accessSecret>表示访问密码
DefaultProfile profile = DefaultProfile.getProfile("cn-hangzhou", "<accessKeyId>", "<
accessSecret>");
IAcsClient client = new DefaultAcsClient(profile);
RunTriggerNodeRequest request = new RunTriggerNodeRequest();
// 设置NodeId，表示触发式节点的节点ID，节点ID可通过ListNodes API查询获取
request.setNodeId(700003742092L);
// 设置CycleTime，表示触发式节点的任务的运行时间戳，需将HTTP触发节点的调度配置中，节点指定的运
行时间，换算为时间戳
// 如果HTTP触发节点所在的地域与调度系统所在的地域不在同个时区，存在时差，这里需配置为触发节点所
在时区的时间。
// 例如，HTTP触发节点在北京地域且Cyctime为北京时间18：00，而调度系统在美西地域，此时调度系统配
置时，需配置为北京时间18：00的时间戳。
request.setCycleTime(1605629820000L);
// 设置BizDate，表示触发式节点实例所在的业务日期时间戳，需将业务日期换算为时间戳。
// 业务日期为运行时间的前一天，且时间精确到日，时分秒均为00000000。以运行日期为2020年11月25日为
例，业务时间为20201124000000000，需将这个时间换算为业务日期的时间戳
// 如果HTTP触发节点所在的地域与调度系统所在的地域不在同个时区，存在时差，这里需配置为触发节点所
在时区的时间。
request.setBizDate(1605542400000L);
// 设置AppId，表示触发式节点所属的Dataworks工作空间ID，工作空间ID可通过ListProjects获取
request.setAppId(123L);
try {
RunTriggerNodeResponse response = client.getAcsResponse(request);
System.out.println(new Gson().toJson(response));
} catch (ServerException e) {
e.printStackTrace();
} catch (ClientException e) {
System.out.println("ErrCode:" + e.getErrCode());
System.out.println("ErrMsg:" + e.getErrMsg());
System.out.println("RequestId:" + e.getRequestId());
}
}
}
```

- Python方式

  i. 安装Python SDK，详情可参见安装。

  其中，DataWorks的SDK请使用下面的命令安装。

```
pip install aliyun-python-sdk-dataworks-public==2.1.2
```

ii. 代码示例

```
#!/usr/bin/env python
#coding=utf-8
from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.acs_exception.exceptions import ClientException
from aliyunsdkcore.acs_exception.exceptions import ServerException
from aliyunsdkdataworks_public.request.v20200518.RunTriggerNodeRequest import RunTrig
gerNodeRequest
# 设置RegionId 访问密钥 访问密码
# cn-hangzhou 表示任务所在的地域，即RegionId
# <accessKeyId> 访问密钥
# <accessSecret> 访问密码
client = AcsClient('<accessKeyId>', '<accessSecret>', 'cn-hangzhou')
request = RunTriggerNodeRequest()
request.set_accept_format('json')
# 设置NodeId，表示触发式节点的节点ID，节点ID可通过ListNodes API查询获取ID
request.set_NodeId(123)
# 设置CycleTime，表示触发式节点的任务的运行时间戳，需将HTTP触发节点的调度配置中，节点指定的运行时间，换算为时间戳
# 如果HTTP触发节点所在的地域与调度系统所在的地域不在同个时区，存在时差，这里需配置为触发节点所在时区的时间。
# 例如，HTTP触发节点在北京地域且Cyctime为北京时间18：00，而调度系统在美西地域，此时调度系统配置时，需配置为北京时间18：00的时间戳。
request.set_CycleTime(1606321620000)
# 设置BizDate，表示触发式节点实例所在的业务日期时间戳，需将业务日期换算为时间戳。
# 业务日期为运行时间的前一天，且时间精确到日，时分秒均为00000000。以运行日期为2020年11月25日为例，业务时间为20201124000000000，需将这个时间换算为业务日期的时间戳
# 如果HTTP触发节点所在的地域与调度系统所在的地域不在同个时区，存在时差，这里需配置为触发节点所在时区的时间。
request.set_BizDate(1606233600000)
# 设置AppId，表示触发式节点所属的Dataworks工作空间ID，工作空间ID可通过ListProjects获取
request.set_AppId(11456)
response = client.do_action_with_exception(request)
# python2: print(response)
print(str(response, encoding='utf-8'))
```

- API调用方式

  API调用方式可参见RunTriggerNode。

6.11.10. 参数节点

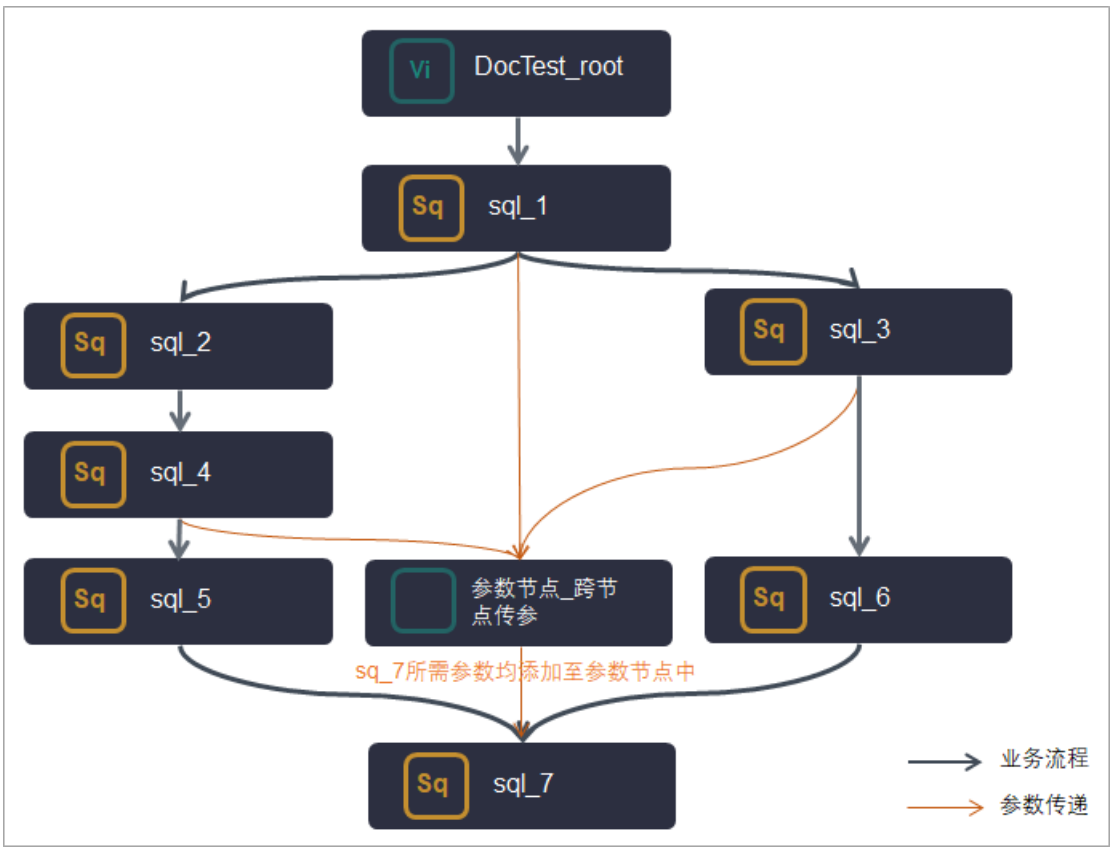
参数节点是一种特殊的虚拟节点，用于管理业务流程中的参数和实现参数在任务节点中传递，支持常量参数、变量参数和透传上游节点的参数，需要引用参数的节点直接依赖参数节点即可。本文为您介绍如何创建参数节点，引导您高效使用DataWorks进行数据开发。

背景信息

参数节点本质上是一种虚拟节点，不会运行数据计算任务产生数据，主要用于跨节点传参、参数管理的场景。

- **跨节点传参**

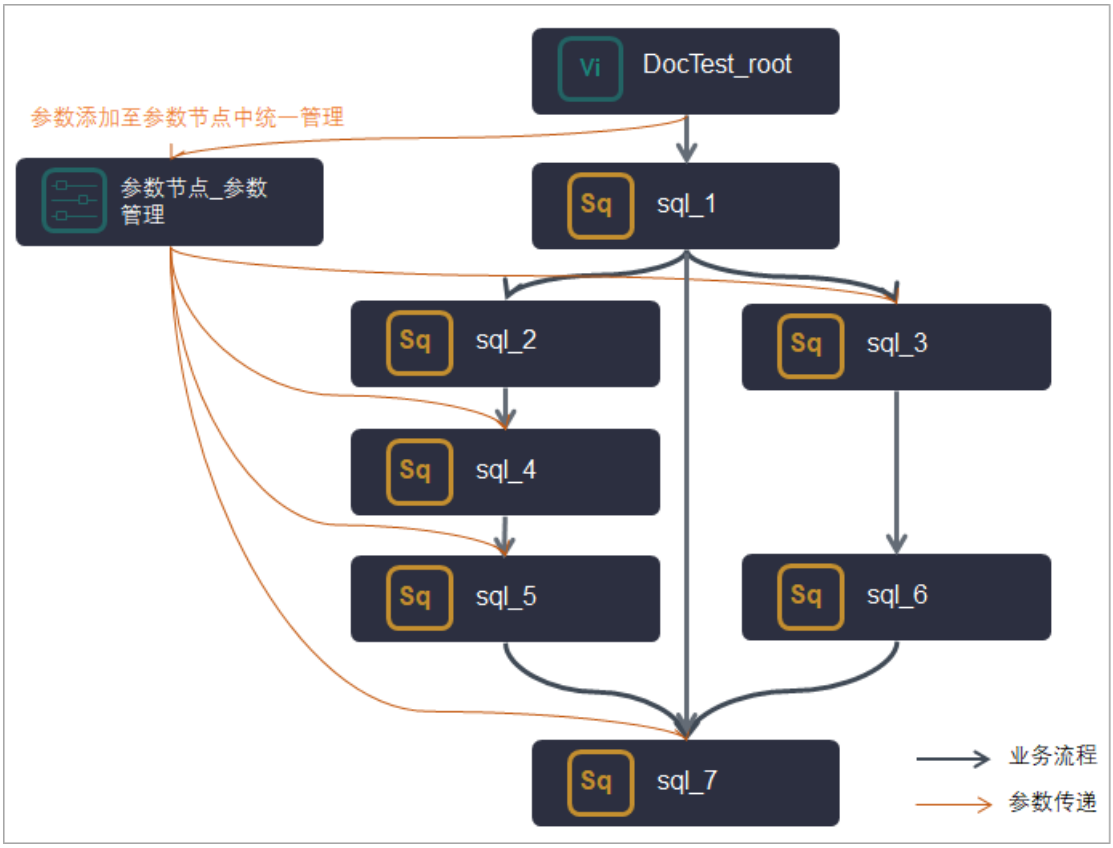
当数据开发的业务流程中，某个下游节点的任务需要获取多个、多级上游节点的输出参数时，您可以使用参数节点，将下游节点需要获取的所有参数统一添加至参数节点中，后续下游节点可直接挂在参数节点之下，即可获取到所有所需参数。



以上图为例，sql_7节点需要获取sql_1、sql_3、sql_4节点的输出参数，此时您可以新增一个参数节点，作为sql_1、sql_3、sql_4的下游节点，并将所有sql_7所需参数添加至参数节点中，将sql_7的挂在此参数节点下游，则sql_7可直接通过参数节点获取到所有所需参数。

- **参数管理**

  当数据开发的业务流程中，下游节点的任务需要使用某些常量参数、变量参数时，您可以使用参数节点，将下游节点需要使用的参数均添加至参数节点中，需使用参数的下游节点直接挂在参数节点之下，即可获取使用所需参数，便于整个业务流程中对所有使用的参数进行统一管理。



以上图为例，sql_3、sql_4、sql_5、sql_7节点均需使用参数，此时您可以新增一个参数节点，将各个下游节点使用的参数都添加至参数节点中，将需要使用参数的节点挂在此参数节点下游。

注意事项

某个任务节点引用参数节点中的参数时，需在业务流程中处于参数节点的直接下游，将参数节点作为本节点的上游依赖。

新建参数节点

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. 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，鼠标悬停至+新建图标，单击**通用 > 参数节点**。

   您也可以打开相应的业务流程，右键单击**通用**，选择**新建 > 参数节点**。
3. 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。
4. 单击**提交**。

配置调度

参数节点创建完成后，您可根据参数节点的应用场景完成参数节点的调度配置。

调度配置包括节点的基础属性、时间属性、资源属性和调度依赖。由于参数节点不运行数据开发任务，仅用于参数管理和透传参数，因此参数节点的调度配置需重点关注调度依赖的配置：

- 任务节点使用参数节点中的参数时，需作为参数节点的下游依赖。
- 产生透传参数的上游节点需作为参数节点的上游依赖。

配置调度的详细操作步骤可参见配置基础属性、时间属性配置说明、配置资源属性、配置同周期调度依赖等章节。

添加参数

完成参数节点的调度配置后，您可将需要管理、需要透传的参数添加至参数节点中，便于后续管理使用。操作步骤如下。

1. 在参数节点的右侧编辑页面单击**新增参数**。
2. 完成**参数名**、**类型**、**取值**、**描述**的配置后，单击**保存**。

参数类型包括**常量**、**变量**和**透传变量**。

- **常量**：参数取值为一个固定值。
- **变量**：参数取值为变量，如果您需要使用系统时间等这类变量参数时，添加参数的参数类型需选择为变量。变量参数的详细介绍可参见调度参数概述。
- **透传变量**：透传变量主要用于将上游节点的产出参数透传至下游节点，参数取值可选择参数节点调度依赖中上游依赖节点的所有输出参数。

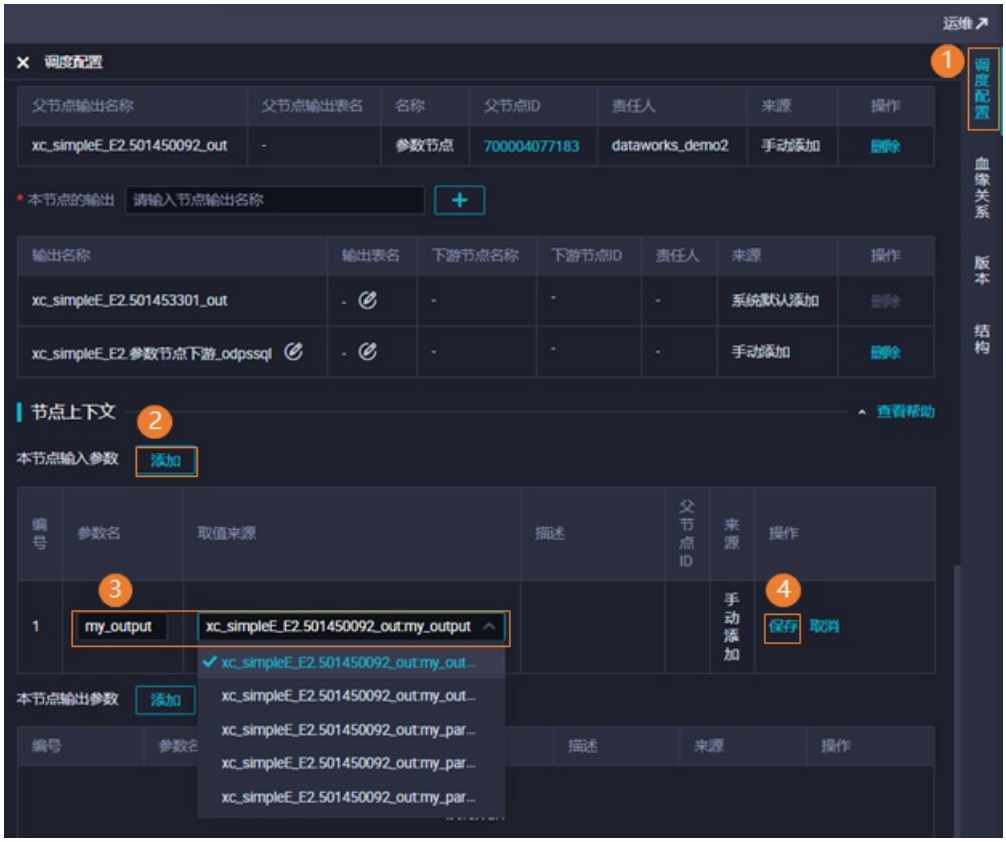
后续步骤：下游节点使用参数

完成参数节点的配置后，您可以在下游节点中直接使用参数节点中的参数，统一管理参数，提高下游节点任务开发效率。

参数节点的下游节点使用参数时，需要在节点上下文配置里引用上游节点参数，然后才能在任务代码中引用参数。

1. 设置下游节点的上下文配置。

   在下游节点的**调度配置 > 节点上下文**的**本节点输入**中，单击**添加**，将需要使用的参数添加进来。



其中：

- 参数名：需要使用的参数的名称，您可以在参数节点中查看参数名。
- 取值来源：选择取用哪个参数节点中的哪个参数。

  当内容较长看不全时，您可以将鼠标悬浮在可选的取值来源上，查看完整的参数来源信息。可选的参数取值来源为本节点上游节点中的所有参数，格式为**节点输出名称:参数名称**，您可以根据后缀参数名称快速找到参数对应的取值来源。

2. 在下游节点的代码编辑时，直接使用参数。

6.11.11. FTP Check节点

FTP Check节点可用于通过FTP协议周期性检测指定文件是否存在。如果文件存在，则启动调度下游任务，不存在，则按照配置的间隔时间重复检测，直到满足检测的停止条件时停止检测。该节点通常作为DataWorks调度系统与其他调度系统之间传递信号使用。本文为您介绍使用FTP Check节点的流程和注意事项。

前提条件

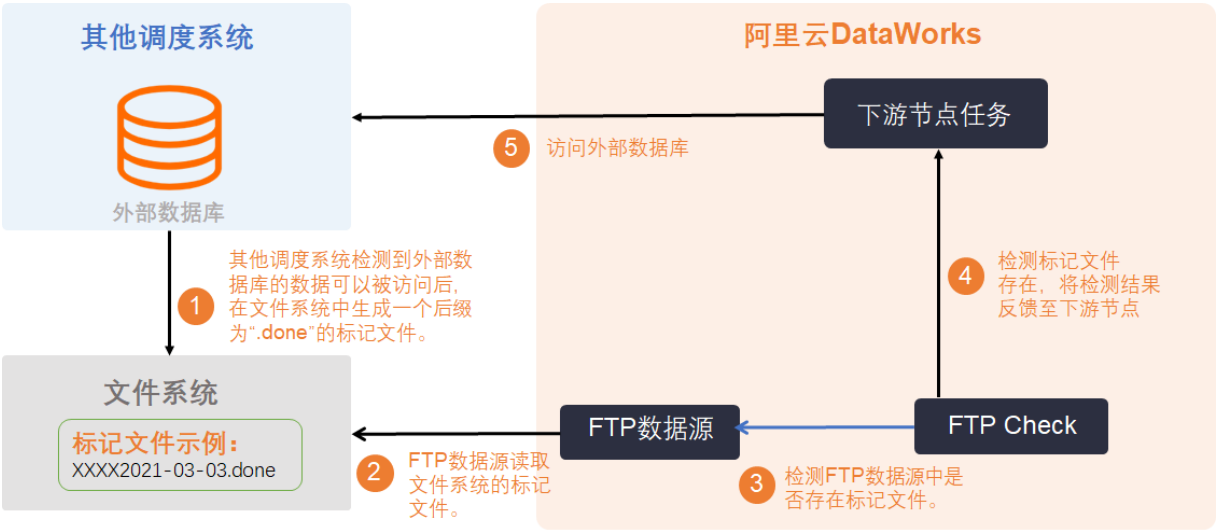
- 已创建FTP数据源。
- 已创建好业务流程，详情请参见创建业务流程。

背景信息

FTP Check节点的典型应用场景：当DataWorks调度系统中的任务需要访问一个外部数据库时，但由于该数据库的相关数据写入任务不在DataWorks中，DataWorks无法知道该数据库何时完成写入任务并可以被访问。如果DataWorks读取未写入完成的数据，则可能导致读取的数据不全或读取失败。为了保证DataWorks成功读取完整的外部数据库的数据，此时，可以让其他调度系统在数据库中的数据写入任务完成后，在指定文件系统中进行文件标记（例如，生成一个 `.done` 文件），表明该任务已完成。然后在DataWorks中配置一个FTP Check节点，周期性检测该 `.done` 文件是否存在，当检测到文件存在时，表明该数据库中的数据写入任务已经完成，可以启动调度需要访问该数据库数据的任务。

> **说明**
> - 其他调度系统可以自行指定生成标记文件的文件系统。
> - 本文以生成的标记文件为 `.done` 示例，在实际使用中，您可以自定义标记文件的格式、名称等信息。

具体如下：



1. 其他调度系统检测到外部数据库的数据已就绪（即数据写入已完成，可以被访问），会在指定的文件系统中生成一个标记文件，例如，`xxxx2021-03-03.done`。本文以 `.done` 为后缀的文件做为标记文件，您也可以根据业务需求，自定义标记文件。
2. FTP数据源读取文件系统中的标记文件。
3. FTP Check节点根据配置的检测策略，定期检测FTP数据源中该标记文件是否存在。
   - 如果检测该标记文件存在，则表示外部数据库中的数据已准备就绪，可以被访问，FTP Check节点会将检测成功的结果反馈至下游节点。
   - 如果检测该标记文件不存在，则表示外部数据库中的数据未准备就绪，不能被访问，FTP Check节点会将检测失败并且不会调度下游节点的结果反馈至下游节点，并根据配置的检测策略继续检测，直到达到预设的检测上限后停止检测。

   FTP Check节点的检测策略，请参见下文*配置检测对象及检测策略*。
4. 下游节点根据FTP Check节点的反馈结果，选择是否启动访问外部数据库的数据。
   - 如果FTP Check节点反馈检测成功，则下游节点启动访问外部数据库的数据。
   - 如果FTP Check节点反馈检测失败，则下游节点不启动访问外部数据库的数据。
5. 下游节点访问外部数据库的数据。

> **说明** 外部数据库可以包含但不限于Oracle、MySQL、SQLServer等各类数据库或存储服务。

使用限制

- FTP Check节点仅支持使用独享调度资源组。
- 分钟和小时周期调度的任务，FTP Check节点的**Check停止策略**不支持配置为**Check停止时间**，该类任务您只能选择使用**Check停止次数**停止策略。

创建FTP Check节点

1. 登录DataWorks控制台。
2. 在**数据开发**页面，鼠标悬停至+新建图标，单击**通用 > FTP Check**。

   您也可以打开相应的业务流程，右键单击**通用**，选择**新建 > FTP Check**。
3. 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（_）和英文句号（.），且不能超过128个字符。
4. 单击**提交**。
5. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性。

   调度属性包括**基础属性**、**时间属性**、**资源属性**和**调度依赖**，详情请参见 配置基础属性、时间属性配置说明、配置资源属性及配置同周期调度依赖。
6. 配置检测对象及检测策略。

   i. 在**选择FTP数据源**下拉列表，选择需要检测的目标FTP数据源。

   您可以选择FTP或SFTP数据源，如果下拉列表没有可用的数据源，则您需要新建数据源，详情请参见配置FTP数据源。

   ii. 在**指定Check的文件**配置需要检测的文件路径。如果您的文件路径是动态变化的，则您可以在文件路径中使用调度参数来配置变量路径，详情请参见调度参数概述。

   i. 在**Check间隔（秒）**中配置定时检测的时间间隔。

ii. 在**Check停止策略**中配置停止检测的策略。

- **Check停止时间**：检测任务的到时时间点，格式为`hh24:mi:ss`，即24小时制时间。每次执行检测任务时，如果没有检测到对应的标记文件，则该检测任务失败，不会触发启动调度下游任务，只有检测成功时，才会启动调度下游任务。检测失败后，该任务会按配置的间隔时间继续检测，直到到达配置的停止检测时间，才不再继续检测。如果检测失败，您可以在任务日志中查看具体的失败原因。

  > **说明** FTP Check节点的调度周期配置结果会影响FTP Check的停止策略：
  >
  > - 当调度周期配置为**分钟**或**小时**时，停止策略不支持配置为**Check停止时间**，只能配置为**Check停止次数**。详情请参见配置FTP Check的检测策略。
  > - 当调度周期开始配置为**天**，并且已经配置好停止策略为**Check停止时间**，此时如果将调度周期修改为**分钟**或**小时**，则停止策略**Check停止时间**选项无效，您需要重新配置停止策略为**Check停止次数**，否则FTP Check节点无法提交。

- **Check停止次数**：检测次数限制。每次执行检测任务时，如果没有检测到对应的标记文件，则该检测任务失败，不会触发启动调度下游任务，只有检测成功时，才会启动调度下游任务。检测失败后，该任务会按配置的间隔时间继续检测，直到到达配置的停止检测次数，才不再继续检测。如果检测失败，您可以在任务日志中查看具体的失败原因。

7. 保存并提交节点。

   > **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标，保存节点。

   ii. 单击工具栏中的图标。

   iii. 在**提交新版本**对话框中，输入**变更描述**。

   iv. 单击**确认**。

   如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

8. 测试节点，详情请参见查看并管理周期任务。

6.12. 自定义节点

6.12.1. 节点配置

6.12.1.1. 概述

DataStudio（数据开发）不仅支持原生的ODPS SQL、Shell等系统节点，也支持自定义节点。

使用限制

仅DataWorks企业版及以上版本支持使用自定义节点。

进入节点配置页面

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**进入数据开发**。
4. 单击页面右上方的**节点配置**，默认进入**节点插件列表**。

   节点配置页面包括**节点插件列表**、**系统节点列表**、**自定义节点列表**和**数据质量插件列表**。

> **注意** 仅支持项目所有者和项目管理员在节点配置页面进行相关操作。

节点插件列表

插件是指节点的核心处理逻辑。以ODPS SQL节点为例，您在编辑器中编写的SQL，提交运行后，会用后台对应的插件来进行解析并执行。新增一个自定义节点，首先需要开发自定义插件的处理逻辑，目前仅支持Java语言。

节点插件列表页面为您展示所有的插件。您可以单击右上方的**新增**，添加不同类型的插件。详情请参见新增节点插件。

最新提交版本、**开发环境部署版本**和**生产环境部署版本**的版本显示逻辑如下：

- 如果节点是新创建的且未进行发布，显示**尚未发布**。
- 如果节点已经发布，显示具体的版本号和部署时间。
- 如果节点正在发布中，显示**发布中**。

系统节点列表

在**节点配置**页面的左侧导航栏，单击**系统节点列表**，查看系统节点的**节点名称**和**启用模块**（默认**数据开发**）。

> **说明** 系统节点列表为展示页面，您无法进行修改等操作。

自定义节点列表

在**节点配置**页面的左侧导航栏，单击**自定义节点列表**。您可以在该页面新增、查看、编辑和删除自定义节点，并可以在**数据开发**页面使用自定义节点。详情请参见新增自定义节点。

数据质量插件列表

DataWorks支持用户自定义、发布和使用数据质量插件，满足多样的数据质量定制化需求。

在**节点配置**页面的左侧导航栏，单击**数据质量插件列表**。您可以在该页面新增、配置、查看和删除数据质量插件。详情请参见新增数据质量插件。

6.12.1.2. 开发自定义插件包

DataWorks自定义节点中运行任务时，需要调用自定义插件，因此在使用自定义节点前您需要创建好自定义插件包，并上传发布至DataWorks，便于使用自定义节点运行任务时使用。本文为您介绍如何创建自定义插件包。

背景信息

DataWorks的自定义节点运行时，插件开发使用过程中涉及以下2个接口。

- `submitJob(String codeFilePath, List args)`：提交插件任务的接口。

  包含以下入参。

  - codeFilePath：为页面开发的代码存储绝对路径。
  - List args：为页面配置的调度参数，格式为 `{"key"="value","key2"="value2"...}` 。

  返回参数如下。

  - 0：表示任务运行成功。
  - 2：表示告知调度系统重新运行任务。
  - 1、4、或3：表示任务被终止。
  - 其他数值：表示任务运行失败。

- `killJob()`：此接口主要用于监听任务终止（kill）信号，然后会触发该接口调用。此接口没有入参和返回参数。

  > **说明** 若业务层面在9秒内未处理完毕，整个插件进程会被强制终止，整个任务运行失败。

下载依赖JAR

点击以下链接下载依赖JAR包：alisa-wrapper-face-1.0.0.jar。

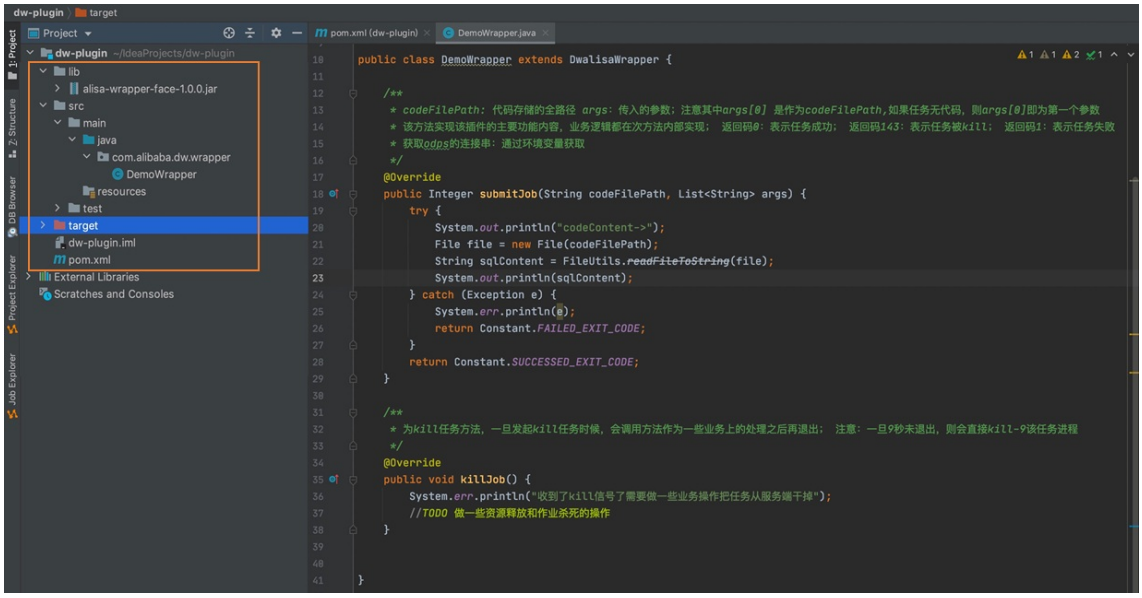
打包代码工程包

1. 新建插件代码工程。

   用IDE工具新建一个Maven工程，文件结构和pom.xml文件配置要求如下所示。

   - 文件结构要求

文件结构要求参考下图，其中lib文件夹下alisa-wrapper-face-1.0.0.jar为上述下载的依赖JAR包，需要用本地依赖引入的方式引入代码工程中。



- pom.xml文件配置要求

  pom.xml文件示例如下，其中插件的类名（ `<mainclass>` ）为submitJob方法所在的类路径，如 `com.alibaba.dw.wrapper` 。

```
<groupId>org.example</groupId>
<artifactId>dw-plugin</artifactId>
<version>1.0-SNAPSHOT</version>
<dependencies>
    <dependency>
        <groupId>com.alibaba.dw</groupId>
        <artifactId>alisa-wrapper-face</artifactId>
        <version>1.0.0-SNAPSHOT</version>
        <scope>system</scope>
        <systemPath>${basedir}/lib/alisa-wrapper-face-1.0.0.jar</systemPath>
    </dependency>
    <dependency>
        <groupId>commons-lang</groupId>
        <artifactId>commons-lang</artifactId>
        <version>2.4</version>
    </dependency>
    <dependency>
        <groupId>commons-io</groupId>
        <artifactId>commons-io</artifactId>
        <version>2.6</version>
    </dependency>
    <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-lang3</artifactId>
        <version>3.8.1</version>
    </dependency>
</dependencies>
<build>
    <plugins>
```

```
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-assembly-plugin</artifactId>
            <version>2.5.5</version>
            <configuration>
                <archive>
                    <manifest>
                        <mainClass>com.alibaba.dw.wrapper.DemoWrapper</mainClass>
                    </manifest>
                </archive>
                <descriptorRefs>
                    <descriptorRef>jar-with-dependencies</descriptorRef>
                </descriptorRefs>
            </configuration>
            <executions>
                <execution>
                    <id>make-assembly</id>
                    <phase>package</phase>
                    <goals>
                        <goal>assembly</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

2. 代码开发。

   代码示例一：基本逻辑

```
package com.alibaba.dw.alisa.wrapper;
import java.io.File;
import java.util.List;
import com.alibaba.dw.alisawrapper.DwalisaWrapper;
import com.alibaba.dw.alisawrapper.constants.Constant;
/**
**DwalisaWrapper 在alisa-wrapper-face包
**/
public class DemoWrapper extends DwalisaWrapper {
   /**
    * codeFilePath：代码存储的全路径。
    * args：传入的参数。注意其中args[0]是作为codeFilePath，如果任务无代码，则args[0]即为第一个参数。
    * 该方法实现该插件的主要功能内容，业务逻辑都在次方法内部实现。
    * 返回码0：表示任务成功。
    * 返回码1、4、3：表示任务被终止。
    * 返回码为其他数值：表示任务失败。
    */
   @SuppressWarnings("deprecation")
   @Override
   public Integer submitJob(String codeFilePath, List<String> args) {
       try {
           System.err.println("your code->");
           //此处实现业务代码，此方法执行完毕后任务执行结束。
       } catch (Exception e) {
           System.err.println(e);
       }
       System.out.println("task finished...");
       return Constant.SUCCESSED_EXIT_CODE;
   }
   /**
    * 为终止任务方法，一旦发起终止任务时，会调用方法作为一些业务上的处理之后再退出。
    * 注意：一旦9秒未退出，则会直接返回kill-9，终止该任务进程。
    *
    */
   @Override
   public void killJob() {
       System.err.println("收到了终止信号，需要做一些业务操作把任务从服务端终止");
   }
}
```

代码示例二：基于数据源开发的自定义节点

MysqlWrapper.java

```
package com.alibaba.dw.alisa.wrapper;
import java.io.File;
import java.nio.charset.StandardCharsets;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;
import java.util.List;
import org.apache.commons.io.FileUtils;
import com.alibaba.dw.alisa.wrapper.util.ConnectionManager;
```

```
import com.alibaba.dw.alisa.wrapper.util.SqlUtils;
import com.alibaba.dw.alisawrapper.DwalisaWrapper;
import com.alibaba.dw.alisawrapper.constants.Constant;
import com.alibaba.dw.alisawrapper.utils.TaskDirUtils;
import com.csvreader.CsvWriter;
public class MysqlWrapper extends DwalisaWrapper {
    private Connection conn = null;
    private Statement stmt = null;
    private static final int MAX_ROWS = 10000;
    /**
     * codeFilePath: 代码存储的全路径 args：传入的参数；注意其中args[0] 是作为codeFilePath,如果任务无代码，则args[0]即为第一个参数
     * 该方法实现该插件的主要功能内容，业务逻辑都在次方法内部实现； 返回码0：表示任务成功； 返回码143：表示任务被kill； 返回码1：表示任务失败 获取odps的信息：环境变量获取： id：ODPS_ACCESSID
     * key:ODPS_ACCESSKEY endpoint:ODPS_ENDPOINT
     */
    @Override
    public Integer submitJob(String codeFilePath, List<String> args) {
        try {
            System.out.println("code-content: ");
            File file = new File(codeFilePath);
            String sqlContent = FileUtils.readFileToString(file);
            String execContent = getParaValueContent(sqlContent, args);
            System.out.println(execContent);
            executeJdbcSql(execContent);
        } catch (Exception e) {
            System.err.println(e);
            return Constant.FAILED_EXIT_CODE;
        }
        return Constant.SUCCESSED_EXIT_CODE;
    }
    private String getParaValueContent(String sqlContent, List<String> args) throws Exc
eption {
        String content = sqlContent;
        if (args == null || args.size() <= 0) {
            return content;
        }
        // 替换${...}参数
        for (String keyValue : args) {
            System.out.println(args);
            if (keyValue.contains("=")) {
                String[] param = keyValue.split("=");
                String target = "${" + param[0] + "}";
                String replacement = param[1];
                content = content.replace(target, replacement);
            } else {
                System.err.println("param format is invalid, key=value");
                throw new Exception("param exception!");
            }
        }
        return content;
    }
    /**
     * 为kill任务方法，一旦发起kill任务时候，会调用方法作为一些业务上的处理之后再退出； 注意：一
```

```
旦9秒未退出，则会直接kill-9该任务进程
     */
    @Override
    public void killJob() {
        System.out.println("Accept kill signal...");
        try {
            if (stmt != null) {
                stmt.close();
            }
            if (conn != null) {
                conn.close();
            }
            System.out.println("Kill succeed");
        } catch (Exception e) {
            System.err.println("kill job error! " + e.getMessage());
        }
    }
    private void executeJdbcSql(String sqlContent) throws Exception {
        List<String> sqlList = SqlUtils.splitSql(sqlContent);
        try {
            /**
             * (1)认证，获取连接
             */
            System.out.println("Connecting to Server...");
            String jdbcConn = System.getenv("SKYNET_CONNECTION");
            // 获取连接串(json字符串)进行解析生成connection
            conn = ConnectionManager.getJDBCConnection(jdbcConn);
            System.out.println("Connected to Server!");
            stmt = conn.createStatement();
            stmt.setMaxRows(MAX_ROWS);
            for (String sql : sqlList) {
                System.out.println("start run... sql: " + sql);
                /**
                 * (2) 执行sql
                 */
                boolean hasResult = stmt.execute(sql);
                if (hasResult) {
                    /**
                     * (3) 写结果文件
                     */
                    storeResult(stmt.getResultSet());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
            System.err.println("sql execute failed! " + e.getMessage());
            throw e;
        } finally {
            if (stmt != null) {
                stmt.close();
            }
            if (conn != null) {
                conn.close();
            }
```

```
            System.out.println("release sql connection...");
            System.out.println("Job Finished!");
        }
    }
    protected void storeResult(ResultSet rs) throws Exception {
        ResultSetMetaData rsmd = rs.getMetaData();
        int columnsNumber = rsmd.getColumnCount();
        // 获取结果文件路径
        String resultFilePath = TaskDirUtils.getDataFile();
        FileUtils.writeStringToFile(new File(resultFilePath), "");
        CsvWriter csvWriter = new CsvWriter(resultFilePath, ',', StandardCharsets.UTF_8
);
        csvWriter.setTextQualifier('"');
        csvWriter.setUseTextQualifier(false);
        csvWriter.setForceQualifier(true);
        String[] headerList = new String[columnsNumber];
        for (int i = 0; i < columnsNumber; i++) {
            headerList[i] = rsmd.getColumnName(i + 1);
        }
        csvWriter.writeRecord(headerList);
        while (rs.next()) {
            String[] lineEles = new String[columnsNumber];
            for (int i = 0; i < columnsNumber; i++) {
                lineEles[i] = rs.getString(i + 1);
            }
            csvWriter.writeRecord(lineEles, true);
        }
        csvWriter.flush();
        csvWriter.close();
        System.out.println("store result finished.");
    }
}
```

ConnectionManager.java

```
public class ConnectionManager {
    public static Connection getJDBCConnection(String connectionJson) throws Exception
{
        Connection connection = null;
        try {
            String jdbcUrl = null;
            String userName = null;
            String password = null;
            if(StringUtils.isNotBlank(connectionJson)){
                JSONObject connectionObj = JSON.parseObject(connectionJson);
                jdbcUrl = connectionObj.getString("jdbcUrl");
                userName = connectionObj.getString("username");
                password = connectionObj.getString("password");
            }else{
                throw new Exception("member in connection is null!");
            }
            String driverClassName = getDriverClassName(jdbcUrl);
            System.out.println("load driver class: " + driverClassName);
            Class.forName(driverClassName);
            DriverManager.setLoginTimeout(20);
            connection = DriverManager.getConnection(jdbcUrl, userName, password);
        } catch (ClassNotFoundException e) {
            System.err.println("acquire connection failed! " + e);
            System.exit(1);
        }
        return connection;
    }
        private static String getDriverClassName(String jdbcUrl) throws Exception{
        if(StringUtils.contains(jdbcUrl, "jdbc:mysql")){
            return "com.mysql.cj.jdbc.Driver";
        }else{
            throw new Exception("unsupported jdbc driver");
        }
    }
```

3. 设置数据查询结果集被页面展示。

   如果您希望后续使用自定义节点时，进行数据查询等操作时，结果能在DataWorks的页面中直接展示，您需参考本步骤设置数据查询结果集被页面展示。

   从submitJob接口业务运行代码后，需将获取到的结果集（比如查询表记录）按照以下规则存储到指定目录中：

   - 结果集文件获取方式

     ```
     String dataFilePath =String.format("%s/%s.data", System.getenv(Constant.TASK_EXEC_PAT
     H), System.getenv(Constant.ALISA_TASK_ID));
     ```

     其中 `System.getenv` 为获取环境变量的方式，详细介绍可参考附录：通过环境变量获取对应的节点信息。

   - 存储格式
     - 文件格式：CSV格式，字段间以逗号分隔。
     - 行分布：首行为列名称，后续各行为具体字段的数据。

样例：

```
"name","age"
"李三","12"
```

4. 代码开发完成后进行Maven构建打包。

   将本身依赖的包打成JAR包。

5. 后续步骤：上传插件包。

   如果依赖了protobuf、guava的包，或者其他外部依赖JAR包，需将这些依赖的包和上述步骤打成的JAR包合并为ZIP包后，作为插件上传至DataWorks。上传插件的详细操作可参考新增节点插件。

附录：通过环境变量获取对应的节点信息

获取环境变量值的方式：`System.getenv("SKYNET_ONDUTY")`。

- SKYNET_ID：节点ID，只有在调度运行才生效，数据开发直接运行是无该参数的。
- SKYNET_BIZDATE：业务日期。
- SKYNET_ONDUTY：
  - 临时运行、补数据、测试时：表示操作者ID。
  - 日常调度：节点责任人ID（工号/baseid）。
- SKYNET_TASKID：节点实例。
- IDSKYNET_SYSTEM_ENV：对应的项目环境，包括：dev 、prod。
- SKYNET_CYCTIME：节点实例的定时时间。
- SKYNET_CONNECTION：连接串（一般是json）。
- SKYNET_TENANT_ID：租户ID。

6.12.1.3. 新增节点插件

新增节点插件包括基本设置、发布到开发环境、在开发环境测试和发布到生产环境四个步骤。

背景信息

插件是节点的核心处理逻辑，目前仅支持Java语言。以ODPS SQL节点为例，您在编辑器中编写的SQL，提交运行后，会使用后台对应的插件来解析并执行。新增一个自定义节点，首先需要开发自定义插件的处理逻辑。

操作步骤

1. 新增节点插件。
   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. 单击页面右上方的**节点配置**，默认进入**节点插件列表**。
   v. 在**节点插件列表**页面，单击右上方的**新增**。

vi. 在**请选择插件类型**对话框中，选中需要新增的类型（示例为**引擎型**），单击**确认**。

插件类型包括**引擎型**和**业务型**：

- **引擎型**：通过上传JAR或其它格式的代码包定义插件的功能，通常用于驱动自定义计算引擎。
- **业务型**：以开发手动业务流程的方式定义插件的功能，通常用于封装多个基础节点实现特定的业务逻辑。

2. 在**基本设置**页面，配置各项参数。

参数	描述
名称	插件的名称。仅支持字母、下划线（_）和数字，且以字母开头。
负责人	根据工作空间的成员进行选择。当选择其他用户时： ○ 如果您是项目管理员，则不能编辑其他用户的自定义插件。 ○ 如果您是项目所有者，则可以编辑其他用户的插件。
资源文件	包括**上传本地文件**和**使用OSS文件**两种方式。 **说明** ○ 仅选择**引擎型**的插件类型时，显示该参数。 ○ 本地文件上传方式最大支持50 MB，OSS下载方式最大支持200 MB。 单击**选择文件**，在**上传文件**对话框中选择相应的方式并进行配置，单击**确认**。
类名	插件实现的类的全路径名称。 **说明** 仅选择**引擎型**的插件类型时，显示该参数。

参数	描述
参数模板	根据您上传的资源文件设计您的参数内容。 **说明** 仅选择**引擎型**的插件类型时，显示该参数。
手动业务流程	单击**选择手动业务流程**，在下拉列表中选择需要的手动业务流程名称，单击**确认**。 **说明** 仅选择**业务型**的插件类型时，显示该参数。
版本号	新增时，请选择使用新版本。编辑和回滚时，请选择覆盖当前版本。
版本描述	对插件进行简单描述。

3. 单击**保存**后，再单击**下一步**。

   单击**保存**后，您可以保存修改的配置至数据库：
   - 如果修改的是基本信息（非插件包），您只需要保存即可生效，无需进行发布。
   - 如果修改的是JAR包，您必须进行发布才会生效。
4. 在**发布到开发环境**页面，确认内容无误后，单击**提交开发环境发布**，实时查看**发布进度**。
5. 发布成功后，单击**下一步**。
6. 在开发环境测试插件。
   i. 在**在开发环境测试**页面，输入**参数**、**环境变量**和**代码**。
   ii. 单击**开始测试**。
   iii. 确认测试结果无误后，选中**已检查，确认测试通过**。
   iv. 单击**下一步**。
7. 在**发布到生产环境**页面，单击**提交生产环境发布**，实时查看**发布进度**。

   说明 提交至生产环境的版本必须是在开发环境已经部署、测试通过的最新版本，否则生产环境会提示发布失败。
8. 单击**完成**，进入**插件列表**。

   您可以在该页面查看新建的插件，并进行**配置**、**查看全部版本**和**删除**等操作：
   - **配置**：单击**配置**后，会根据插件的状态，自动跳转至相应的页面。
   - **查看全部版本**：单击**查看全部版本**，在**全部版本**对话框中，查看、回滚或下载插件的版本，单击**确认**：
     - **查看**：单击后进入新增节点插件的基本信息页面。
     - **回滚**：提示使用旧版JAR和配置重新发布插件，但会更新版本号。回滚的版本号为全部版本页面中最大的版本号+1。
     - **下载**：单击**下载**，即可下载对应的资源文件。
   - **删除**：单击**删除**，在**删除插件**对话框中，单击**确认**。

> 注意　仅支持删除没有节点关联的插件。

6.12.1.4. 新增自定义节点

本文为您介绍如何通过设置基本信息、插件、编辑器和交互的方式，新增自定义节点。

前提条件

您需要开通DataWorks企业版及以上版本，才可以使用自定义节点。

操作步骤

1. 进入**自定义节点列表**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**。
   iv. 单击页面右上方的**节点配置**，默认进入**节点插件列表**。
   v. 在左侧导航栏，单击**自定义节点列表**。
2. 单击页面右上方的**新增**，进入**自定义节点配置**页面。
3. 在**基本信息**区域，配置各项参数。

参数	描述
名称	自定义节点的名称，请确保唯一性。仅支持英文字母、空格，且以字母开头。
图标	在下拉列表中选择需要的图标。
发布模块	支持**临时查询**、**手动业务流程**和**数据开发**。
描述	对自定义节点进行简单描述。
使用说明	您可以根据自身需求修改使用说明。

4. 在**插件配置**区域，选中**插件类型**后，在下拉列表中**选择插件**。
   插件类型包括**引擎型**和**业务型**：

- **引擎型**：通过上传JAR或其它格式的代码包定义插件的功能，通常用于驱动自定义计算引擎。
- **业务型**：以开发手动业务流程的方式定义插件的功能，通常用于封装多个基础节点实现特定的业务逻辑。

5. 在**编辑器设置**区域，配置各项参数。

参数	描述
编辑器类型	包括**编辑器**和**数据源选择+编辑器**。
是否使用MaxCompute引擎	如果您的插件需要使用MaxCompute引擎则必须开启，如果不需要则可以关闭。 **说明** 仅选择**编辑器类型**为**编辑器**时，显示该参数。
数据源类型	在下拉列表中选择数据源类型，仅支持以URL方式配置的数据源。 **说明** 仅选择**编辑器类型**为**数据源选择+编辑器**时，显示该参数。
编辑器语言类型	支持ODPS SQL、JSON、Shell、Python、MySQL、XML和YAML等类型。
代码模板	可用变量：作者${author}，创建时间{createTime}。
高级设置	节点运行时参数，参数格式需要自定义。
参数模板	设置自定义节点的参数模板。 **说明** 仅选中**高级设置**时，显示该参数。

参数	描述
节点中文Tips	节点Tips展示在节点编辑器工具栏的右侧，通常用于提示重要的信息，不得超过256个字符。
节点英文Tips	

6. 在**交互配置**区域，配置各项参数。

参数	描述
文件右键操作	○ 默认选项：重命名、移动、克隆、偷锁、查看历史版本、在运维中心定位、删除和发起Review。 ○ 可选项：编辑、复制文件名和添加为桌面快捷方式。
顶部操作按钮	○ 默认选项：保存、提交、提交并解锁、偷锁编辑、运行、折叠、高级运行（带参数运行）、停止、重新加载、在开发环境执行冒烟测试、查看开发环境冒烟测试日志、执行冒烟测试、查看冒烟测试日志、前往开发环境的调度系统和格式化。 ○ 可选项：运维中心、发布和预编译。
右侧Tab	○ 默认选项：调度配置和结构。 ○ 可选项：版本、血缘关系和参数配置。
自动解析	如果开启，调度配置界面会显示相关入口。如果关闭，则调度配置界面没有相关入口。自动解析是根据代码中的血缘关系，解析出本节点的输入及本节点的输出。

7. 单击**保存并退出**，进入**自定义节点列表**查看新建的自定义节点。

   启用模块包括**数据开发**、**手动业务流程**和**临时查询**，您可以根据自身需求进行选择。

目前仅项目所有者或节点创建人可以**编辑**和**删除**自定义节点：

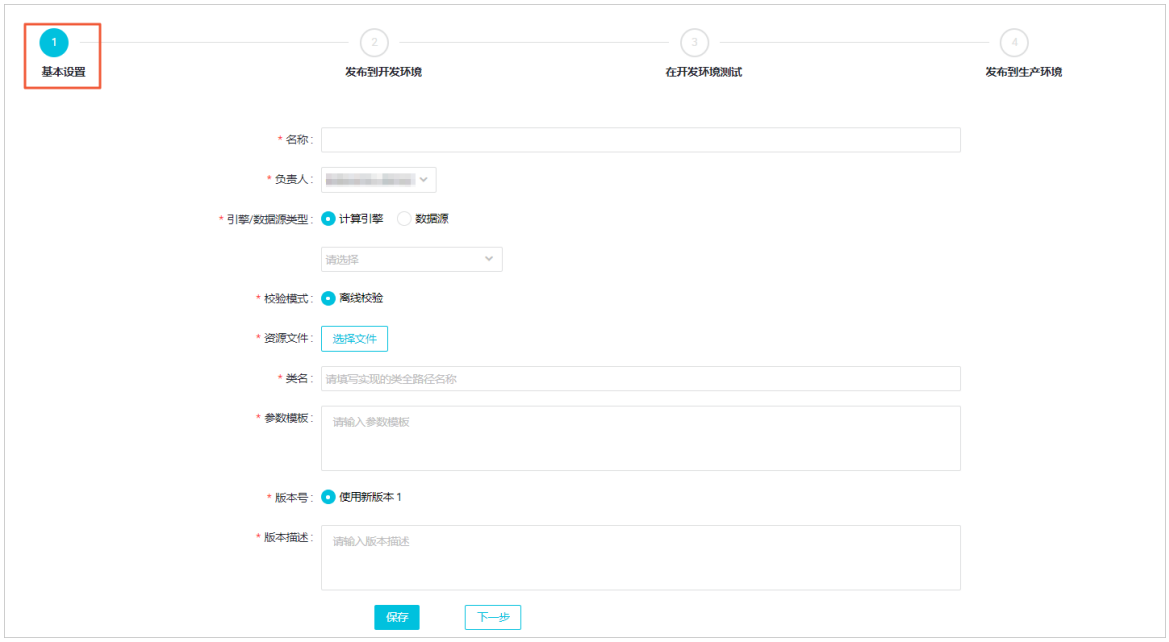
- **编辑**：单击**编辑**进入创建自定义节点页面，您可以根据自身需求更改相关的节点。
- **删除**：如果没有任务使用该节点，您可以直接删除。如果有任务使用该节点，会提示报错。您需要先下线任务，才可以进行删除操作。

6.12.1.5. 新增数据质量插件

DataWorks支持用户自定义、发布和使用数据质量插件，满足多样的数据质量定制化需求。

操作步骤

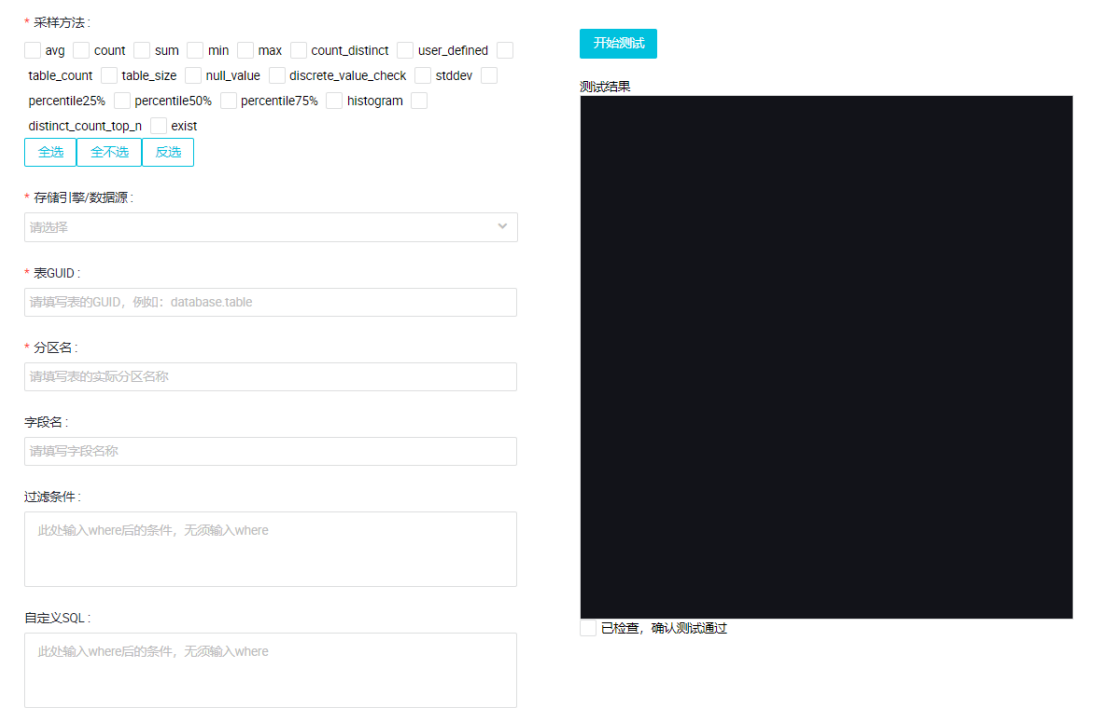
1. 新增数据质量插件。
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**工作空间列表**。
   iii. 选择工作空间所在地域后，单击相应工作空间后的**进入数据开发**。
   iv. 单击页面右上方的**节点配置**，默认进入**节点插件列表**。
   v. 在左侧导航栏，单击**数据质量插件列表**。
   vi. 在**数据质量插件列表**页面，单击右上方的**新增**。
2. 在**基本设置**页面，配置各项参数。



参数	描述

参数	描述
名称	数据质量插件的名称。 **说明** 插件名称必须以字母开头，且仅支持字母、下划线（_）和数字。
负责人	在下拉列表中选择新增插件的负责人。
引擎/数据源类型	包括**计算引擎**和**数据源**。选中类型后，再从下拉列表中选择相应的计算引擎或数据源。
校检模式	仅支持**离线校检**。
资源文件	包括**上传本地文件**和**使用OSS文件**两种方式。 单击**选择文件**，在**上传文件**对话框中选择相应的方式并进行配置，单击**确认**。
类名	数据质量插件实现的类的全路径名称。
参数模板	根据您上传的资源文件设计您的参数内容。
版本号	新增时，请选择使用新版本。编辑和回滚时，请选择覆盖当前版本。
版本描述	对插件进行简单描述。

3. 单击**保存**后，再单击**下一步**。
4. 在**发布到开发环境**页面，确认内容无误后，单击**提交开发环境发布**，实时查看**发布进度**。
5. 发布成功后，单击**下一步**。
6. 单击**下一步**，进入**发布到开发环境**对话框。确认基本设置无误后，单击**提交开发环境发布**。
   待提示**节点在开发环境发布成功**后，单击**下一步**，进入**在开发环境测试**对话框。
7. 在开发环境测试插件。
   i. 在**在开发环境测试**页面，配置左侧的各项参数。



参数	描述
采样方法	选中需要使用的采样方法。
存储引擎/数据源	选择存储的计算引擎或数据源。
表GUID	表的GUID，例如database.table。
分区名	表的实际分区名称。
字段名	字段的名称。
过滤条件	过滤语句。此处输入where后的条件，无需输入where。
自定义SQL	自定义SQL语句。

ii. 单击**开始测试**。

iii. 确认测试结果无误后，选中**已检查，确认测试通过**。

iv. 单击**下一步**。

8. 在**发布到生产环境**页面，单击**提交生产环境发布**，实时查看**发布进度**。
9. 单击**完成**，进入**插件列表**。

   您可以在该页面查看新建的插件，并进行**配置**、**查看全部版本**和**删除**等操作：

   - 单击**配置**后，会根据插件的状态，自动跳转至相应的页面。
   - 单击**查看全部版本**，在**全部版本**对话框中，查看、回滚或下载插件的版本，单击**确认**：
     - **查看**：单击后，进入新增数据质量插件的基本信息页面。
     - **回滚**：单击后，在**回滚**对话框中，单击**确认**，即可回滚至上个版本。
     - **下载**：单击**下载**，即可下载当前版本的插件。

- 单击**删除**，在**删除插件**对话框中，单击**确认**。

6.12.2. 创建Hologres开发节点

您可以新建、编辑Hologres开发节点，并更新节点版本。

背景信息

Hologres开发节点用于接入阿里云产品交互式分析，详情请参见什么是实时数仓Hologres。

自定义Hologres节点用于将HoloStudio中的开发节点加载到DataStudio进行离线调度配置。

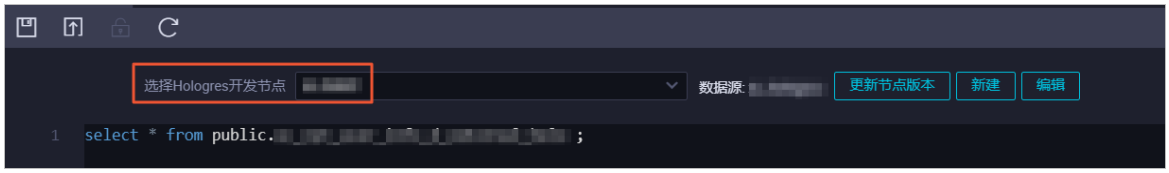
操作步骤

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. 鼠标悬停至**+新建**图标，单击**自定义 > Hologres开发**。

   您也可以打开相应的业务流程，右键单击**自定义**，选择**新建 > Hologres开发**。
3. 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。

4. 单击**提交**。
5. 在**Hologres开发**编辑页面**选择Hologres开发节点**。

   如果没有可以选择的Hologres开发节点，请单击右侧的**新建**。您也可以单击**编辑**，对已有的节点内容进行修改。

   > **说明** 一个Hologres开发节点对应一个DataStudio节点，如果当前选择的Hologres节点已有对应的DataStudio节点，此处重复选择会报错已存在同一实验的节点。

6. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性，详情请参见配置基础属性。
7. 保存并提交节点。

   > **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标，保存节点。
   ii. 单击工具栏中的图标。
   iii. 在**提交新版本**对话框中，输入**变更描述**。

iv. 单击**确认**。

如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

8. 测试节点，详情请参见查看并管理周期任务。

6.12.3. 创建Data Lake Analytics节点

您可以在DataWorks中新建Data Lake Analytics节点，构建在线ETL数据处理流程。

背景信息

Data Lake Analytics节点用于接入阿里云产品Data Lake Analytics，详情请参见什么是Data Lake Analytics。

> **注意** Data Lake Analytics节点仅支持使用独享调度资源组执行任务，详情请参见新增和使用独享调度资源组。

操作步骤

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. 鼠标悬停至+新建图标，单击**自定义 > Data Lake Analytics**。

   您也可以打开相应的业务流程，右键单击**自定义**，选择**新建 > Data Lake Analytics**。
3. 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。
4. 单击**提交**。
5. 编辑Data Lake Analytics节点。
   i. 选择数据源。

      选择任务要执行的目标数据源。如果下拉列表中没有需要的数据源，请单击右侧的**新建数据源**，在**数据源管理**页面新建，详情请参见配置Data Lake Analytics（DLA）数据源。
   ii. 编辑SQL语句。

      选择相应的数据源后，即可根据Data Lake Analytics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。
   iii. 单击工具栏中的图标。
   iv. 单击工具栏中的图标，执行SQL语句。

   如果您需要修改在**数据开发**页面测试时使用的任务执行资源，请单击工具栏中的图标，选择相应的独享调度资源组。

> **说明** 因为访问专有网络环境的数据源需要使用独享调度资源组执行任务，所以此处必须选择测试连通性成功的独享调度资源组。

6. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性，详情请参见配置基础属性。

   配置资源属性时，请选择调度资源组为已经和Data Lake Analytics网络连通的独享调度资源组，作为周期调度时使用的资源组。

7. 保存并提交节点。

   > **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标，保存节点。

   ii. 单击工具栏中的图标。

   iii. 在**提交新版本**对话框中，输入**变更描述**。

   iv. 单击**确认**。

   如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

8. 测试节点，详情请参见查看并管理周期任务。

6.12.4. 创建AnalyticDB for MySQL节点

您可以在DataWorks中新建AnalyticDB for MySQL节点，构建在线ETL数据处理流程。

背景信息

AnalyticDB for MySQL节点用于接入阿里云产品分析型数据库MySQL版，详情请参见分析型数据库MySQL版。

> **说明** AnalyticDB for MySQL节点仅支持使用独享调度资源组，详情请参见新增和使用独享调度资源组。

操作步骤

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. 鼠标悬停至 +新建 图标，单击**自定义 > AnalyticDB for MySQL**。

   您也可以打开相应的业务流程，右键单击**自定义**，选择**新建 > AnalyticDB for MySQL**。

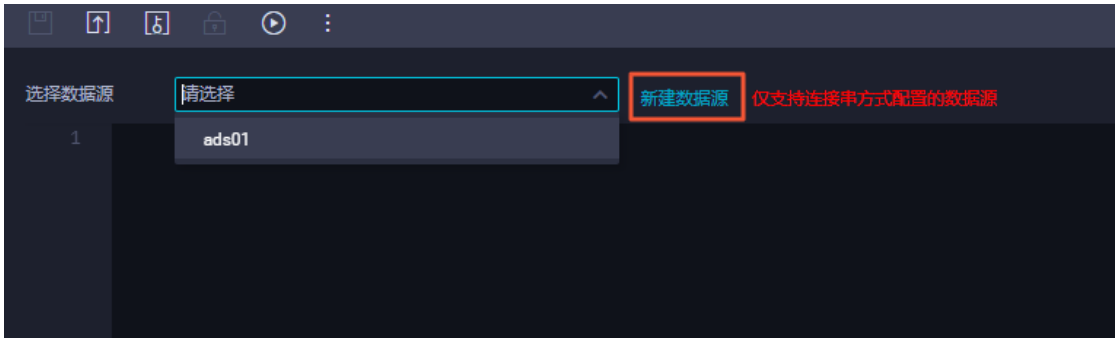
3. 在**新建节点**对话框中，输入**节点名称**，并选择**目标文件夹**。

   > **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。

4. 单击**提交**。

5. 编辑AnalyticDB for MySQL节点。
   i. 选择数据源。

      选择任务要执行的目标数据源。如果下拉列表中没有需要的数据源，请单击右侧的**新建数据源**，在**数据源管理**页面新建，详情请参见支持的数据源与读写插件。

   ii. 编辑SQL语句。

      选择相应的数据源后，即可根据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。

   iii. 单击工具栏中的图标，将其保存至服务器。

   iv. 单击工具栏中的图标，执行编辑的SQL语句。

      第一次运行该节点时，您需要在**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，单击**确定**。

      下一次运行会自动使用第一次选择的资源组和变量的赋值，如果您需要修改资源组或变量的赋值，请单击工具栏中的图标，使用**高级运行**功能。

      > **说明** 因为访问专有网络环境的数据源需要使用独享调度资源组执行任务，所以此处必须选择测试连通性成功的独享调度资源组。

6. 单击节点编辑区域右侧的**调度配置**，配置节点的调度属性，详情请参见配置基础属性。

   配置资源属性时，请选择调度资源组为已经和AnalyticDB for MySQL网络连通的独享调度资源组，作为周期调度时使用的资源组。

7. 保存并提交节点。

   > **注意** 您需要设置节点的**重跑属性**和**依赖的上游节点**，才可以提交节点。

   i. 单击工具栏中的图标，保存节点。

   ii. 单击工具栏中的图标。

   iii. 在**提交新版本**对话框中，输入**变更描述**。

   iv. 单击**确认**。

   如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的**发布**。具体操作请参见发布任务。

8. 测试节点，详情请参见查看并管理周期任务。

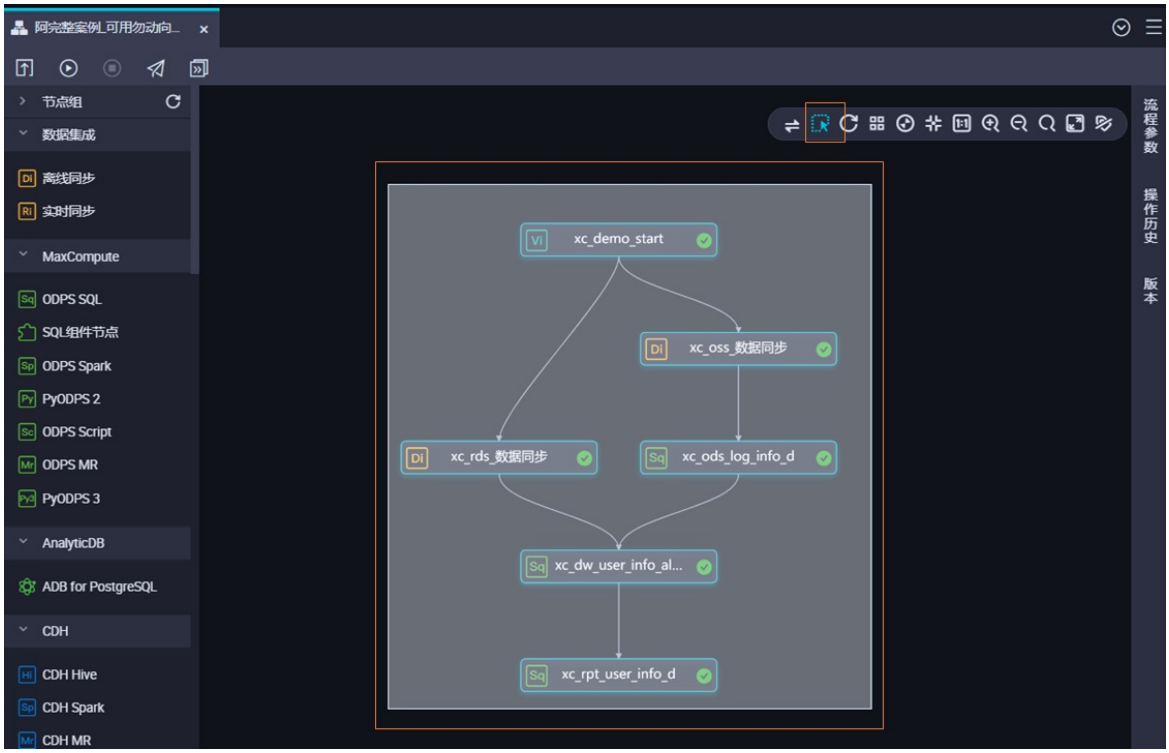
6.13. 节点管理

6.13.1. 节点组

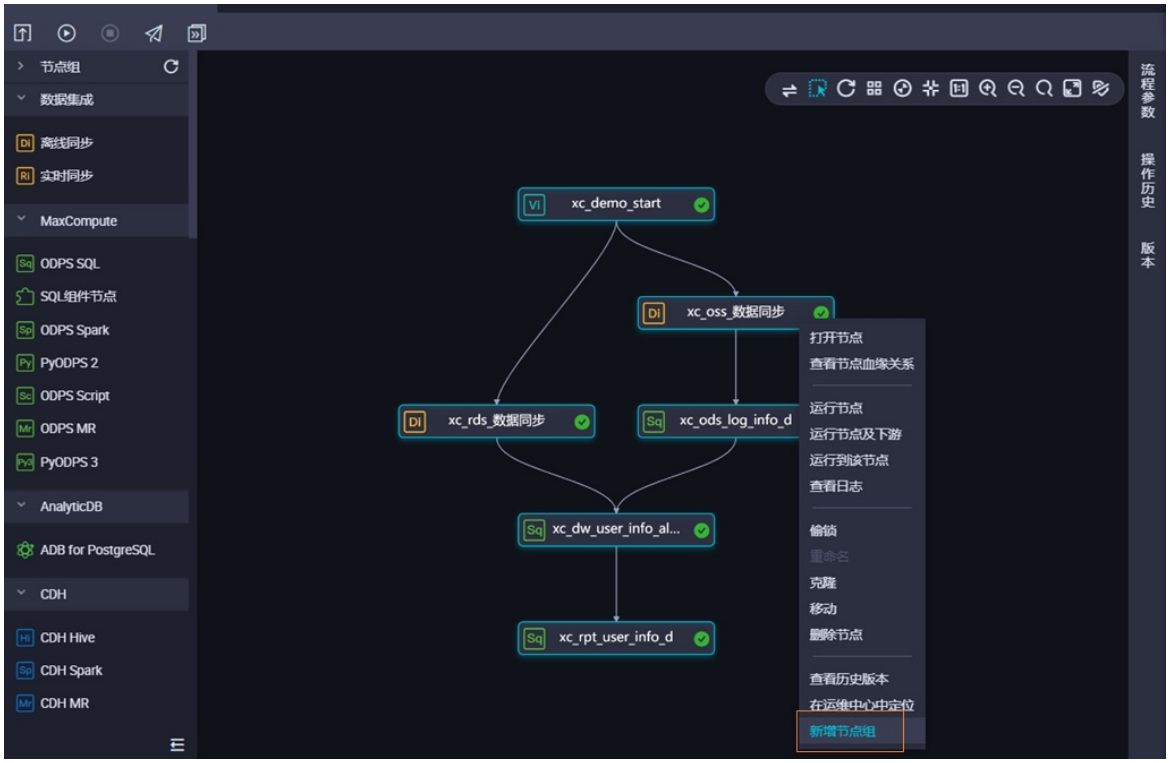
您可以通过节点组的功能，将业务流程内复用率较高的节点组合为一个节点组，以便在其他业务流程中快速复用该节点组（快速克隆这批节点），节点组中各节点的配置信息保持不变，本文将为您介绍如何新建和引用节点组。

新建节点组

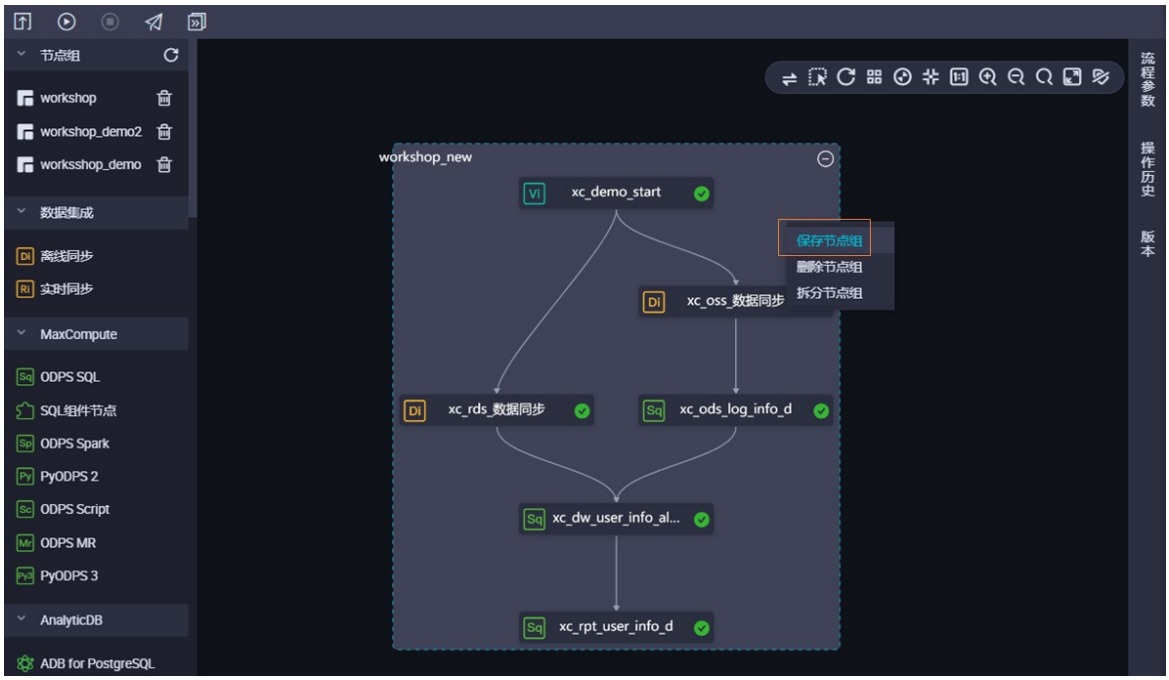
1. 进入**数据开发**页面，新建业务流程。详情请参见管理业务流程。
2. 进入业务流程看板，单击右上角的框选图标，选中节点组中需要包含的节点。



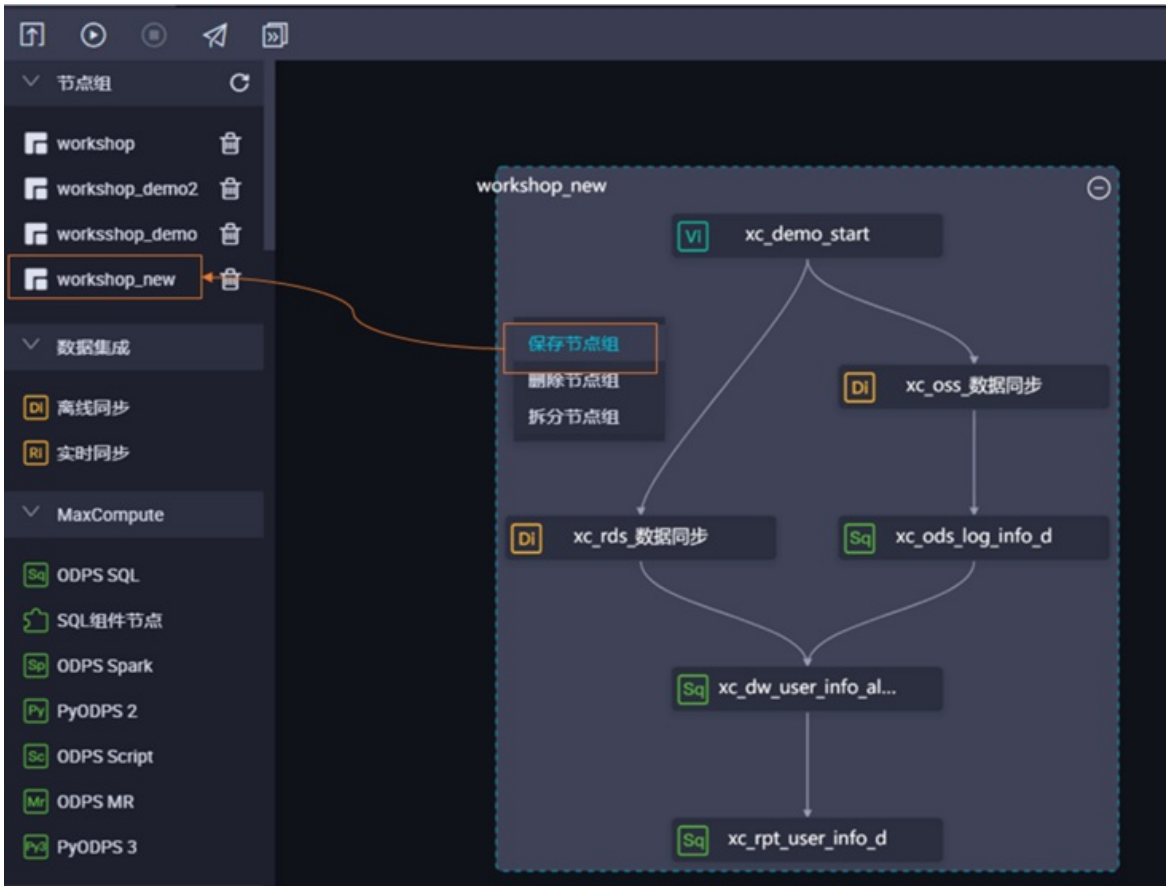
3. 右键单击节点组中的任意节点，选择**新增节点组**。



4. 在**新建节点组**对话框中，填写**名称**，单击**确认**。

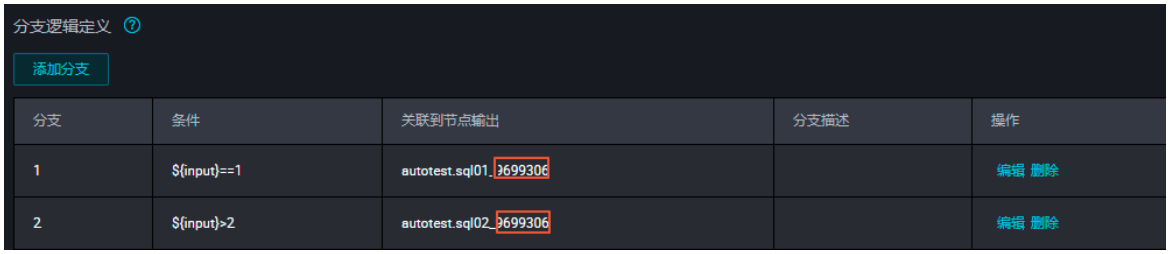


5. 新建完成后，右键单击节点组，选择**保存节点组**，即可在节点组下拉框看到相应的内容。



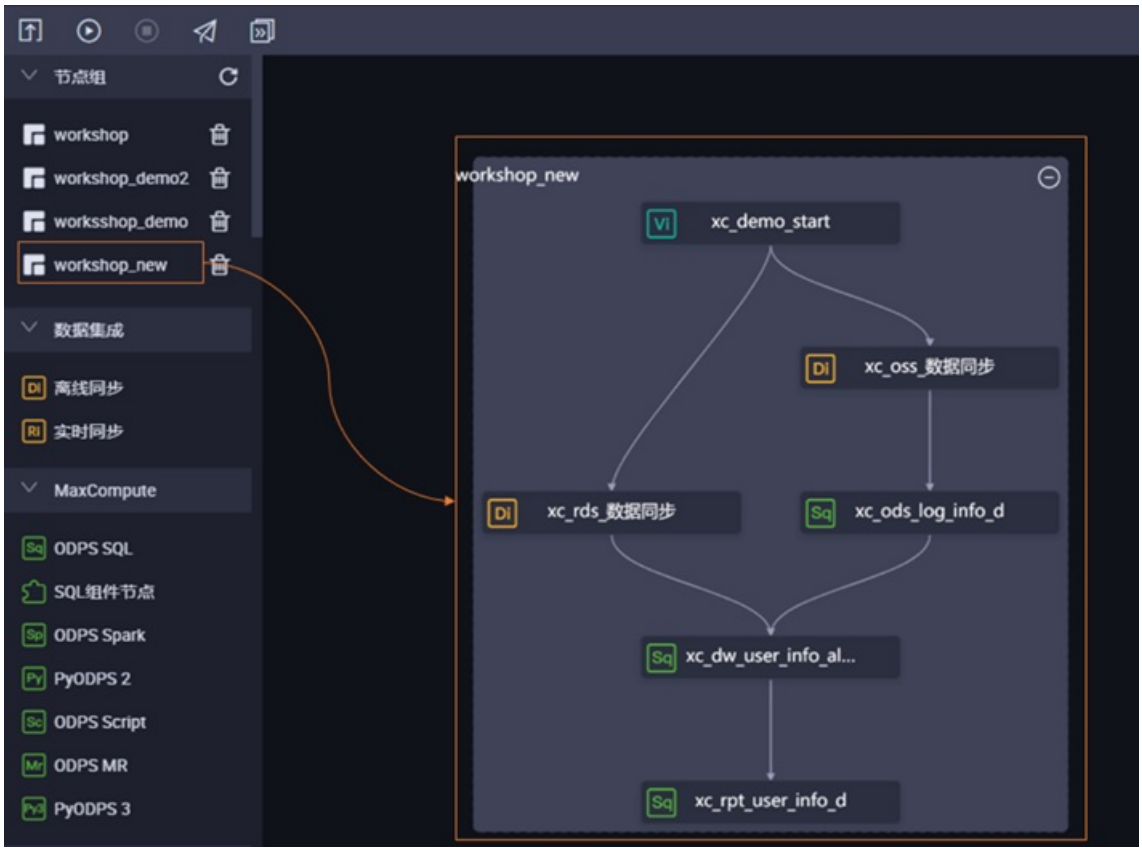
操作	描述
保存节点组	单击**保存节点组**，才能在节点组列表中展现。如果不保存，则不能在其它业务流程中进行引用。
删除节点组	您可以单击**删除节点组**。 说明　此操作将直接删除选择的节点。
拆分节点组	拆分节点组仅拆分您当前选择的节点，拆分后您可以重新选择节点。已经保存在节点组列表的节点组不受影响。

如果创建节点组中包含PAI节点，请在其它业务流程中重新创建实验。如果是分支节点，请在**关联到节点输出**加上数字。



引用节点组

直接将节点组拖入业务流程中，即可在其他业务流程引用该节点组，节点组中的依赖关系保持不变。



您可以直接运行或提交后发布业务流程，进入**运维中心**页面查看相关的运行结果。

6.13.2. 回收站

DataWorks拥有自己的回收站，用于存放当前工作空间下所有删除的节点，您可以对节点进行恢复或彻底删除。

单击左侧导航栏中的**回收站**，即可进入回收站页面。

您可以在回收站中查看当前工作空间下所有删除的节点，右键单击选中的节点，可以选择还原此节点或者彻底删除此节点。

> 说明
> - 回收站仅显示100个节点，如果多于100个，只能彻底删除展示靠前的节点。
> - 组合节点被删除后不会显示在回收站中。

单击右上角的**我的文件**，即可查看该工作空间被删除的节点。

> 说明　如果在回收站彻底删除此节点，将无法恢复。回收站目前只能针对节点任务生效，不能回收针对业务流程、表、资源等。

6.13.3. 组件管理

6.13.3.1. 创建组件

本文为您介绍组件的定义、构成以及如何创建组件。

使用限制

仅DataWorks标准版及以上版本支持创建组件。您需要提前购买DataWorks标准版及以上版本，购买详情请参见：版本服务计费说明。

组件的定义

组件是一种带有多个输入参数和输出参数的SQL代码过程模板，SQL代码过程的处理过程通常是引入1到多个源数据表，通过过滤、连接和聚合等操作，加工出新的业务需要的目标表。

组件的价值

在业务实践中，有大量的SQL代码过程很类似，过程中输入表和输出表的结构一致，或是类型兼容而名称不同。此时，组件的开发者可以抽象SQL过程为一个SQL组件节点，抽象可变的输入表为输入参数，输出表为输出参数，即可实现SQL代码的复用。

在使用SQL组件节点时，您只需要从组件列表中选择和自己业务处理过程类似的组件，为这些组件配置上自己业务中特定的输入表和输出表，不用再重复复制代码，就可以直接生成新的组件SQL节点。从而极大地提升开发效率，避免重复开发。SQL组件节点生成后的发布与调度的操作方法都和普通的SQL节点一样。

组件的构成

一个组件就和一个函数的定义一样，由输入参数、输出参数和组件代码过程构成。

组件的输入参数

组件的输入参数具有参数名、参数类型、参数描述和参数定义等属性，参数类型分为表（table）和字符串（string）类型：

- 表类型的参数：指定组件过程中要引用到的表，在使用组件时，组件的使用者可以为该参数填入其特定业务需要的表。
- 字符串类型的参数：指定组件过程中需要变化的控制参数。例如，指定过程的结果表只输出每个区域的头N个城市的销售额，您可以通过字符串类型的参数控制N的值。

  例如，指定过程的结果表要输出哪个省份的销售总额。您可以设置一个省份字符串参数，指定不同的省份，即可获得指定省份的销售数据。
- 参数描述：描述该参数在组件过程中发挥的作用。
- 参数定义：是表结构的一个文本定义，仅表类型的参数需要。指定组件的使用者需要为该参数提供的和该表参数定义的名字一致并且类型兼容的输入表，组件过程才会正确运行。否则，组件的过程运行时，会因为找不到输入表中指定的字段名而报错。该输入表必须具有该表参数定义中指定的字段名和类型，顺序不限，不禁止多余字段。参数定义仅供您参考。
- 建议定义表参数的格式如下。

  ```
  字段1名 字段1类型 字段1注释
  字段2名 字段2类型 字段2注释
  字段n名 字段n类型 字段n注释
  ```

  示例如下。

  ```
  area_id string '区域id'
  city_id string '城市id'
  order_amt double '订单金额'
  ```

组件的输出参数

- 组件的输出参数具有参数名、参数类型、参数描述和参数定义等属性，参数类型只有表类型（table），字符串类型的输出参数没有逻辑意义。
- 表类型的参数：指定组件过程中最终产出的表。使用组件的，组件的使用者可以为该参数填入其特定业务下通过该组件过程要产出的结果表。
- 参数描述：描述该参数在组件过程中发挥的作用。
- 参数定义：是表结构的一个文本定义。组件的使用者需要为该参数的和该表参数定义的数目一致、并且类型兼容的输出表，组件过程才会正确运行。否则，运行的时候会因为字段个数不匹配或类型不兼容而报错。对于输出表的字段名，不要求和表参数定义的字段名必须一致。参数定义仅供您参考。
- 建议定义表参数的格式如下所示。

```
字段1名 字段1类型 字段1注释
字段2名 字段2类型 字段2注释
字段n名 字段n类型 字段n注释
```

示例如下。

```
area_id string '区域id'
city_id string '城市id'
order_amt double '订单金额'
rank bigint '排名'
```

组件的过程体

在过程体中参数的引用格式为：`@@{参数名}`。

过程体通过编写抽象的SQL加工过程，将指定的输入表按照输入参数进行控制加工出有业务价值的输出表。

组件过程的开发具有一定的技巧，组件过程的代码需要巧妙的利用输入参数和输出参数，使得组件过程能够在使用时填入不同的输入参数和输出参数，也能生成正确的可运行的SQL代码。

创建组件

1. 登录DataWorks控制台，单击相应工作空间后的**进入数据开发**。
2. 单击左侧导航栏中**组件管理**。
3. 鼠标悬停至**+新建**，单击**新建 > 组件**。
4. 在**新建组件**对话框中，输入**组件名称**和**描述**，并选择**目标文件夹**。
5. 配置完成后，单击**提交**。

原始表结构定义

销售数据的原始MySQL结构定义如下。

字段名称	字段类型	字段描述
order_id	varchar	订单编号
report_date	datetime	订单日期
customer_name	varchar	客户名称
order_level	varchar	订单等级
order_number	double	订单数量
order_amt	double	订单金额
back_point	double	折扣点
shipping_type	varchar	运输方式
profit_amt	double	利润金额
price	double	单价

字段名称	字段类型	字段描述
shipping_cost	double	运输成本
area	varchar	区域
province	varchar	省份
city	varchar	城市
product_type	varchar	产品类型
product_sub_type	varchar	产品小类
product_name	varchar	产品名称
product_box	varchar	产品包箱
shipping_date	datetime	运输日期

组件的业务含义

组件的名字：get_top_n

通过指定的销售明细数据表作为输入参数（表类型）和取前多少名作为输入参数（字符串），按照城市销售总额的大小作为排名依据。通过该组件过程，组件的使用者可以轻松获取到各个区域下，指定的前多少名的城市排行。

组件的参数定义

输入参数1

- 参数名：myinputtable
- 类型：table

输入参数2

- 参数名：topn
- 类型：string

输出参数3

- 参数名：myoutput
- 类型：table

参数定义：

- area_id string
- city_id string
- order_amt double
- rank bigint

建表语句如下所示。

```
CREATE TABLE IF NOT EXISTS company_sales_top_n
(
area STRING COMMENT '区域',
city STRING COMMENT '城市',
sales_amount DOUBLE COMMENT '销售额',
rank BIGINT COMMENT '排名'
)
COMMENT '公司销售排行榜'
PARTITIONED BY (pt STRING COMMENT '')
LIFECYCLE 365;
```

定义组件过程举例

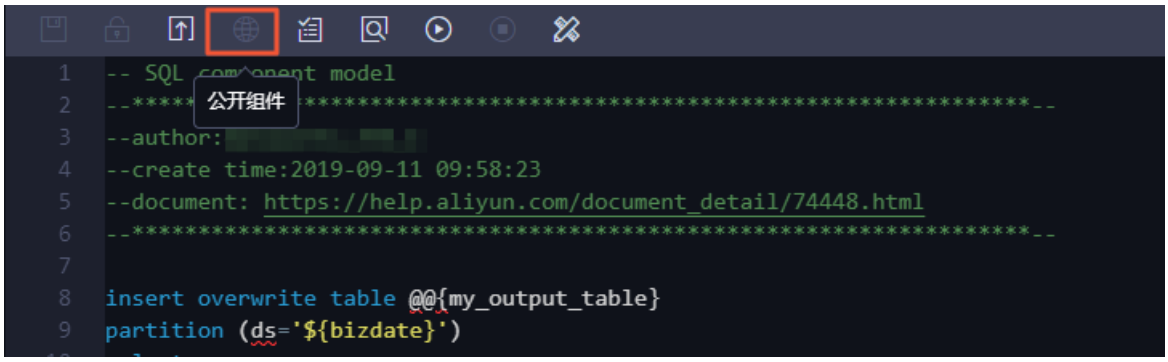
```
INSERT OVERWRITE TABLE @@{myoutput} PARTITION (pt='${bizdate}')
    SELECT r3.area_id,
    r3.city_id,
    r3.order_amt,
    r3.rank
from (
SELECT
    area_id,
    city_id,
    rank,
    order_amt_1505468133993_sum as order_amt ,
    order_number_150546813****_sum,
    profit_amt_15054681****_sum
FROM
    (SELECT
    area_id,
    city_id,
    ROW_NUMBER() OVER (PARTITION BY r1.area_id ORDER BY r1.order_amt_1505468133993_sum DESC
)
AS rank,
    order_amt_15054681****_sum,
    order_number_15054681****sum,
    profit_amt_1505468****_sum
FROM
    (SELECT area AS area_id,
     city AS city_id,
     SUM(order_amt) AS order_amt_1505468****_sum,
     SUM(order_number) AS order_number_15054681****_sum,
     SUM(profit_amt) AS profit_amt_1505468****_sum
FROM
    @@{myinputtable}
WHERE
    SUBSTR(pt, 1, 8) IN ( '${bizdate}' )
GROUP BY
    area,
    city )
    r1 ) r2
WHERE
    r2.rank >= 1 AND r2.rank <= @@{topn}
ORDER BY
    area_id,
    rank limit 10000) r3;
```

组件的分享范围

组件的分享范围包括**组件**和**公共组件**。

组件发布后，默认在本工作空间内的其它用户可见且可用。组件的开发者通过单击**公开组件**，即可将具有全局通用性的组件发布到整个租户内，所有租户内的用户都能看到该公共组件并可以进行使用。

您可以通过查看下图中的**公开组件**图标是否可编辑，确认组件是否为公开组件。

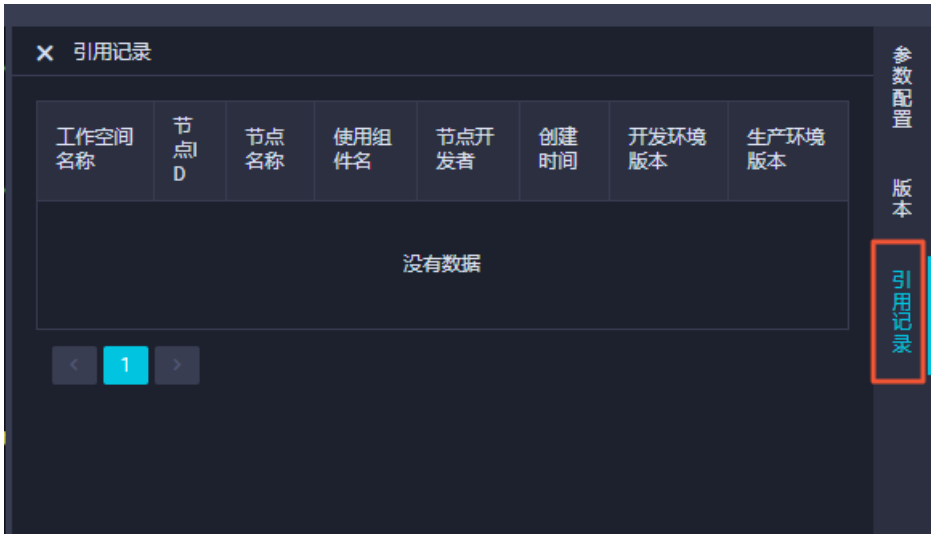


使用组件

组件开发完成后的使用方法请参见使用组件。

组件的引用记录

打开相应的组件，单击右侧的**引用记录**，组件的开发者可以查看该组件的引用记录。



6.13.3.2. 使用组件

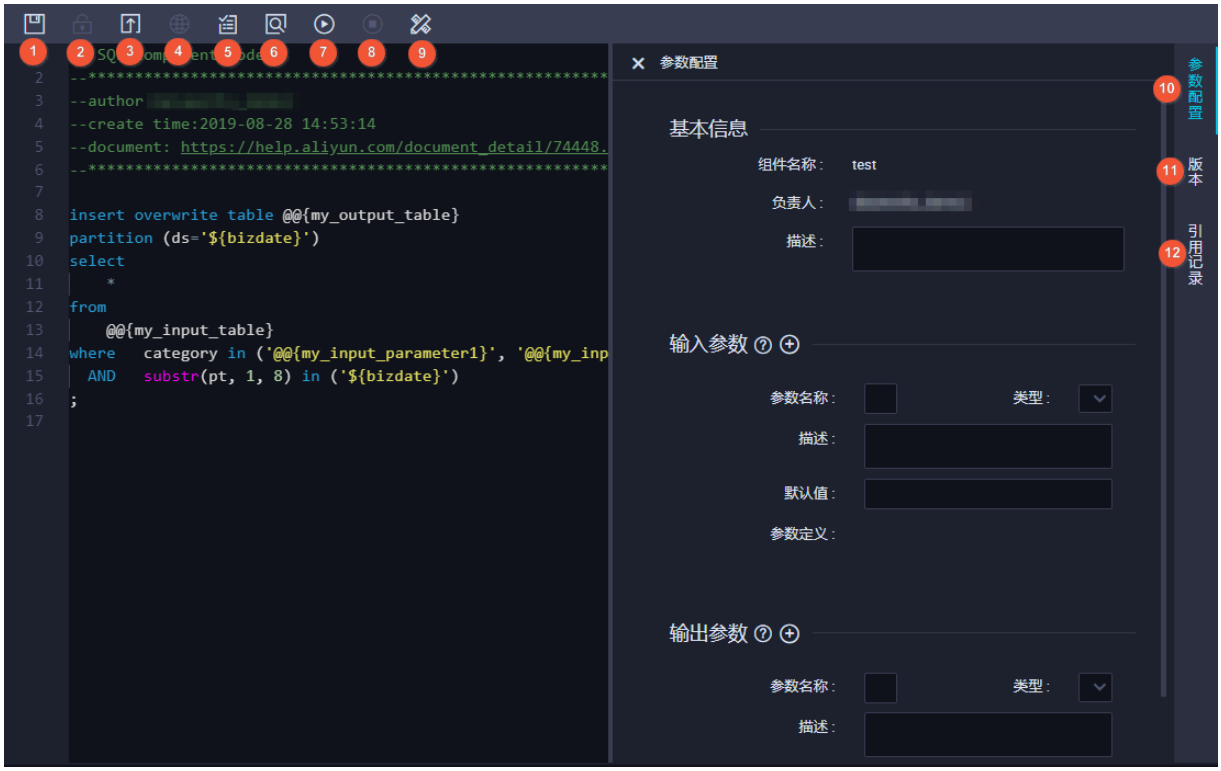
为提高开发效率，数据任务的开发者可以使用工作空间成员和租户成员贡献的组件，来新建数据处理节点。

使用组件的注意事项：

- 本工作空间成员创建的组件在**组件**下。
- 租户成员创建的组件在**公共组件**下。

组件的具体使用方法请参见创建SQL组件节点。

界面功能介绍



序号	功能	说明
1	**保存**	保存当前组件的设置。
2	**偷锁编辑**	非组件责任人可以偷锁编辑此节点。
3	**提交**	将当前组件提交到开发环境。
4	**公开组件**	将具有全局通用性的组件发布到整个租户内，所有租户内的用户都能看到该公共组件并可使用。
5	**输入输出参数解析**	解析当前代码的输入输出参数。 **说明** 此处填写的参数通常为表名称，而非调度参数。
6	**预编译**	对当前组件的自定义参数、组件参数进行编辑。
7	**运行**	在本地（开发环境）运行组件。
8	**停止运行**	停止运行的组件。
9	**格式化**	对当前组件代码根据关键字格式排列。
10	**参数配置**	组件信息、输入参数、输出参数配置。
11	**版本**	组件提交发布的记录。
12	**引用记录**	汇总组件被引用的记录。

6.13.4. 删除节点常见问题

本文为您介绍删除节点的常见问题。

> **注意** 下线节点为高危操作，请谨慎操作。

- 如何删除节点
- 删除节点时，提示：节点存在子节点，下线失败
- 如何确认节点是否成功删除？
- 节点误删后如何找回

如何删除节点

> **注意**
> - 如果您的DataWorks为标准模式，则开发环境和生产环境分离，在DataWorks的DataStudio删除节点时，只删除了开发环境的节点，生产环境的节点需要将删除操作发布到生产环境，生产环境的节点才会被删除下线。
> - 当节点与其他节点存在依赖关系，被其他节点依赖时，节点无法直接删除，需要先解除依赖关系才能删除节点，详情可参见删除节点时，提示：节点存在子节点，下线失败。

以下以删除生产环境的节点为例，为您演示删除节点的操作步骤。

1. 在开发环境中删除节点。

   进入数据开发DataStudio页面后，右键待删除的节点后单击**删除**，在弹出的页面中单击**确认**。

2. 在生产环境中删除节点。

   在数据开发DataStudio页面的右上角单击**任务发布**，过滤变更类型为下线，找到上述步骤下线节点的变更发布包后，单击**操作**列的**发布**，在弹出的页面中单击**发布**。

   完成发布后，生产环境的节点才会被删除。

删除节点时，提示：节点存在子节点，下线失败

当节点与其他节点存在依赖关系，被其他节点依赖时，节点无法直接删除，您可以进入数据开发DataStudio页面或运维中心页面后，找到依赖待删除节点的子节点，修改子节点的依赖配置，解除子节点与待删除节点的依赖关系，解除后再删除待删除节点。

> 说明
>
> - 如果您使用的DataWorks为标准模式，开发环境和生成环境隔离，您在开发环境解除节点的依赖关系后，需提交发布至生产环境，提交发布完成后生产环境的节点才能被删除。
> - 跨周期依赖也是依赖关系的一种，解除依赖关系时，除了**调度配置**的**调度依赖**外，您还需关注**时间属性**中的**依赖上一周期**的配置情况，如果节点开启了**依赖上一周期**，您需要同时关闭**依赖上一周期**的配置。

如何确认节点是否成功删除？

删除节点后，您可以进入运维中心的周期任务页面，通过节点ID查询已删除的节点，如果查询不到此节点，说明节点已成功删除。

节点误删后如何找回

节点删除后会放在回收站，如果需要找回误删的节点，您可以到回收站去还原代码，回收站的操作可参见回收站。

7.创建并管理资源及函数

7.1. 创建资源及注册函数

7.1.1. 创建MaxCompute资源

本文为您介绍如何创建JAR和Python类型的资源，以及如何引用和下载资源。

前提条件

您在**工作空间配置**页面添加MaxCompute引擎后，当前页面才会显示MaxCompute目录。详情请参见配置工作空间。

背景信息

如果您的代码或函数中需要使用.jar等资源文件，您可以先上传资源至该工作空间，再进行引用。

如果现有的系统内置函数无法满足您的需求，DataWorks支持创建自定义函数，实现个性化处理逻辑。将实现逻辑的JAR包上传至工作空间下，便可以在创建自定义函数时进行引用。

> **说明**
> - 您可以在**函数列表**面板查看系统内置的函数，详情请参见查看函数列表。
> - 您可以在**MaxCompute函数**面板查看在DataWorks提交或发布的函数，详情请参见MaxCompute函数。

您可以将文本文件、Python代码以及.zip、.tgz、.tar.gz、.tar、.jar等压缩包作为不同类型的资源上传至MaxCompute，在UDF及MapReduce的运行过程中读取、使用资源。

MaxCompute为您提供读取、使用资源的接口。目前资源包括以下类型：

- **Python**：您编写的Python代码，用于注册Python UDF函数。
- **JAR**：编译好的Java JAR包。
- **Archive**：通过资源名称中的后缀识别压缩类型，支持的压缩文件类型包括.zip、.tgz、.tar.gz、.tar和.jar。
- **File**：仅支持.zip、.so和.jar类型的File资源。

JAR和**File**类型的资源，区别如下：

- JAR资源是您在线下Java环境编辑Java代码，打包为JAR包上传至DataWorks。
- File类型的小文件资源可以直接在DataWorks上编辑。
- 新建File类型资源时，选中**大文件**，可以上传超过500 KB的本地资源文件。

> **说明** 目前支持最大可以上传50 MB资源。超过50 MB的资源，您可以通过MaxCompute客户端上传，并使用MaxCompute资源提交至DataWorks。详情请参见MaxCompute资源。

创建JAR资源

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. 鼠标悬停至**+新建**图标，单击**MaxCompute > 资源 > JAR**。

   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 > JAR**。

   如果您需要创建业务流程，请参见创建业务流程。

3. 在**新建资源**对话框中，输入**资源名称**，并选择**目标文件夹**。

   > **说明**
   > - 新创建的JAR资源如果未在MaxCompute（ODPS）客户端上传过，则需要勾选**上传为ODPS资源**，否则上传会报错。如果该JAR资源已经在MaxCompute（ODPS）客户端上传过，则需要取消勾选**上传为ODPS资源**，否则上传会报错。
   > - 资源名称无需与上传的文件名保持一致。
   > - 资源名称命名规范：1~128个字符，字母、数字、下划线、小数点，大小写不敏感，JAR资源的后缀为.jar，Python资源的后缀为.py。

4. 单击**点击上传**，选择相应的文件进行上传。
5. 单击**确定**。
6. 单击工具栏中的图标，提交资源至调度开发服务器端。

   如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的发布。具体操作请参见发布任务

创建Python资源并注册函数

1. 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**，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. 在您新建的Python资源内编写Python资源代码，示例如下。

```
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
```

v. 单击工具栏中的图标，提交节点。

如果您使用的是标准模式的工作空间，提交成功后，请单击右上方的发布。具体操作请参见发布任务。

2. 注册函数。

i. 在DataStudio（数据开发）页面，鼠标悬停至+新建图标，单击MaxCompute > 函数。

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

ii. 在新建函数对话框中，输入函数名称，并选择目标文件夹。

iii. 单击提交。

iv. 在注册函数对话框中，输入函数的类名（示例为 `ipint.ipint` ），在资源列表输入提交的资源名称，单击工具栏中的图标。

v. 验证ipint函数是否生效并满足预期值。您可以在DataWorks上新建一个ODPS SQL类型节点运行SQL语句查询。

您也可以在本地创建ipint.py文件，使用MaxCompute客户端上传资源，详情请参见MaxCompute客户端。

```
odps@ MaxCompute_DOC>add py D:/ipint.py;
OK: Resource 'ipint.py' have been created.
```

```
odps@ MaxCompute_DOC>create function ipint as ipint.ipint using ipint.py;
Success: Function 'ipint' have been created.
```

完成上传后，使用客户端直接注册函数，详情请参见注册函数。完成注册后，即可正常使用该函数。

引用和下载资源

- 在函数中引用资源请参见注册函数。
- 在节点中引用资源请参见创建ODPS MR节点。

如果您需要下载资源，请双击资源选择您需要的资源，单击下载。通过MaxCompute客户端下载资源的详情请参见资源操作。

其他操作

资源创建成功后，您还可以在对应业务流程下，选择MaxCompute > 资源，右键单击对应资源，进行资源的重命名、引用及删除等操作。删除资源的具体操作请参见数据开发与运行。

7.1.2. 创建和使用EMR资源

DataWorks支持可视化创建EMR（E-MapReduce）JAR、EMR（E-MapReduce）FILE资源，用于上传提交自定义函数或开源MR示例源码作为资源，便于EMR 计算节点的数据开发过程中引用。本文为您介绍如何创建资源，并上传提交资源，为资源的使用做好前期准备。

前提条件

创建EMR资源

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. 鼠标悬停至+新建图标，单击**EMR > 资源**。

   您也可以找到相应的业务流程，右键单击**EMR**，选择**资源 > EMR JAR或EMR FILE**。

3. 在**新建资源**对话框中，配置各项参数。

参数	描述
引擎类型	默认新建EMR类型的资源，不可修改。
引擎实例	从下拉列表中选择需要新建资源的目标引擎实例。 **说明** 此处展示工作空间下绑定好的EMR引擎。
目标文件夹	默认当前所在文件夹的路径，您可以进行修改。
存储路径	为该资源选择存储的路径，包括**OSS**和**HDFS**两种存储类型： ○ 如果您选择**OSS**，需要先授权再选择目录的位置。 **说明** 需要主账号在此处进行授权操作。 ○ 如果您选择**HDFS**，需要手动输入存储路径。
文件类型	仅支持EMR JAR、EMR FILE类型的资源。

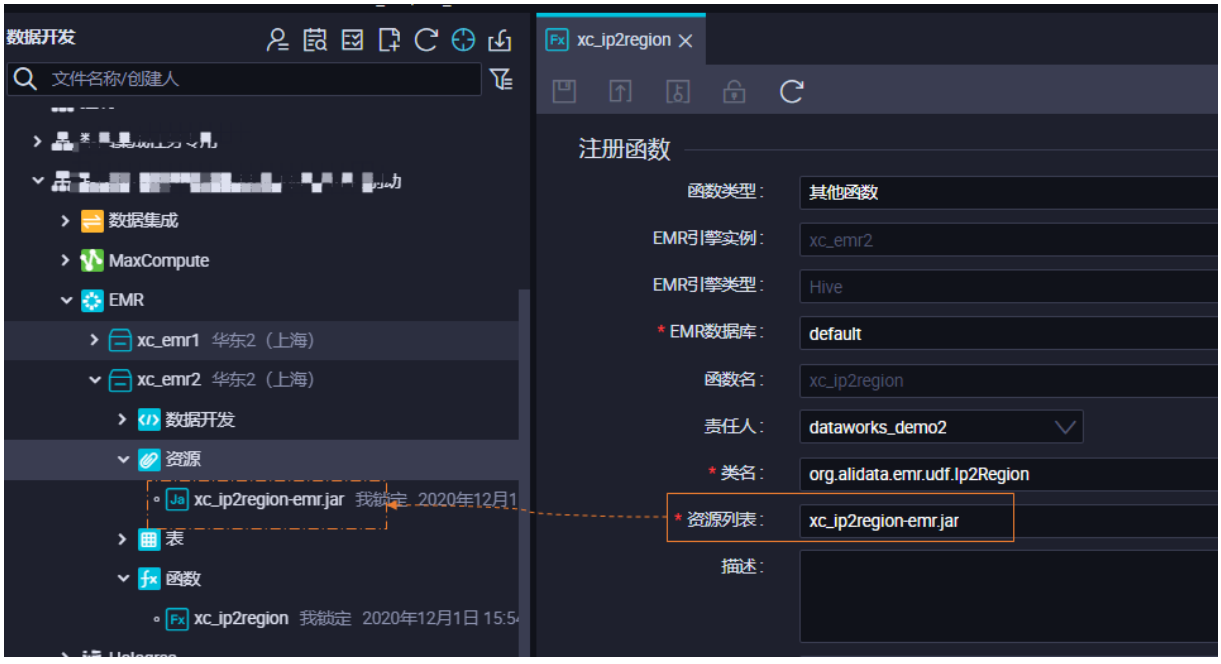
参数	描述
上传文件	单击**点击上传**，在本地选择相应文件后，单击**打开**。
资源名称	新建的EMR资源的名称，如果您上传的是jar资源，您需要添加后缀名.jar。

4. 在**新建资源**对话框中，单击**确定**。
5. 单击工具栏中的和图标，保存并提交资源至调度开发服务器端。

> **说明**
>
> 提交时，您需要选择提交资源所用的调度资源组，当使用独享调度资源组提交表时，DataWorks平台将下发对应新建资源的任务到引擎侧执行，并打印执行过程的执行日志，如果资源提交过程中出现问题，您可以通过日志先自助排查。如果您目前无可用的独享调度资源组，请购买并配置独享调度资源组便于使用，操作详情请参见新增和使用独享调度资源组。

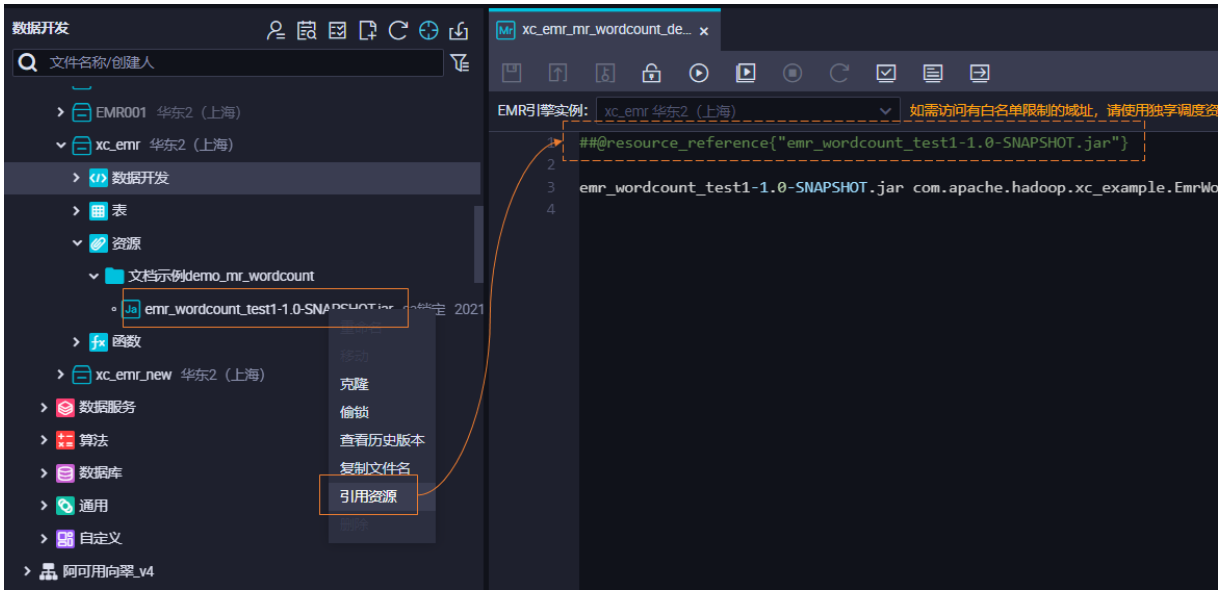
使用资源注册函数

DataWorks支持可视化方式使用资源来注册函数，当您将函数注册所需的资源通过DataWorks可视化上传后，便可通过可视化方式在函数注册时使用该资源。



节点中使用资源

创建完成EMR JAR资源后，如果您需要在节点中直接使用资源，您需要右键**资源**，选择**引用资源**，引用方式如下图所示。

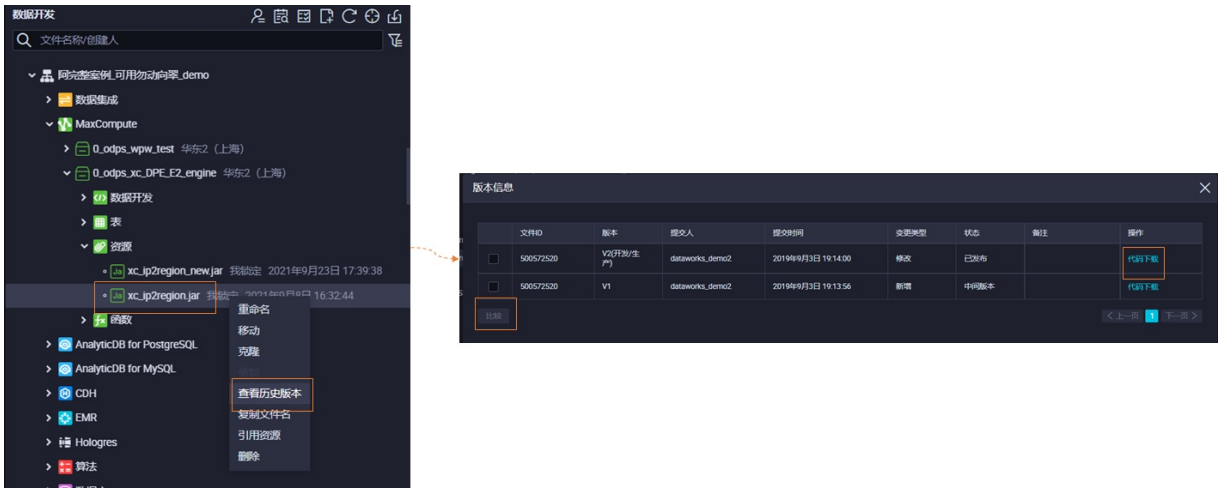


> **说明** 节点中引用资源后，会自动添加一条@resource_reference{"resourcename}，表示节点内已经引用该资源。

详细的引用操作步骤可参见创建并使用EMR MR节点。

资源版本管理

每次提交资源都将生成一个资源版本，您可以通过右键**资源**，**查看历史版本**查看并下载资源。



7.1.3. 注册MaxCompute函数

DataWorks支持Python和Java两种语言接口，本文为您介绍如何注册函数。

前提条件

您需要先上传资源，才可以注册函数。

操作步骤

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. 创建业务流程，详情请参见创建业务流程。
3. 创建JAR或Python类型的资源，并提交发布。详情请参见创建MaxCompute资源。
4. 新建函数。
   i. 打开相应的业务流程，右键单击**MaxCompute**，选择**新建 > 函数**。
   ii. 在**新建函数**对话框中，输入**函数名称**，并选择**目标文件夹**。
   iii. 单击**提交**。
   iv. 在**注册函数**对话框中，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	默认显示为当前登录账号，您也可以选择其他账号。

参数	描述
类名	UDF函数的类名，格式为 `资源名.类名` 。其中，资源名可以为Java包名称或Python资源名称。 DataWorks创建自定义函数时支持使用JAR及Python两种类型的MaxCompute资源，不同类型资源的类名配置如下： ■ 当资源类型为JAR时，配置的**类名**格式为 `Java包名称.实际类名` ，您可以在IDEA中通过 `copy reference` 语句获取。 例如， `com.aliyun.odps.examples.udf` 为Java包的名称， `UDAFExample` 为实际类名，则**类名**参数配置为 `com.aliyun.odps.examples.udf.UDAFExample` 。 ■ 当资源类型为Python时，配置的**类名**格式为 `Python资源名称.实际类名` 。 例如， `LcLognormDist_sh` 为Python资源名称， `LcLognormDist_sh` 为实际类名，则**类名**参数配置为 `LcLognormDist_sh.LcLognormDist_sh` 。 **说明** ■ 填写资源名称时，无需添加.jar或.py后缀。 ■ 资源需要提交并发布后才可以使用，详情请参见创建MaxCompute资源。
资源列表	支持模糊匹配查找本工作空间中已添加的资源，必填。 **说明** ■ 无需填写已添加的资源的路径。 ■ 如果UDF中调用了多个资源，则多个资源使用英文逗号（,）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。
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。

5. 单击工具栏中的图标。
6. 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**变更描述**。
iii. 单击**确认**。

7.1.4. 注册EMR函数

本文为您介绍如何注册EMR（E-MapReduce）函数。

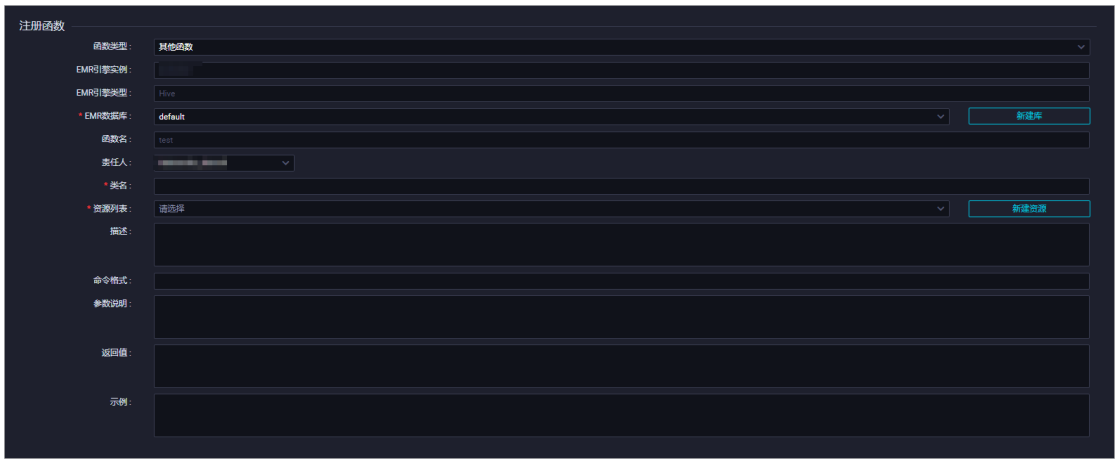
前提条件

-
-
- 您需要先上传资源，才可以注册函数。新建EMR资源详情可参考文档：创建和使用EMR资源

操作步骤

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. 新建业务流程，详情请参见创建业务流程。
3. 在本地Java环境编辑程序并生成JAR包后，再新建JAR资源，并提交发布。详情请参见创建和使用EMR资源。
4. 新建函数。
   i. 打开相应的业务流程，右键单击**EMR**，选择**新建 > 函数**。
   ii. 在**新建函数**对话框中，输入**函数名称**，并选择**EMR引擎实例**和**目标文件夹**。
   iii. 单击**提交**。

iv. 在**注册函数**对话框中，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**。
EMR引擎实例	默认不可以修改。
EMR引擎类型	默认不可以修改。
EMR数据库	从下拉列表中选择相应的数据库。如果您需要新建数据库，请单击**新建库**。在**新建库**对话框中，配置各项参数，单击**确认**。
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。
资源列表	从下拉列表中选择本工作空间中已添加的资源，必填。如果您需要新建资源，请单击**新建资源**。在**新建资源**对话框中，配置各项参数，单击**确定**。
描述	对当前UDF进行简单描述。
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。
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。

5. 单击工具栏中的图标。
6. 提交函数。

i. 单击工具栏中的图标。

> 说明
>
> 提交时，您需要选择提交函数所用的调度资源组，当使用独享调度资源组提交表时，DataWorks平台将下发对应的注册函数的任务到引擎侧执行，并打印执行过程的执行日志，如果资源提交过程中出现问题，您也可以通过日志先进行自助排查。如果您目前无可用的独享调度资源组，请购买并配置独享调度资源组便于使用，操作详情请参见新增和使用独享调度资源组。

ii. 在**提交新版本**对话框中，输入**变更描述**。

iii. 单击**确认**。

7. 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**变更描述**。

iii. 单击**确认**。

7.2. 管理MaxCompute资源和函数

7.2.1. 函数列表

函数列表页面为您展示MaxCompute系统自带的函数，您可以在该页面查看函数的分类、说明和示例。

1. 登录DataWorks控制台，单击相应工作空间后的**进入数据开发**。
2. 单击左侧导航栏中的**函数列表**，即可查看相应的函数。

   函数列表中包括聚合函数、窗口函数、日期函数、数学函数、字符串函数和其他函数。

   上述函数为系统自带的函数，您可以单击相应的函数，查看函数的具体说明。

7.2.2. MaxCompute模块管理

MaxCompute模块包括MaxCompute资源和MaxCompute函数，本文将为您介绍如何添加或移除MaxCompute模块。

背景信息

MaxCompute模块默认隐藏，如果您需要使用该功能，请首先在**设置 > 配置中心**页面进行添加。

操作步骤

1. 登录DataWorks控制台，单击相应工作空间后的**进入数据开发**。
2. 单击左下角的，默认进入**设置 > 配置中心**页面。
3. 单击**模块管理**下的**MaxCompute**，即可查看**已添加模块**和**全部模块**。

4. 单击**全部模块**中未添加的模块，即可成功添加，并在**数据开发**页面的左侧导航栏下显示MaxCompute模块。

   如果您需要移除已添加的模块，单击**已添加模块**中相应模块后的⊖即可。移除后的模块将不会在**数据开发**页面的左侧导航栏显示。

7.2.3. MaxCompute函数

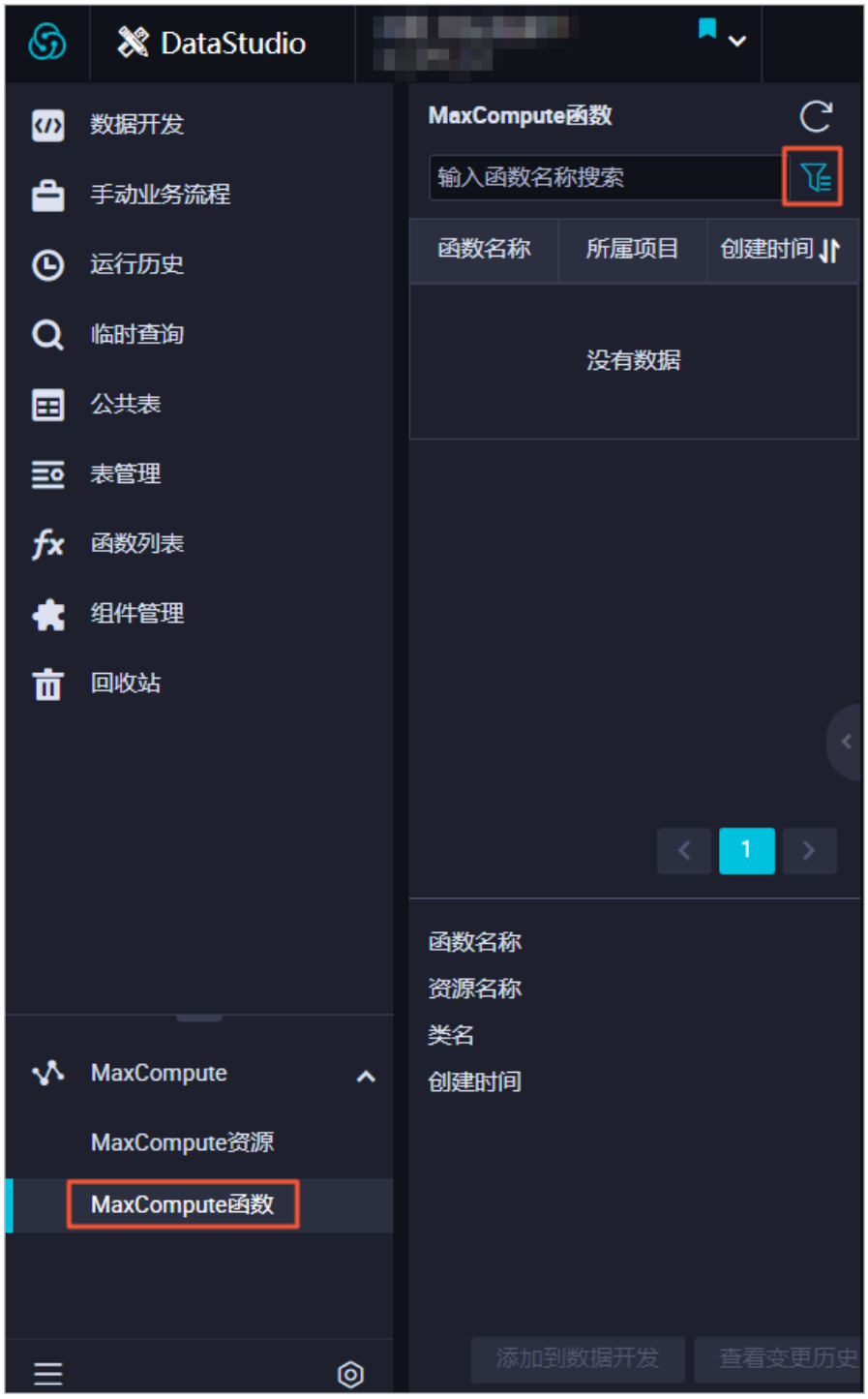
您可以通过MaxCompute函数面板，查看在MaxCompute计算引擎中存在的函数、函数的变更历史，并可以一键添加函数至数据开发面板的业务流程中。

前提条件

MaxCompute函数模块默认隐藏，请在**设置 > 模块管理**页面添加该模块，才可以在**数据开发**页面的左侧导航栏中进行查看，详情请参见MaxCompute模块管理。

查看函数

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. 展开左侧导航栏中的**MaxCompute**，单击**MaxCompute函数**。



MaxCompute函数默认按照创建时间倒序排列，单击图标，可以切换条目排序。

您可以在MaxCompute函数面板查看数据开发页面提交或发布的函数，详情请参见注册MaxCompute函数。

3. 单击某项函数，即可查看其详细信息。

   MaxCompute函数默认显示**生产环境**下的函数。如果您需要查看提交但未发布的函数，请单击图标切换环境。

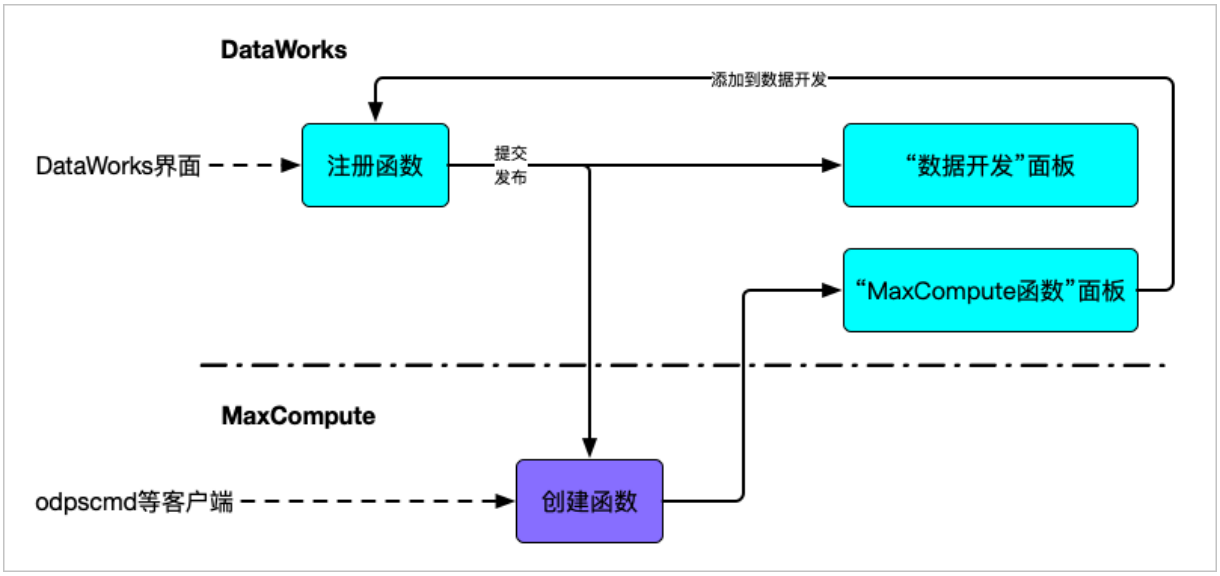
> **说明**
> - 简单模式的工作空间仅支持生产环境。
> - 通过MaxCompute客户端、MaxCompute Studio等非DataWorks方式上传的函数，您可以在MaxCompute函数面板进行查看，但不会显示在**数据开发**中。

删除函数

如果您需要删除函数，请切换至**数据开发**面板，右键单击相应业务流程下的函数名称，单击**删除**。

添加函数至数据开发面板

下图为添加函数至数据开发面板的流程。



1. 在**MaxCompute函数**面板中，单击相应的函数，即可在下方查看该函数的详细信息。
2. 单击**添加到数据开发**。

   您可以通过此操作，快速将**MaxCompute函数**面板中的函数同步至**数据开发**面板的业务流程中。
3. 在**新建函数**对话框中，输入**函数名称**，并选择**目标文件夹**。

   > **说明** 在上传过程中，您可以重命名函数名称、选择目标文件夹（即修改函数所处的业务流程），但不可以修改函数定义。
4. 单击**提交**。

> 说明
> - 创建完成后，您还需要手动完成保存、提交、发布等过程，与在业务流程中的MaxCompute函数一致。
> - 函数提交、发布过程中，会同样上传到开发、生产环境的MaxCompute，也会同时更新在**MaxCompute函数**面板的自定义函数中。
> - 由于函数在MaxCompute项目中的唯一性，如果在同一工作空间中有同名的函数，添加过程将会覆盖原有函数。如果原有函数处于不同的业务流程，将会在新业务流程下完成覆盖。

查看函数的变更历史

单击相应函数下的**查看变更历史**，即可查看该函数的创建、修改记录。

7.2.4. MaxCompute资源

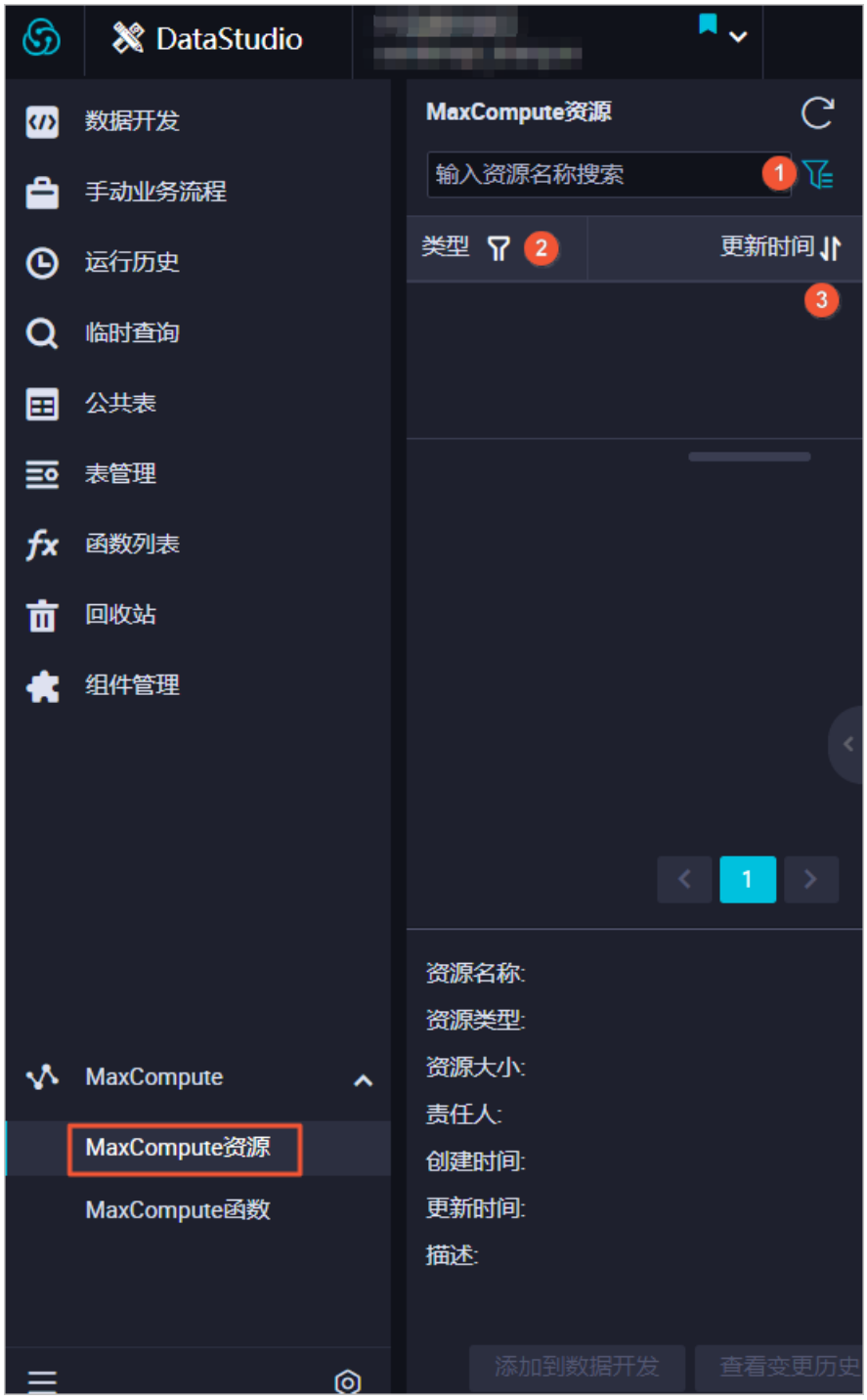
您可以通过MaxCompute资源面板，查看在MaxCompute计算引擎中存在的资源、资源的变更历史，并可以一键添加资源文件至数据开发面板的业务流程中。

前提条件

MaxCompute资源模块默认隐藏，请在**设置 > 模块管理**页面添加该模块，才可以在**数据开发**页面的左侧导航栏中进行查看，详情请参见MaxCompute模块管理。

查看资源

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. 展开左侧导航栏中的**MaxCompute**，单击**MaxCompute资源**。


DataStudio
数据开发
手动业务流程
运行历史
临时查询
公共表
表管理
函数列表
回收站
组件管理
MaxCompute
MaxCompute资源
MaxCompute函数
MaxCompute资源
输入资源名称搜索
1
类型
2
更新时间
3
1
资源名称:
资源类型:
资源大小:
责任人:
创建时间:
更新时间:
描述:
添加到数据开发
查看变更历史

序号	图标	说明

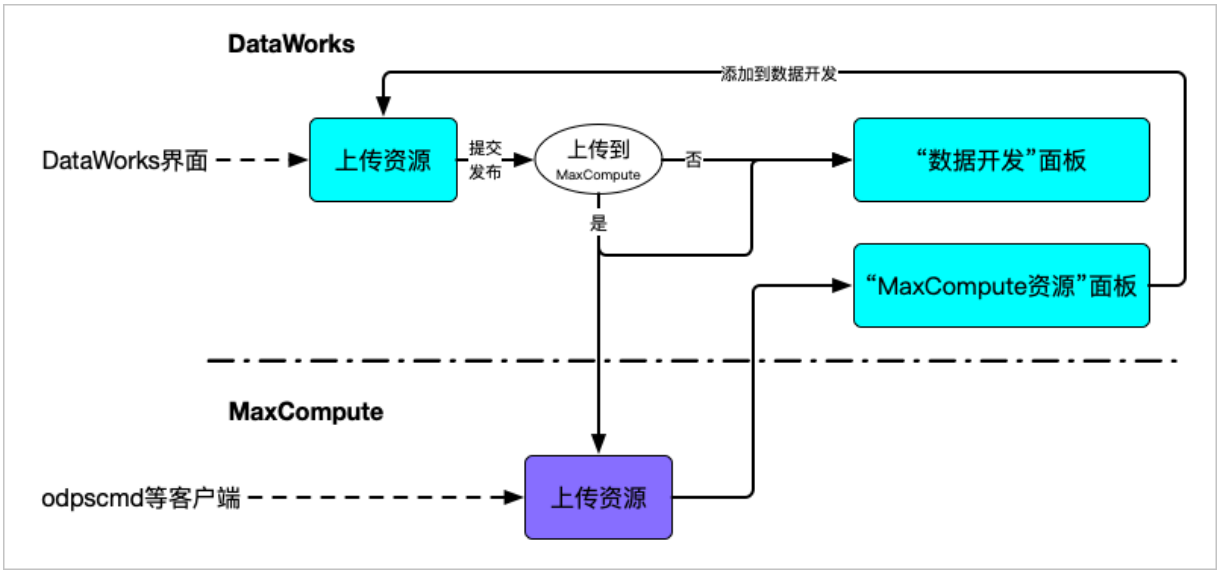
序号	图标	说明
①		MaxCompute资源默认显示**生产环境**下的资源，如果您需要查看提交但未发布的资源，请单击图标切换环境。  **说明** 简单模式的工作空间只有生产环境。
②		单击后，可以筛选不同的资源类型。 **说明** 目前支持Jar、Archive、File和Python等资源类型。
③		单击后，可以切换条目排序，默认按更新时间倒序排列。

3. 单击相应的资源，即可查看**资源名称**、**资源类型**和**资源大小**等详细信息。
   - **MaxCompute资源**面板中所列的资源，并非一定与**数据开发**面板中的资源一致。
     - **数据开发**面板中的资源只有**同时上传到MaxCompute**，并**提交**或**发布**后，才会出现在**MaxCompute资源**面板的开发环境或生产环境中。
     - 通过MaxCompute客户端、MaxCompute Studio等非DataWorks渠道上传的资源文件，不会在**数据开发**面板显示，但会出现在**MaxCompute资源**面板中。
   - 使用**MaxCompute资源**面板中所列的资源时，请注意与**数据开发**面板中资源的区别。

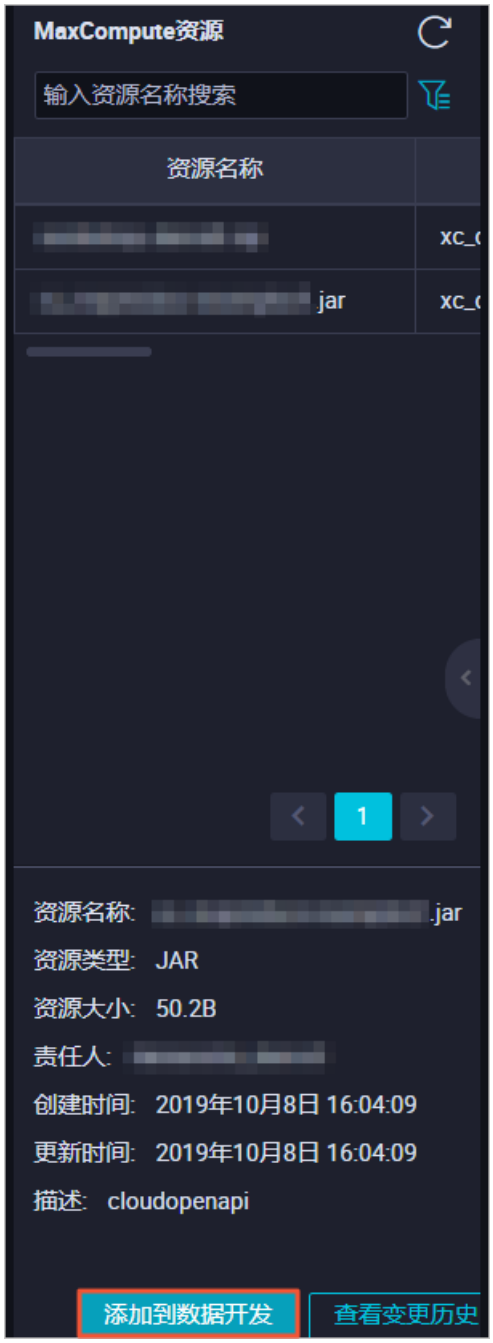
使用场景	数据开发	MaxCompute资源
在ODPS SQL节点中使用	是（需同时上传至MaxCompute）	是
在ODPS MR节点中使用	是（需同时上传至MaxCompute）	否
在Shell节点中使用	是	否
在临时查询中使用	是（需同时上传至MaxCompute）	是
在业务流程中创建函数	是（需同时上传至MaxCompute）	否

添加资源至数据开发面板

下图为添加资源至数据开发面板的流程。



1. 找到需要的资源后，单击**添加到数据开发**。



您可以通过此操作，快速将**MaxCompute资源**面板中的资源文件同步至**数据开发**的业务流程中。

2. 在**新建资源**对话框中，输入**资源名称**，并选择**目标文件夹**，单击**点击上传**，选择相应的文件进行上传。

   上传过程中，您可以进行如下操作：

   - 重命名资源名称。
   - 选择目标文件夹，即修改资源所处的业务流程。

   上传过程中，您不可以进行如下操作：

   - 更改资源类型。
   - 选择是否上传为ODPS资源。
   - 重新上传文件。

3. 配置完成后，单击**确定**，即可完成资源的创建。

> **说明**
>
> - 创建完成后，您需要手动完成保存、提交、发布等操作，与业务流程中对资源进行的操作一致。详情请参见创建MaxCompute资源。
> - 资源提交、发布过程中，会同样上传至开发环境、生产环境的MaxCompute，同时更新在**MaxCompute资源**面板的资源文件中。
> - 由于资源在MaxCompute项目中的唯一性，如果在同一工作空间中有同名资源，添加过程将会覆盖原有资源。如果原有资源处于不同的业务流程，将会在新业务流程下完成覆盖。

查看资源的变更历史

单击相应资源下的**查看变更历史**，即可查看资源文件的创建、修改记录。

8.代码开发与质量保障
8.1. SQL代码编码原则和规范

本文为您介绍SQL编码的基本原则和详细的编码规范。

编码原则

SQL代码的编码原则如下：

- 代码功能完善。
- 代码行清晰、整齐，代码行的整体层次分明、结构化强。
- 代码编写充分考虑执行速度最优的原则。
- 代码中需要添加必要的注释，以增强代码的可读性。
- 规范要求并非强制性约束开发人员的代码编写行为。实际应用中，在不违反常规要求的前提下，允许存在可以理解的偏差。
- SQL代码中应用到的所有SQL关键字、保留字都需使用全大写或小写，例如select/SELECT、from/FROM、where/WHERE、and/AND、or/OR、union/UNION、insert/INSERT、delete/DELETE、group/GROUP、having/HAVING和count/COUNT等。不能使用大小写混合的方式，例如Select或seLECT等方式。
- 4个空格为1个缩进量，所有的缩进均为1个缩进量的整数倍，按照代码层次对齐。
- 禁止使用`select *`操作，所有操作必须明确指定列名。
- 对应的括号要求在同一列的位置上。

SQL编码规范

SQL代码的编码规范如下：

- 代码头部

  代码头部添加主题、功能描述、作者和日期等信息，并预留修改日志及标题栏，以便后续添加修改记录。注意每行不超过80个字符，模板如下。

```
-- MaxCompute(ODPS) SQL
--********************************************************************************
-- ** 所属主题：交易
-- ** 功能描述：交易退款分析
-- ** 创建者 ：有码
-- ** 创建日期：20170616
-- ** 修改日志：
-- ** 修改日期 修改人 修改内容
-- yyyymmdd name comment
-- 20170831 无码 增加对biz_type=1234交易的判断
--********************************************************************************
```

- 字段排列要求
  - SELECT语句选择的字段按照每行1个字段的方式编排。
  - 首个选择的字段与SELECT之间隔1个缩进量。
  - 换行缩进2个缩进量后，添加逗号再输入其它字段名。
  - 2个字段之间的逗号分隔符紧跟在第2个字段的前面。

- AS语句应与相应的字段在同一行，多个字段的AS建议尽量对齐在同一列上。

```
select  channel_id              as channel_id
        ,trade_channel_desc     as trade_channel_desc
        ,trade_channel_edesc    as trade_channel_edesc
        ,inst_date              as inst_date
        ,trade_iswap            as trade_iswap
        ,channel_type           as channel_type
        ,channel_second_desc    as channel_second_desc
from    (
```

- INSERT子句排列要求

  INSERT子句写在同一行，请勿换行。

- SELECT子句排列要求

  SELECT语句中所用到的from、where、group by、having、order by、join和union等子句，需要遵循如下要求：

  - 换行编写。
  - 与相应的SELECT语句左对齐编排。
  - 子句首个单词后添加2个缩进量，再编写后续的代码。
  - WHERE子句下的逻辑判断符and、or等，与WHERE左对齐编排。
  - 超过2个缩进量长度的子句加1个空格后，再编写后续代码，例如order by和group by等。

```
select      trim(channel) channel
            ,min(id)      id
from        ods_trd_trade_base_dd
where       channel is not null
and         dt = ${tmp_uuuummdd}
and         trim(channel) <> ''
group by    trim(channel)
order by    trim(channel)
```

- 运算符前后间隔要求

  算术运算符、逻辑运算符前后要保留1个空格，并写在同一行（超过每行80个字符长度的限制除外）。

```
select      trim(channel) channel
            ,min(id)      id
from        ods_trd_trade_base_dd
where       channel is not null
and         dt = ${tmp_uuuummdd}
and         trim(channel) <> ''
group by    trim(channel)
order by    trim(channel)
```

- CASE语句的编写

  CASE语句可以用于SELECT语句中对字段值进行判断取值的操作。CASE语句编排的规则如下：

  - WHEN子语在CASE语句的同一行，并缩进1个缩进量后开始编写。

- 每个WHEN子句尽量在1行内编写，如果语句较长可以换行。

```
,case   when p1.trade_from = '3008' and p1.trade_email is null then 2
        when p1.trade_from = '4000' and p1.trade_email is null then 1
        when p9.trade_from_id is not null then p9.trade_from_id
 end                                  as trade_from_id
,p1.trade_email                       as partner_id
```

- CASE语句必须包含ELSE子语，ELSE子句与WHEN子句对齐。

- 查询嵌套编写规范

  在数据仓库系统ETL开发中经常使用子查询嵌套，其编写规范示例如下。

```
select     p.channel
           ,rownumber()  order_id
from       (
               select   s1.channel
                        ,s1.id
               from     (
                            select  trim(channel)       as channel
                                    ,min(id)            as id
                            from    ods_trd_trade_base_dd
                            where   channel is not null
                            and     dt = ${tmp_yyyymmdd}
                            and     trim(channel) <> ''
                            group by trim(channel)
                        ) s1
               left outer join
                       dim_trade_channel s2
               on      s1.channel = s2.trade_channel_edesc
               where   s2.trade_channel_edesc is null
               order by id
           ) p
;
```

- 表别名定义约定
  - 一旦在SELECT语句中给操作表定义了别名，在整个语句中对此表的引用都必须以别名替代，所以需要给所有的表添加别名。
  - 表别名采用简单字符命名，建议按a、b、c、d...的顺序进行命名，并避免使用关键字。

- 多层次的嵌套子查询别名之前要体现层次关系，SQL语句的别名需要分层命名，从第1层次至第4层次，分别用P（Part）、S（Segment）、U（Unit）和D（Detail）表示。您也可以用a、b、c、d来表示第1层次到第4层次。

  对于同一层次的多个子句，在字母后加1、2、3、4.....区分，并根据情况对表别名添加注释。

```
select     p.channel
           ,rownumber()  order_id
from       (
               select    s1.channel
                         ,s1.id
               from      (
                                 select  trim(channel)       as channel
                                         ,min(id)            as id
                                 from    ods_trd_trade_base_dd
                                 where   channel is not null
                                 and     dt = ${tmp_yyyymmdd}
                                 and     trim(channel) <> ''
                                 group by trim(channel)
                          ) s1
               left outer join
                         dim_trade_channel s2
               on        s1.channel = s2.trade_channel_edesc
               where     s2.trade_channel_edesc is null
               order by id
           ) p
;
```

- SQL注释
  - 每条SQL语句均应添加注释说明。
  - 每条SQL语句的注释单独成行，并放在语句的前面。
  - 字段注释紧跟在字段后面。
  - 对不易理解的分支条件表达式添加注释。
  - 对重要的计算添加注释，说明其功能。
  - 过长的函数实现，应将其语句按实现的功能分段，添加注释进行说明。
  - 添加常量及变量的注释时，应注释被保存值的含义（必选），合法取值的范围（可选）。
  - 将鼠标放置对应SQL语句之后，使用 `Crtl+/` 或 `Cmd+/` 快捷键即可注释当前语句。如果您需要注释多行语句，则可以选中需要注释的语句，使用 `Crtl+/` 或 `Cmd+/` 批量完成注释。

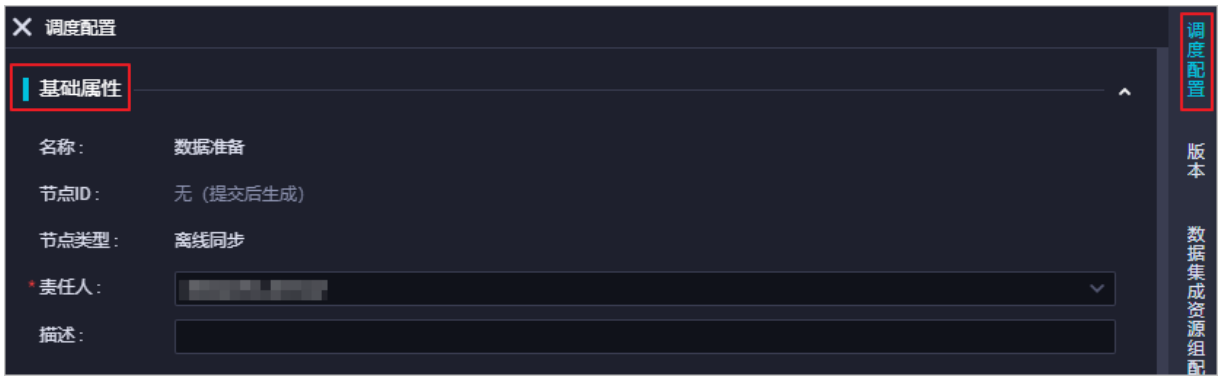
    > **说明**
    > - Windows系统仅支持使用 `Crtl+/` 快捷键注释SQL语句。
    > - Mac系统仅支持使用 `Cmd+/` 快捷键注释SQL语句。

9.调度配置

9.1. 配置基础属性

您可以在调度配置对话框查看调度节点的名称、ID、类型，并配置责任人、描述等基本信息。本文为您介绍调度节点基础属性的参数配置。

进入数据开发节点的编辑页面，单击右侧导航栏的**调度配置**，在**基础属性**区域配置调度节点的基本信息。



参数	描述
名称	新建数据开发节点时输入的节点名称，不可以修改。
节点ID	数据开发节点提交后会生成唯一的节点ID，不可以修改。
节点类型	新建数据开发节点时选择的节点类型，不可以修改。
责任人	数据开发节点的责任人。默认为当前登录用户，您也可以根据实际需求修改责任人。 **说明** 仅支持选择当前DataWorks工作空间中的成员为责任人。
描述	调度节点的描述。通常用于呈现节点业务、用途等信息。

9.2. 配置调度参数

9.2.1. 调度参数概述

调度参数是DataWorks任务调度时使用的参数，调度参数会根据任务调度的业务日期、定时时间及参数的取值格式自动替换为具体的值，实现在任务调度时间内参数的动态替换。本文为您介绍调度参数的相关概况。

您可以通过以下内容了解调度参数：

- 调度参数是根据任务调度的业务日期、定时时间进行参数取值，其取值定义详情请参见参数取值定义。
- 调度参数通过赋值方式可分为**自定义参数**和**系统内置变量**两大类，详情请参见参数分类。
- 调度参数的具体参数列表，详情请参见参数列表。

了解调度参数后，您可以进入**调度配置**界面进行参数的实际配置与使用，详情请参见配置及使用调度参数。

另外，DataWorks为您提供了调度参数常用场景的相关示例：

- 各类型节点调度参数的配置，详情请参见各类型节点的调度参数配置示例。
- 对于自定义参数，不同格式的取值对比，详情请参见自定义参数取值差异对比。
- 调度参数的典型场景配置，详情请参见调度参数返回值二次处理的典型场景。

参数取值定义

调度参数的取值定义如下表所示。

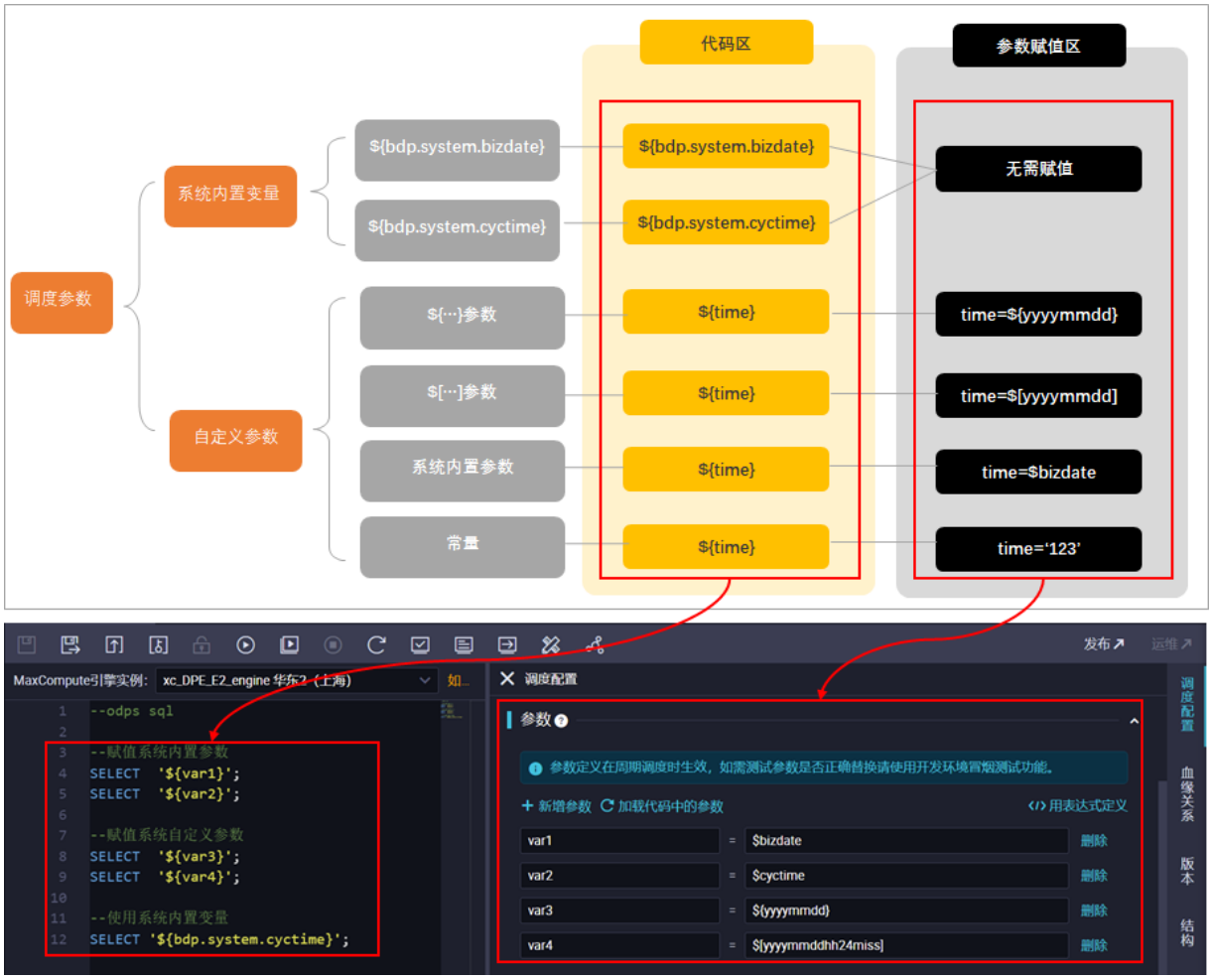
参数取值	描述
业务日期	指在调度时间内，任务预期调度运行时间的前一天（即昨天），精确到天。业务日期可通过 `$bizdate` 、 `${yyyymmdd}` 获取。通常，业务日期为 **定时时间所在日期-1** 。
定时时间	指在调度时间内，任务预期调度运行的时间点（即今天），精确到秒。定时时间可通过 `$cyctime` 、 `$[yyyymmddhh24miss]` 获取。

参数分类

调度参数通过赋值方式分为**自定义参数**和**系统内置变量**两大类。

- **系统内置变量**：在代码中直接获取业务日期与定时时间。
- **自定义参数**：支持您在代码中根据业务需求自定义变量名称，再通过**调度配置 > 参数**赋值区域，统一为代码中的变量赋值自定义的调度参数格式，从而动态获取不同格式的时间参数。

具体如下图所示。



> **说明**
>
> - 常用类型节点均可按照上图方式定义参数并为参数赋值。通用Shell节点、Pyodps节点使用的调度参数有部分差异，详情请参见各类型节点的调度参数配置示例。
> - 部分节点不支持使用调度参数，例如，HTTP触发式节点。各类型的节点是否支持使用调度参数，具体请以对应节点的帮助文档为准。

- **自定义参数**

  DataWorks支持赋值的自定义参数类别如下表所示。

参数类别	描述	参数格式	说明
系统内置参数	支持在赋值区域为变量赋值 `$bizdate` 、 `$cyctime` 来获取业务时间与定时时间。	格式固定。 ○ `$bizdate` 的格式为 `yyyymmdd` 。 ○ `$cyctime` 的格式为 `yyyymmddhh24miss`	不涉及

<table>
<tr><th>参数类别</th><th>描述</th><th>参数格式</th><th>说明</th></tr>
<tr><td>${..}参数</td><td>基于系统内置参数 $bizdate ，通过 yyyy 、 yy 、 mm 和 dd 自定义组合而生成的时间参数。</td><td>可自定义格式。
例如， ${yyyy} 、 ${yyyymm} 、 ${yyyymmdd} 和 ${yyyy-mm-dd} 等。
说明 $bizdate 的取值与 ${yyyymmdd} 一致。</td><td rowspan="2">○ 取N年前、N月前的时间数据需要使用${...}参数。
说明
■ ${...}参数只能精确到年月日，因此${...}参数不支持 ${yyyy-mm-dd-1/24} 用法。
■ 如果需要对年份、月份进行计算，建议使用${...}参数。例如， ${yyyy-N} 、 ${mm-N} 。
○ 取N小时前、N分钟前的时间数据需要使用$[...]参数。
说明
■ $[...]参数不支持获取 $[yyyy-N] 、 $[mm-N] 等时间数据。此处N指年、月，即$[...]参数不支持通过此格式直接获取多少年前，多少月前等时间数据。
■ 如果需要获取精确到秒的时间数据时，建议使用$[...]参数。例如， $[yyyy-mm-dd-1-1/24] 。
调度参数配置完成后，您可以参考测试调度参数测试调度参数来测试参数配置是否正确。</td></tr>
<tr><td>$[...]参数</td><td>基于系统内置参数 $cyctime ，通过 yyyy 、 yy 、 mm 、 dd 、 hh24 、 mi 和 ss 自定义组合而生成的时间参数。</td><td>可自定义格式。
例如， $[yyyymmdd] 、 $[yyyy-mm-dd] 、 $[hh24miss] 、 $[hh24:mi:ss] 和 $[yyyymmddhh24miss] 等。
说明 $cyctime 的取值与 $[yyyymmddhh24miss] 一致。</td></tr>
<tr><td>常量</td><td>支持在赋值区域为变量赋值常量。例如， "abc" 、 1234 。</td><td>无固定格式。</td><td>不涉及</td></tr>
</table>

自定义参数取值格式对比示例，详情请参见自定义参数取值差异对比。

- **系统内置变量**

  DataWorks支持的系统内置变量如下表所示。

参数类别	参数格式	说明
业务日期：`${bdp.system.bizdate}`	固定格式：`yyyymmdd`。	无需手动赋值，参数可以在代码中直接引用。
定时时间：`${bdp.system.cyctime}`	固定格式：`yyyymmddhh24miss`	

> **说明** 仅公共云和专有云支持上述两个参数。

参数列表

- **系统内置参数**

  DataWorks支持的系统内置参数如下表所示。

内置参数	定义
$bizdate	业务日期，格式为`yyyymmdd`。 该参数的应用较为广泛，日常调度中默认任务预期运行时间的前一天为业务日期。
$cyctime	任务的定时时间，格式为`yyyymmddhh24miss`。
$gmtdate	当前日期，格式为`yyyymmdd`。 该参数默认取当天日期，执行补数据操作时输入的日期为`业务日期+1`。
$bizmonth	业务月份，格式为`yyyymm`。 ○ 如果业务日期的月份与当前月份一致，则`$bizmonth=业务日期月份-1`。 ○ 如果业务日期的月份与当前月份不一致，则`$bizmonth=业务日期月份`。
$jobid	任务所属的业务流程ID。
$nodeid	节点ID。
$taskid	节点产生的实例ID。

- **自定义参数${...}**

  根据业务日期的系统内置参数`$bizdate`（昨天）获取以下时间周期的取值。

日期加减周期	获取方式
前/后N年	`${yyyy±N}`

日期加减周期	获取方式
前/后N月	`${yyyymm±N}`
前/后N周	`${yyyymmdd±7*N}`
前/后N天	`${yyyymmdd±N}`
年月日加/减N天	`${yyyymmdd±N}`
加/减N年（yyyy格式）	`${yyyy±N}`年
加/减N年（yy格式）	`${yy±N}`年

其中：

- `yyyy`：表示4位的年份，取值为`$bizdate`的年份。
- `yy`：表示2位的年份，取值为`$bizdate`的年份。
- `mm`：表示月份，取值为`$bizdate`的月份。
- `dd`：表示天，取值为`$bizdate`的天。

> 说明
> - 获取`多少月`、`多少年`前等时间数据请使用${...}参数。
> - 由于`$bizdate`只支持精确到天，因此${...}参数仅支持取值到天。
> - 您可以结合引擎函数，获取更多参数取值，详情请参见调度参数返回值二次处理的典型场景。

- **自定义参数$[...]**

  根据任务定时时间的系统内置参数`$cyctime`（今天）获取以下时间周期的取值。

时间加减周期	获取方式
后N年	`$[add_months(yyyymmdd,12*N)]`
前N年	`$[add_months(yyyymmdd,-12*N)]`
后N月	`$[add_months(yyyymmdd,N)]`
前N月	`$[add_months(yyyymmdd,-N)]`
前/后N周	`$[yyyymmdd±7*N]`
前/后N天	`$[yyyymmdd±N]`
前/后N小时	获取该时间数据包含如下两种方式： - `$[hh24miss±N/24]` - `$[自定义时间格式±N/24]`。例如，`$[hh24±N/24]`

时间加减周期	获取方式
前/后N分钟	获取该时间数据包含如下两种方式： ○ `$[hh24miss±N/24/60]` ○ `$[自定义时间格式±N/24/60]` 。例如，`$[mi±N/24/60]` 、`$[yyyymmddhh24miss±N/24/60]`

其中：

- `yyyy` ：表示4位的年份，取值为 `$cyctime` 的年份。
- `yy` ：表示2位的年份，取值为 `$cyctime` 的年份。
- `mm` ：表示月份，取值为 `$cyctime` 的月份。
- `dd` ：表示天，取值为 `$cyctime` 的天。
- `hh24` ：表示小时（12进制使用 `hh` ），取值为 `$cyctime` 的小时。
- `ss` ：表示秒，取值为 `$cyctime` 的秒。
- `mi` ：表示分钟，取值为 `$cyctime` 的分钟。

> 说明
>
> - 获取 `多少小时` 、 `多少分钟` 前等时间数据请使用$[...]参数。
> - 由于 `$cyctime` 精确到时分秒，因此$[...]参数可以精确到时分秒。
> - 您可以结合引擎函数，获取更多参数取值，详情请参见调度参数返回值二次处理的典型场景。
> - 调度参数替换值在实例生成时已经确定，所以调度参数的替换值不会随着实例实际运行时间的改变而改变。
> - 当调度参数取小时、分钟时，参数替换值由实例的定时时间决定，即由节点调度配置的定时调度时间决定。
>
>   如下示例：
>
>   - 如果当前节点为日调度节点，并且设置定时调度时间为 `01：00` ，则小时的参数取值为 `01` 。
>   - 如果当前节点为小时调度节点，并且设置定时调度时间为 `00：00~23：59` ，每小时调度一次，则：
>   - 第一个小时实例定时时间为0点，小时的参数取值为 `00` 。
>   - 第二个小时实例定时时间为1点，小时的参数取值为 `01` 。
>   - 以此类推。

9.2.2. 配置及使用调度参数

调度参数是根据任务调度的业务时间及调度参数的取值格式自动替换为具体的值，实现在任务调度时间内参数的动态替换。本文为您介绍如何配置及使用调度参数，并以ODPS SQL节点为例，讲解调度参数配置完成后使用冒烟测试功能测试调度参数的替换情况。

调度参数配置流程

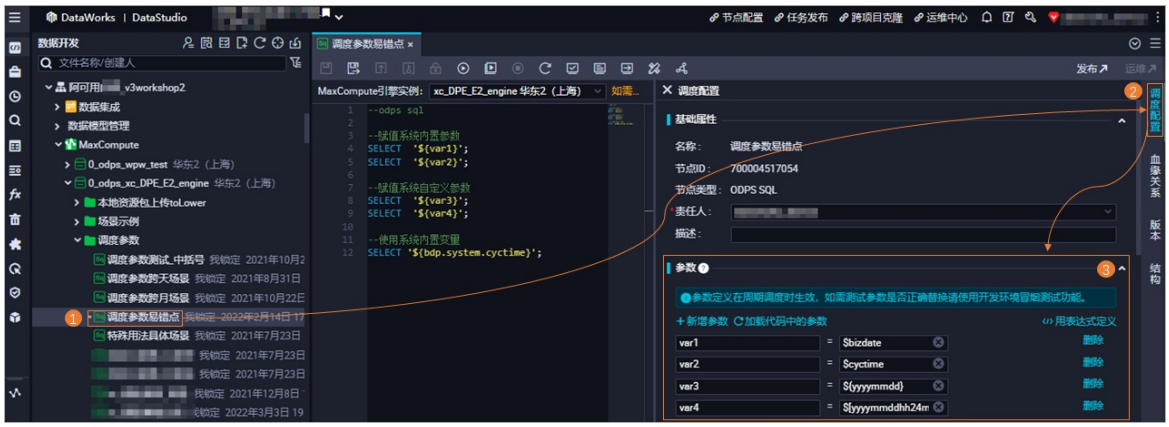
配置与使用调度参数的基本流程如下：

1. 进入调度参数的配置页面，详情请参见调度参数配置入口。
2. 配置调度参数，详情请参见配置调度参数。
3. 测试调度参数配置是否符合预期，详情请参见测试调度参数。
4. 确认生产环境任务的调度参数的配置情况，详情请参见确认生产环境任务的调度参数配置。

完整的调度参数配置流程示例，详情请参见完整配置示例。更多类型节点调度参数的配置示例，详情请参见各类型节点的调度参数配置示例。

调度参数配置入口

1. 进入数据开发。
   i. 登录DataWorks控制台，选择目标区域后，在左侧导航栏单击**工作空间列表**。
   ii. 单击目标工作空间后的**进入数据开发**，即可进入该工作空间的数据开发（DataStudio）模块。
2. 进入调度参数配置界面。



   i. 在数据开发界面的目录树，双击目标节点（本文以*调度参数易错点*节点为例），进入节点的编辑页面。
   ii. 在节点编辑页面，单击右侧导航栏的**调度配置**。
   iii. 在**调度配置**对话框的**参数**区域，即可配置目标节点的调度参数。

配置调度参数

在**参数**区域，您可以通过可视化方式新增参数、加载代码编辑器中已有的参数，或使用表达式方式新增参数。

参数定义方式	功能点	描述	配置图示

参数定义方式	功能点	描述	配置图示
可视化方式	**新增参数**	同一个调度任务可以配置多个调度参数，当需要使用多个调度参数时，您可以单击**新增参数**添加。 系统为您提供了如下参数赋值： • **昨天**：`${yyyymmdd}` • **今天**：`$[yyyymmdd]` • **前一小时**：`$[hh24-1/24]` • **本月第一天**：`${yyyymm}01` • **上月第一天**：`${yyyymm-1}01` 更多调度参数的赋值，详情请参见调度参数概述。	
	加载代码中的参数	用于自动识别当前任务代码中定义的变量名，并将识别到的变量名添加为调度参数，便于调度任务后续使用。 **说明** • 系统内置变量无需赋值，因此所加载的变量均为自定义参数，加载后您需要自行为该参数赋值。 • 通常，代码中是按照`${自定义变量名}`方式来定义变量名。Pyodps节点、通用Shell节点对于变量名的定义方式与其他节点存在差异。各类型节点的调度参数配置格式，详情请参见各类型节点的调度参数配置示例。	

参数定义方式	功能点	描述	配置图示
表达式方式	**用表达式定义**	调度参数配置界面默认使用可视化方式定义参数，如果您习惯使用表达式定义，则可单击**用表达式定义**进行参数定义。 **说明** • 使用表达式定义时，多个参数之间使用空格分隔。例如，配置 `datetime1=$[yyyymmdd]` 、 `datdatetime2=$bizdate` 两个调度参数，则表达式定义的格式为 `datetime1=$[yyyymmdd] datetime2=$bizdate` 。 • 当您通过**用表达式定义**方式，添加、删除或修改调度参数时，DataWorks会对当前表达式的语法进行校验，校验不通过则无法配置相应调度参数。例如，DataWorks会对*等号左右不允许使用空格*等语法规则进行检测。	

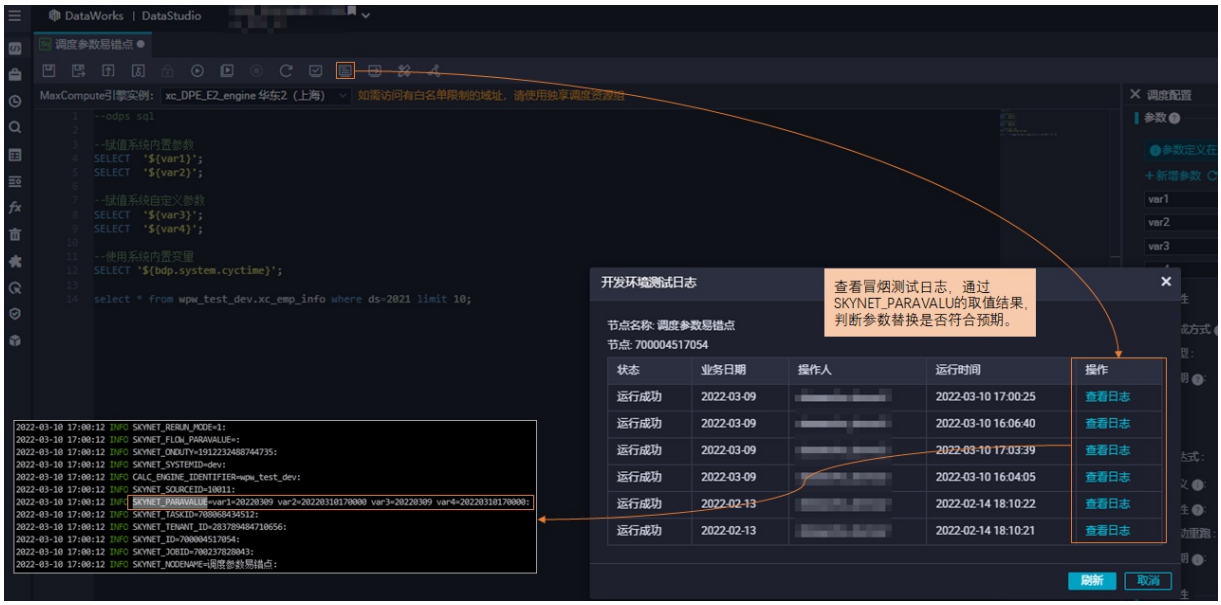
说明 如果任务代码中直接使用 `${bdp.system.bizdate}` 或 `${bdp.system.cyctime}` 参数，则无需在**参数**区域进行赋值。

测试调度参数

调度参数配置完成后，您需要使用**在开发环境执行冒烟测试**（）功能，通过配置业务日期，模拟目标任务的调度场景，来验证该场景下调度参数的替换情况是否符合预期。

说明 执行冒烟测试时，会生成相应实例产生实例费用。实例的费用详情请参见公共调度资源组计费说明：按量付费及独享调度资源组计费说明：包年包月。

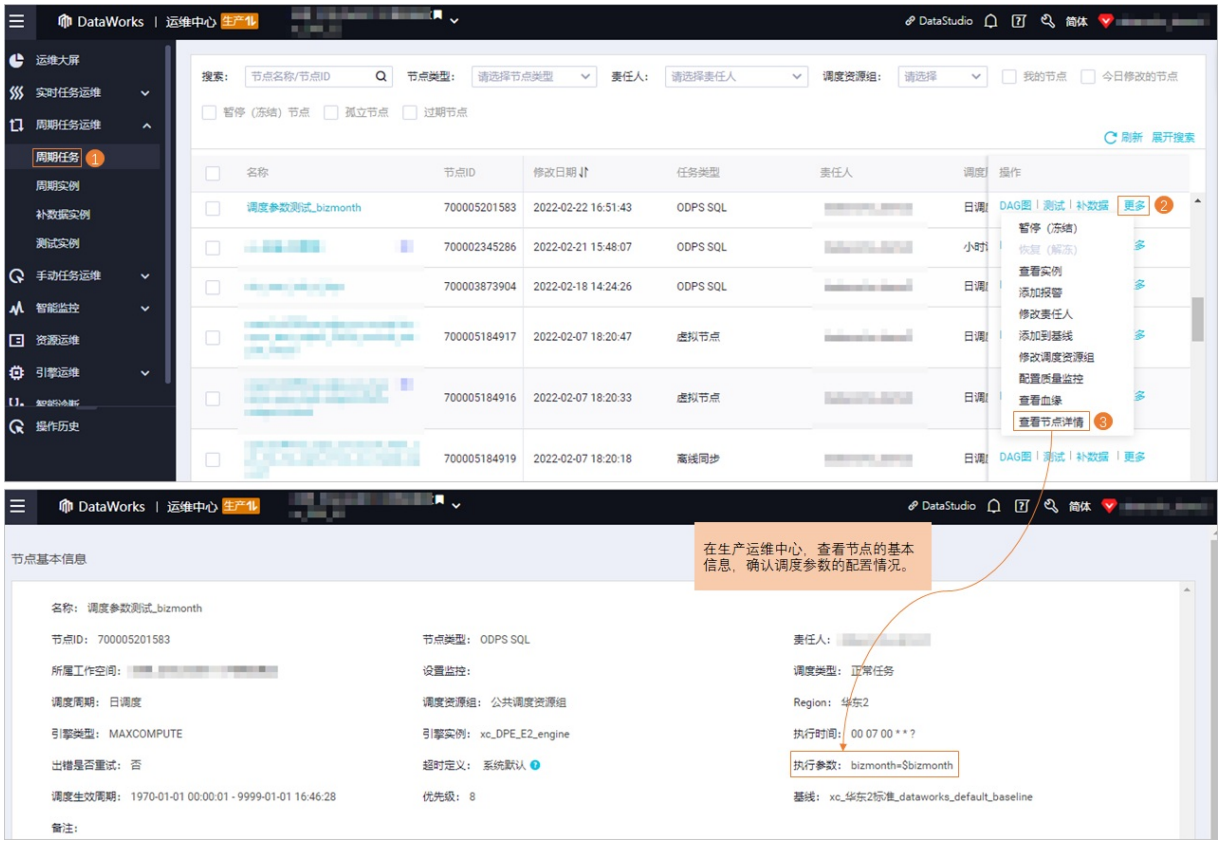
在指定运行时间完成后，单击图标，查看冒烟测试日志中的结果是否符合预期。



> **说明**
>
> - 运行（）与高级运行（）功能需要您手动为代码中的变量赋值常量，因此无法校验配置的调度参数是否符合预期。
> - 修改代码后，请及时保存（）并提交（）节点。当节点最新代码提交至开发环境后，才可使用**在开发环境执行冒烟测试**功能。

确认生产环境任务的调度参数配置

为避免周期调度任务运行时，由于配置的调度参数不符合预期，导致任务运行出现问题，建议您在任务发布后，前往生产运维中心的**周期任务**界面，查看生产环境下该周期任务的调度参数配置情况。查看周期任务，详情请参见查看并管理周期任务。



> **说明** 如果周期任务的调度参数配置不符合预期，或运维中心搜索不到目标任务，请确认该任务是否发布成功。任务的发布操作，详情请参见发布任务。

完整配置示例

本文以ODPS SQL节点为例，通过**在开发环境执行冒烟测试**功能测试配置的调度参数是否符合预期，并在任务发布后，查看生产运维中心中该任务的调度参数配置情况。

> **说明** 各类型节点的调度参数配置，详情请参见各类型节点的调度参数配置示例。

1. 编辑节点代码并配置调度参数。

   ODPS SQL节点的代码及调度参数的配置情况如下图所示。

i. 代码中定义变量。

在ODPS SQL节点代码中引用系统内置参数 `'${var1}'` 、 `'${var2}'` ，系统自定义参数 `'${var3}'` 、 `'${var4}'` 变量（如区域1）。

ii. 为变量赋值。

在**调度配置 > 参数**区域，为变量赋值（如区域2）。

- `var1=$bizdate` ，即取 `yyyymmdd` 格式的业务日期。
- `var2=$cyctime` ，即取 `yyyymmddhh24miss` 格式的任务定时运行时间。
- `var3=${yyyymmdd}` ，即取 `yyyymmdd` 格式的业务日期。
- `var4=$[yyyymmddhh24miss]` ，即取 `yyyymmddhh24miss` 格式的任务定时运行时间。

iii. （可选）配置时间周期。

配置ODPS SQL节点的调度周期为小时调度（如区域3）。

> **说明** 您可以根据实际情况选择是否配置时间周期，本文以添加时间周期示例。

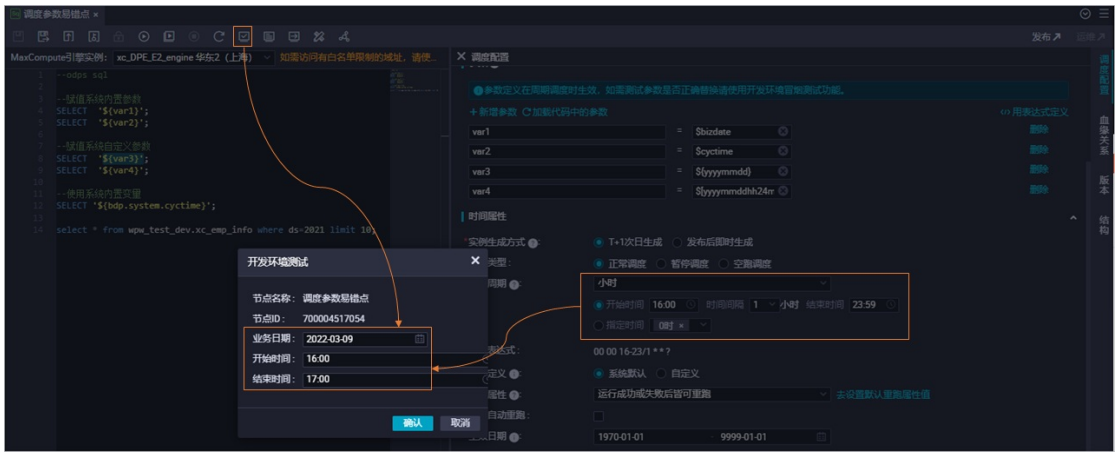
- 调度开始时间为 `16:00`
- 调度结束时间为 `23:59`
- 调度时间间隔为 `1` 小时。

更多时间周期配置，详情请参见时间属性配置说明。

2. 在节点编辑页面的顶部工具栏，单击及图标，保存并提交ODPS SQL节点的配置。

3. 执行开发环境冒烟测试。

i. 单击图标，在**开发环境测试**对话框配置业务时间，模拟节点的调度周期。



业务时间配置如下：

- 业务日期：`2022-03-09`
- 开始时间：`16：00`
- 结束时间：`17：00`

ODPS SQL任务为小时调度任务，则该任务在`2022-03-10`的`16：00`、`17：00`时间会生成两个实例。

> **说明** 因为业务日期为运行日期的前一天，因此，任务实际运行日期为`2022-03-10`。

`16：00`节点预期的取值结果如下：

- `var1=20220309`。
- `var2=20220310160000`。
- `var3=20220309`。
- `var4=20220310160000`。

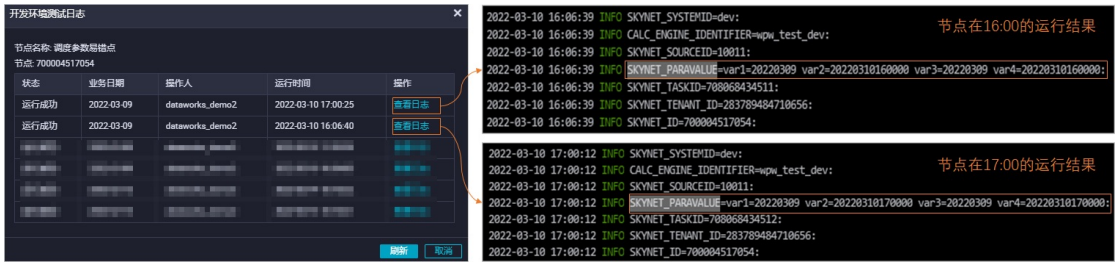
`17：00`节点预期的取值结果如下：

- `var1=20220309`。
- `var2=20220310170000`。
- `var3=20220309`。
- `var4=20220310170000`。

ii. 单击**确认**，节点按照指定时间启动运行。

iii. 运行时间结束后，单击图标，查看冒烟测试日志。

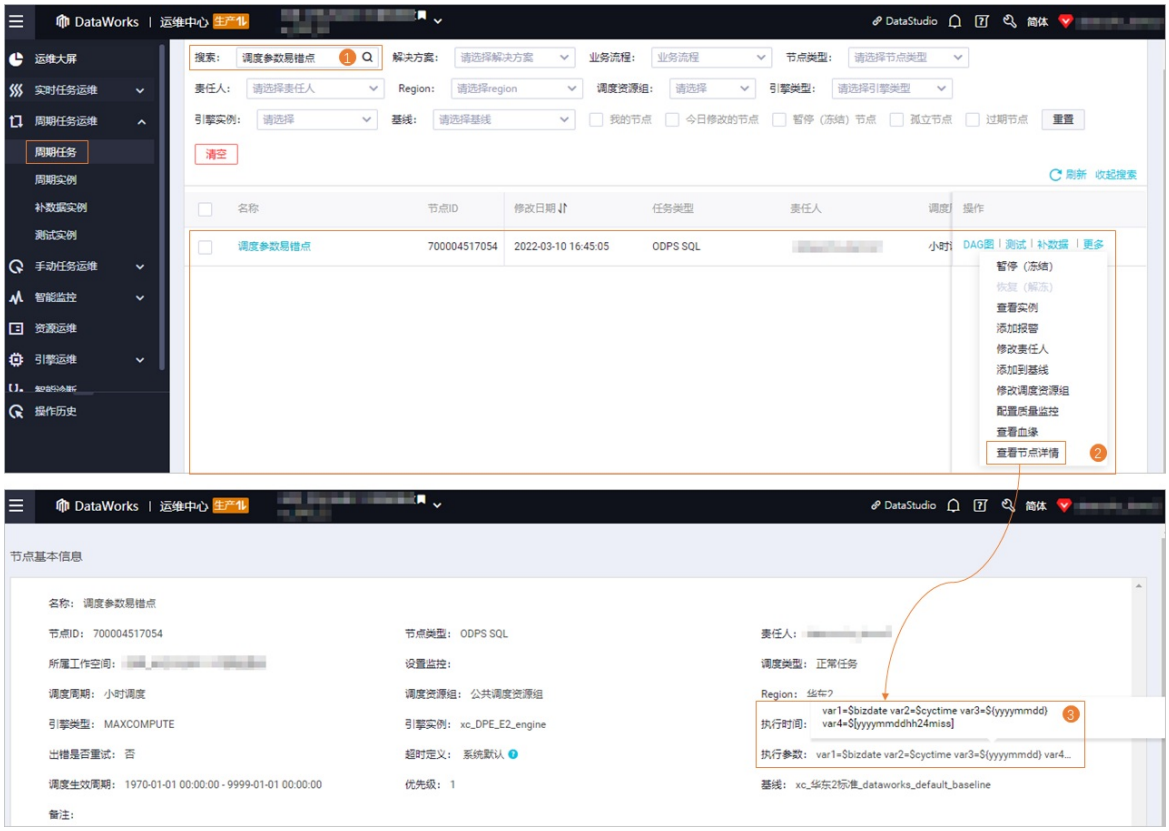
节点生成的两个实例运行成功，且节点运行结果符合预期。



4. 在ODPS SQL节点编辑页面，单击顶部菜单栏右侧的**发布**，发布当前节点。

任务的发布操作，详情请参见发布任务。

5. 进入运维中心，确认节点的调度参数配置。



i. 单击DataStudio顶部菜单栏右侧的**运维中心**，进入运维中心页面。

ii. 在**周期任务运维 > 周期任务**界面，搜索目标节点。

> **说明** 节点发布成功后，您才能在周期任务界面搜索到。

iii. 单击目标节点**操作**列的**更多 > 查看节点详情**，在**节点基本信息**中查看**执行参数**。

本次示例中，节点的执行参数为 `var1=$bizdate var2=$cyctime var3=${yyyymmdd} var4=$[yyyymmddhh24miss]`，符合预期。

9.2.3. 场景示例

9.2.3.1. 各类型节点的调度参数配置示例

除通用Shell节点、PyODPS节点外，其他类型节点均可参考SQL类型节点（例如，ODPS SQL）的配置方式定义参数并为参数赋值。通用Shell节点、PyODPS节点的调度参数使用存在部分差异。本文为您介绍各类型节点的调度参数配置示例。

SQL类型节点及离线同步节点

SQL类型节点及离线同步节点的调度参数配置与多数类型节点的配置相似，可供多数节点参考使用。本文以ODPS SQL节点为例，为您展示如何为系统内置变量及自定义参数赋值，并在代码中进行调用。

> **说明** 部分节点可能不支持使用调度参数。各节点是否支持使用调度参数，详情请参考具体的节点文档。

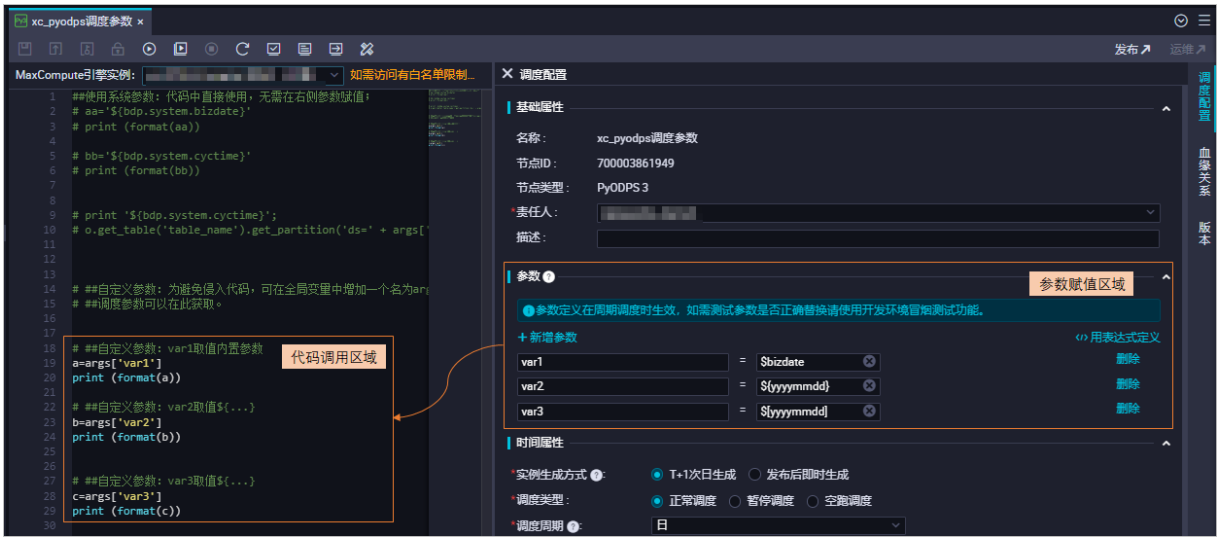
如上图，在参数赋值区域为参数赋值，然后进入代码调用区域引用系统内置变量var1、var3，自定义参数var2、var4，常量var5。赋值示例如下：

- 系统内置变量var1赋值取业务时间：`var1=$bizdate`
- 系统内置变量var3赋值取定时时间：`var3=$cyctime`
- 自定义参数var2赋值取业务时间：`var2=${yyyymmdd}`
- 自定义参数var4赋值取定时时间：`var4=$[yyyymmddhh24:mi:ss]`
- 常量var5参数赋值为abc：`var5=abc`

配置及使用调度参数，详情请参见配置及使用调度参数，更多调度参数的赋值方式，详情请参见调度参数概述。

PyODPS节点

为避免代码入侵，PyODPS节点不支持在代码中直接使用 `${param_name}` 格式的字符串替换定义的变量。执行代码前，您需要在全局变量中增加一个名为 `args` 的dict（字典对象），在此处获取调度参数。



如上图，在参数赋值区域为参数赋值，然后进入代码调用区域引用内置参数var1，自定义参数var2、var3。添加字典对象后的参数为 `args['var1']` 、 `args['var2']` 、 `args['var3']` 。赋值示例如下：

- 内置参数var1赋值取业务时间： `var1=$bizdate`
- 自定义参数var2赋值取业务时间： `var2=${yyyymmdd}`
- 自定义参数var3赋值取业务时间： `var3=$[yyyymmdd]`

配置及使用调度参数，详情请参见配置及使用调度参数，更多调度参数的赋值方式，详情请参见调度参数概述。

通用Shell节点配置示例

通用Shell节点中的变量不允许自定义命名，只能以 `$1` 、 `$2` 、 `$3` ...命名（参数序号由小到大，依次递增），当参数的数量大于10时，请使用 `${10}` 的方式声明变量。

如上图，在参数赋值区域为参数赋值，然后进入代码调用区域定义内置参数$1，自定义参数$2、$3。赋值示例如下：

> **说明** 通用Shell节点仅支持使用表达式方式为参数赋值。多个参数赋值使用空格分隔，并且参数取值与定义参数时的顺序对应。例如，上图Shell节点定义的第一个参数为$1，则参数赋值区域序号第一的赋值内容 `$bizdate` 即为$1的参数取值。

- 内置参数$1赋值取业务时间：`$bizdate`
- 自定义参数$2赋值取业务时间：`${yyyymmdd}`
- 自定义参数$3赋值取业务时间：`$[yyyymmdd]`

配置及使用调度参数，详情请参见配置及使用调度参数，更多调度参数的赋值方式，详情请参见调度参数概述。

9.2.3.2. 自定义参数取值差异对比

自定义参数分为系统内置参数、自定义参数${..}、自定义参数$[...]、常量，不同类别的参数赋值格式及取值不同。本文为您介绍不同格式自定义参数的取值差异对比情况。

不同格式自定义参数的应用对比

以当前时间为`2021年11月01日`，任务每天`00:00`定时运行，示例不同格式自定义参数的赋值情况，具体如下表。

> **说明** 假设代码引用方式均为`pt=${datetime}`。

参数格式	描述	调度参数赋值	参数替换结果
`${yyyymmdd}`	获取业务时间。	`datetime=${yyyymmdd}`	`datetime=20211031`
`$[yyyymmddhh24miss]`	获取定时时间，精确到秒。	`datetime=$[yyyymmddhh24miss]`	`datetime=202111010000`
`$bizdate`	获取业务时间。	`datetime=$bizdate`	`datetime=20211031`
`$cyctime`	获取定时时间，精确到秒。	`datetime=$cyctime`	`datetime=20211101000000`
`$gmtdate`	获取当前时间，精确到天。	`datetime=$gmtdate`	`datetime=20211101`

参数格式	描述	调度参数赋值	参数替换结果
$bizmonth	获取业务月份。	datetime=$bizmonth	• 当业务时间与当前月份一致时，$bizmonth的值为当前时间上个月的月份。 • 当业务时间与当前月份不一致时，$bizmonth的值为业务时间的月份。 示例当前时间为2021年11月01日： • 如果业务时间为2021年11月02日（当前月份），则datetime=202110。 • 如果业务时间为2021年10月31日（非当前月份），则datetime=202110。

${...}和$[...]参数的功能差异

${...}和$[...]参数的功能差异如下表所示。

对比项	${...}参数	$[...]参数
时间基准	以$bizdate的取值时间为基准参与运算。 $bizdate为业务日期，默认为当前时间的前一天。	以$cyctime的时间为基准参与运算。 $cyctime为任务的定时调度时间。例如，任务的定时调度时间为当天的00:30，即时间格式为yyyy-mm-dd 00:30:00。
补数据功能	补数据时选择的业务日期和调度参数的替换结果保持一致。	执行补数据时，调度参数替换结果为选择的业务日期+1天。 例如，补数据选择的业务日期为20220315，则执行补数据时，$cyctime参数的替换结果为20220316。
时间精确度	精确到天。	精确到秒。 **说明** $[...]参数不支持${yyyy-mm-dd-1/24}等用法，建议您使用$[yyyy-mm-dd-1-1/24]。

- ${...}和$[...]参数的更多用法，详情请参见调度参数概述。
- 自定义参数的配置及使用，详情请参见配置及使用调度参数。

本文以ODPS SQL节点为例，假设当前时间为2021年07月20日10时30分00秒，为您展示${...}和$[...]参数的时间取值配置，具体如下表。

时间取值	${...}参数	$[...]参数
取年份：2021	• 调度参数赋值：`datetime=${yyyy}` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=${yyyy}=2021`	• 调度参数赋值：`datetime=$[yyyy]` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=$[yyyy]=2021`
取年份：21	• 调度参数赋值：`datetime=${yy}` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=${yy}=21`	• 调度参数赋值：`datetime=$[yy]` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=$[yy]=21`
取年份：2020	• 调度参数赋值：`datetime=${yyyy-1}` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=${yyyy-1}=2020`	不支持
取月份：07	• 调度参数赋值：`datetime=${mm}` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=${mm}=07`	• 调度参数赋值：`datetime=$[mm]` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=$[mm]=07`
取日期（天）：20	• 调度参数赋值：`datetime=${dd}` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=${dd}=20`	• 调度参数赋值：`datetime=$[dd]` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=$[dd]=20`
取日期：2021年06月20日	• 调度参数赋值：`datetime=${yyyy-mm-dd-29}` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=${yyyy-mm-dd-29}=2021-06-20`	• 调度参数赋值：`datetime=$[add_months(yyyymmdd,-1)]` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=$[add_months(yyyymmdd,-1)]=2021-06-20`
取日期：2021年07月19日	• 调度参数赋值：`datetime=${yyyy-mm-dd}` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=${yyyy-mm-dd}=2021-07-19`	• 调度参数赋值：`datetime=$[yyyy-mm-dd-1]` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=$[yyyy-mm-dd-1]=2021-07-19`

时间取值	${...}参数	$[...]参数
取日期：2020年07月20日	• 调度参数赋值：`datetime=${yyyy-mm-dd-364}` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=${yyyy-mm-dd}=2020-07-20`	• 调度参数赋值：`datetime=$[add_months(yyyymmdd,-12*1)]` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=$[add_months(yyyymmdd,-12*1)]=2020-07-20`
取时间：10:30:00	不支持	• 调度参数赋值：`datetime=$[hh24:mi:ss]` • 代码引用：`pt=${datetime}` • 参数替换结果：`pt=$[hh24:mi:ss]=10:30:00`

取时间：2021-07-20 10:30:00	不支持	• 调度参数赋值：`datetime1=$[yyyy-mm-dd] datetime2=$[hh24:mi:ss]`
		说明 您需要自定义两个参数datetime1和datetime2，两个参数间使用一个空格分隔。
		• 代码引用：`pt=${datetime1} ${datetime2}`
		• 参数替换结果：
		  ◦ `datetime1=$[yyyy-mm-dd]=2021-07-20`
		  ◦ `datetime2=$[hh24:mi:ss]=10:30:00`
		  ◦ `pt=2021-07-20 10:30:00`

时间取值	${...}参数	$[...]参数

取时间：2021-07-20 10:29:00	不支持	• 调度参数赋值：`datetime1=$[yyyy-mm-dd] datetime2=$[hh24:mi:ss-1/24/60]`
		说明 您需要自定义两个参数datetime1和datetime2，两个参数间使用一个空格分隔。
		• 代码引用：`pt=${datetime1} ${datetime2}`
		• 参数替换结果：
		  ○ `datetime1=$[yyyy-mm-dd]=2021-07-20`
		  ○ `datetime2=$[hh24:mi:ss-1/24/60]=10:29:00`
		  ○ `pt=2021-07-20 10:29:00`

取时间：2021-07-20 09:30:00	不支持	• 调度参数赋值：`datetime1=$[yyyy-mm-dd] datetime2=$[hh24:mi:ss-1/24]`
		说明 您需要自定义两个参数datetime1和datetime2，两个参数间使用一个空格分隔。
		• 代码引用：`pt=${datetime1} ${datetime2}`
		• 参数替换结果：
		  ○ `datetime1=$[yyyy-mm-dd]=2021-07-20`
		  ○ `datetime2=$[hh24:mi:ss-1/24]=09:30:00`
		  ○ `pt=2021-07-20 09:30:00`

9.2.3.3. 调度参数返回值二次处理的典型场景

调度参数只支持获取时间类型数据，部分节点（例如，离线同步节点）配置了调度参数后，调度参数的返回值不能直接使用，需要进行函数转换等二次处理。本文为您介绍二次处理调度参数返回结果的典型场景。

背景信息

调度参数支持的时间类型未覆盖所有的时间场景，如果您的业务需要使用特殊时间格式，则可以使用引擎函数进行处理。二次处理调度参数时，部分节点不支持直接使用函数转换参数返回结果。对于不支持直接使用函数转换的节点，您可以通过赋值节点进行相关处理。

直接使用函数，或通过赋值节点二次处理调度参数的典型场景如下：

- **直接使用函数二次处理调度参数**

  本文以ODPS SQL节点为例，为您介绍直接使用函数二次处理调度参数的典型场景：

  - 获取上个月最后一天
  - 获取当前季度
  - 获取定时时间15分钟前的年、月、日、小时、分钟
  - 获取时间区间，调度间隔为1天
  - 获取时间区间，调度间隔为1小时

- **通过赋值节点二次处理调度参数**

  无法直接使用函数二次处理调度参数的节点，如果想直接使用时间戳或其他时间格式，则可以先通过赋值节点将时间类型数据进行相应转换，再将处理后的结果传递给该节点使用。赋值节点的使用，详情请参见赋值节点。

  例如，如果离线同步节点需要使用时间戳类型字段进行增量同步，则可以先通过赋值节点使用函数将时间类型数据转换为时间戳，再传递给离线同步节点使用。

更多操作参考如下：

- ODPS SQL节点的调度参数配置操作，详情请参见配置及使用调度参数及SQL类型节点及离线同步节点。
- 更多调度参数的赋值，详情请参见调度参数概述。
- 跨天场景的调度参数配置，详情请参见调度参数往前取一个小时，如何处理跨天参数替换的问题。

获取上个月最后一天

使用调度参数 **获取上个月最后一天** 的配置及测试结果如下表。

参数配置	测试定时时间CYCTIME	返回结果
• 调度参数配置：`last_month=$[yyyy-mm]` • 处理调度参数的返回值：`SELECT REPLACE(DATEADD(date'${last_month}-01',-1,'dd'),'-','');` • 预期返回格式：`yyyymmdd`	`20210926000000`	`20210831`

获取当前季度

使用调度参数 **获取当前季度** 的配置及测试结果如下表。

参数配置	测试定时时间CYCTIME	返回结果
• 调度参数配置：`month=$[mm]` • 处理调度参数返回值：`SELECT CEIL(INT('${month}')/3);` • 预期返回类型：正整数。	`20211025001700`	4

获取定时时间15分钟前的年、月、日、小时、分钟

使用调度参数 **获取定时时间15分钟前的年、月、日、小时、分钟** 的配置及测试结果如下表。

参数配置	测试定时时间CYCTIME	返回结果

• 调度参数配置：
  ○ `year=$[yyyy-15/24/60]`
  ○ `month=$[yyyymm-15/24/60]`
  ○ `day=$[yyyymmdd-15/24/60]`
  ○ `hour=$[hh24-15/24/60]`
  ○ `mi=$[mi-15/24/60]`
• 处理调度参数返回值：`select 'year=${year} month=${month} day=${day} hour=${hour} mi=${mi}';`
• 预期返回格式：
  ○ 年：`yyyy`
  ○ 月：`yyyymm`
  ○ 日：`yyyymmdd`
  ○ 小时：`hh`
  ○ 分钟：`mm`
	`20210727000500`	• `year=2021`
		• `month=202107`
		• `day=20210726`
		• `hour=23`
		• `mi=50`

获取时间区间，调度间隔为1天

获取昨天 `00:00:00` 到今天 `00:00:00` 的时间区间，调度间隔为1天。时间格式为 `yyyymmddhh24miss`，精确到秒。

> **说明** 使用Kafka和LogHub离线同步指定时间区间的数据时，如需配置调度参数，则调度参数的日期格式为 `yyyymmddhh24miss`，日期区间为左闭右开。详情请参见Kafka Reader和LogHub（SLS）Reader。不同场景的数据增量同步，详情请参见数据增量同步。

调度参数的配置及测试结果如下表。

参数配置	测试定时时间CYCTIME	返回结果

• 调度参数配置：
  ○ `beginDateTime=$[yyyymmdd-1]`
  ○ `endDateTime=$[yyyymmdd]`
• 处理调度参数返回值：`select '${beginDateTime}000000 ${endDateTime}000000';`
• 预期返回格式：`yyyymmddhh24miss`
	`20220117002300`	• `20220116000000`
		• `20220117000000`

获取时间区间，调度间隔为1小时

获取昨天 `00:00:00` 到今天 `00:00:00` 的时间区间，调度间隔为1小时。时间格式为 `yyyymmddhh24miss` ，精确到秒。

> **说明** 使用Kafka和LogHub离线同步指定时间区间的数据时，如需配置调度参数，则调度参数的日期格式为 `yyyymmddhh24miss` ，日期区间为左闭右开。详情请参见Kafka Reader和LogHub（SLS）Reader。不同场景的数据增量同步，详情请参见数据增量同步。

调度参数的配置及测试结果如下表。

参数配置	测试定时时间CYCTIME	返回结果

• 调度参数配置：
  ○ `beginDateTime=$[yyyymmddhh24-1/24]`
  ○ `endDateTime=$[yyyymmddhh24]`
• 处理调度参数返回值：`select '${beginDateTime}0000${endDateTime}0000';`
• 预期返回格式：`yyyymmddhh24miss`
	`20220117002300`	• `20220116230000`
		• `20220117000000`

9.3. 配置时间属性

9.3.1. 时间属性配置说明

用于定义节点在生产环境的周期调度方式。您可以通过调度配置的时间属性，配置节点生成周期实例的方式，实例调度周期与执行时间，是否支持重跑，任务执行超过多长时间自动退出等。本文为您介绍如何配置节点的调度时间属性。

背景信息

您需要进入数据开发节点的编辑页面，单击右侧导航栏的**调度配置**，配置节点的时间属性。

时间属性包含的配置类别如下表所示。

类别	描述
实例生成方式	用于定义节点在生产环境生成实例的时间。
调度类型	用于定义节点在生产环境的运行方式。
调度周期	用于定义节点在生产环境中的运行频率（生成周期实例个数及实例运行的时间）。
超时定义	用于定义节点运行超过多长时间会失败退出。
重跑说明	用于定义节点生成的实例是否可以重跑，即从数据幂等性考虑，任务是否可以重跑，或者在什么情况下可以重跑。
生效日期	用于定义节点正常自动调度运行的时间范围，该时间范围外，节点将不再自动调度。

实例生成方式

节点提交发布生产环境调度系统时，调度节点会根据您配置的**实例生成方式**生成**周期实例**并自动调度。节点的实例生成方式包括**T+1次日生成**和**发布后即时生成**：

- **T+1次日生成**：全量转实例。即节点发布后第二天周期实例生效，并将会自动调度运行。
- **发布后即时生成**：即实时转实例。发布节点当天便会生成周期实例，但只有节点设置的定时时间为未来时间的实例才会正常执行，定时时间为过去时间的实例将会空跑不真实执行任务（此处有十分钟时间间隔），详情请参见实例生成方式：发布后即时生成实例。

> **说明**
>
> 1. 节点发布生产运维中心后，您可以在运维中心周期任务立即查看该节点。调度系统会根据生产运维中心周期任务来生成每天自动调度的周期实例，实例生成方式功能控制的是生产运维中心周期实例面板何时实例生效。进入运维中心查看周期任务，详情请参见查看并管理周期任务。
> 2. 当天23:30分之后发布操作（T+1次生成实例和发布后即时生成实例）不生效。
>   - 每天23:30前提交发布的任务，第二天实例生效。
>   - 每天23:30后提交发布的任务，第三天实例生效。
> 3. 发布后即时生成实例，其定时时间在发布时间点10分钟后的任务才会真实执行并产出数据。
> 4. 发布后即时生成实例，对调度周期变更的任务，当天依赖关系可能存在影响，已存在且为已完成状态的实例不会删除，定时时间为未来时间，且未运行的实例将会被替换，详情请参见：实例生成方式：发布后即时生成实例。

调度类型

DataWorks支持的调度类型如下表所示。

调度类型	影响说明	使用场景
正常调度	按照调度周期配置的定时时间启动调度，正常执行任务（即会真实跑数据）。 当前节点正常执行后，也会触发下游节点正常调度执行，通常任务默认选中该项。	正常状态运行的周期任务，并且生成的周期实例也是正常状态运行。
暂停调度	按照调度周期配置的定时时间启动调度，但节点状态被置为暂停（即不会真实跑数据）。 调度到该任务时，系统会直接返回失败，并且会阻塞依赖当前节点的下游节点执行。 **说明** 暂停状态的节点在运维中心的图标标识为 。	冻结状态的周期任务，其生成的周期实例也为冻结状态。当前节点不可执行，并且阻塞下游节点执行。 当某一类业务流程在一定时间内不需要执行时，可选择此调度类型来冻结业务流程根节点，当业务需要执行时，再对业务流程根节点执行解冻操作。解冻任务，详情请参见任务冻结与解冻。
空跑调度	按照调度周期配置的定时时间启动调度，但该节点为空跑状态（即不会真实跑数据）。 调度到该任务时，系统会直接返回成功（执行时长为 0 秒），不会真正执行任务（即执行日志为空）、不会阻塞依赖当前节点的下游节点执行（即下游节点正常执行）、也不会占用资源。	当某一个节点在一定时间内不需要执行，并且不阻塞他的下游节点执行时，可选择此类型调度。

调度周期

> **说明** 仅当DataWorks工作空间开启**启用调度周期**开关后，工作空间中的任务才可以根据其配置自动调度运行。您需要进入工作空间的**调度设置**页面，开启相应开关，详情请参见调度设置。

- **概念说明**

调度周期即在生产环境调度系统中，多久会真实执行一次节点中的代码逻辑。DataWorks中，当一个任务被成功提交后，底层的调度系统从第二天开始，将会每天按照该任务的时间属性生成自动调度的周期实例，并根据上游依赖的实例运行结果和时间点运行。23:30之后提交成功的任务从第三天开始才会生成周期实例自动调度。

- **周期类型**
  - 分钟调度，详情请参见调度周期：分钟调度。
  - 小时调度，详情请参见调度周期：小时调度。
  - 日调度，详情请参见调度周期：日调度。
  - 周调度，详情请参见调度周期：周调度。
  - 月调度，详情请参见调度周期：月调度。
  - 年调度，详情请参见调度周期：年调度。
- **注意事项**
  - 23:30之后提交成功的任务从第三天开始才会生成实例。
  - 一个周期运行的任务，其依赖关系的优先级大于时间属性。在时间属性决定的某个时间点到达时，任务实例不会马上运行，而是先检查上游是否全部运行成功。

    > **说明**
    > - 上游依赖的实例没有全部运行成功，并且已到定时运行时间，则实例仍为未运行状态。
    > - 上游依赖的实例全部运行成功，并且未到定时运行时间，则实例进入等待时间状态。
    > - 上游依赖的实例全部运行成功，并且已到定时运行时间，则实例进入等待资源状态准备运行。
    >
    > 关于任务运行的条件与排查您可以参考文档：任务到定时时间，为什么还没运行？。

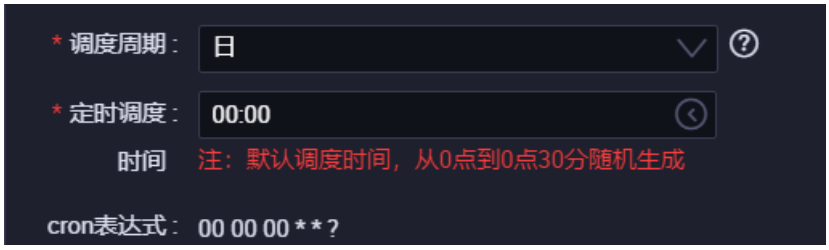
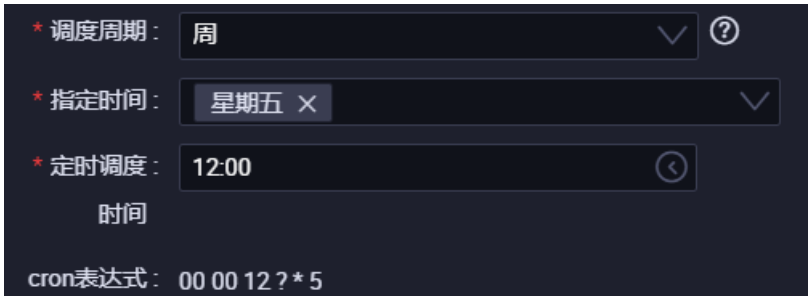
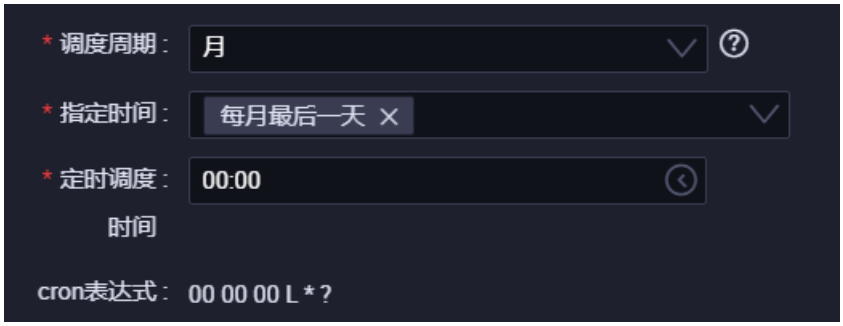
  - 调度周期中设置的是周期实例自动调度的定时执行时间，但任务实际运行时，可能会因为等待资源等原因，导致实际运行时间与定时时间不一致。
  - 包含指定执行日期的周、月、年调度节点，在不真实跑数据的日期内，同样会按照调度周期的定时时间启动调度。但该实例的状态为空跑状态（即不会真实跑数据）。当调度到空跑状态的实例时，其空跑表现如下：
    - 系统直接返回运行成功，即执行时长为 `0` 秒。
    - 不会真正执行任务，即执行日志为空。
    - 不会阻塞依赖当前节点的下游节点执行，即下游节点正常执行。
    - 不会占用资源。

    > **说明**
    >
    > 例如：当一个任务需要每周一执行一次，则只有运行时间是周一的情况下，该任务才会真正执行。运行时间不是周一的情况下，该任务会空跑（直接将任务置为成功），不会实际运行。所以在测试或补数据时，周调度任务需要选择星期天，业务日期为 `运行时间-1` 。从平台维度看，业务时间前一天的任务会在今天执行。

- **场景示例说明**
  - 如果下游日调度节点依赖了上游每周一调度的节点，那么在非周一的时间段内，周调度实例会空跑，下游日调度节点每天正常执行代码逻辑。

- 如果下游节点需要和周调度节点保持一致，即每周一跑一次，其他时间均不真实执行代码逻辑，那么您需要配置下游节点为每周一调度。

- **典型场景与配置示例**

典型场景	配置示例
每天凌晨24点运行	
每周五中午12点运行	
每个月最后一天运行	
每季度首月最后一天运行	

超时定义

设置超时时间后，如果任务运行时长超过超时时间，任务将自动终止运行。其配置说明如下：

- 超时时间对周期实例、补数据实例、测试实例均生效。
- 超时时间默认值为3~7天，系统根据实际负载情况动态调整默认的任务超时时间，范围为3～7天不等。
- 手动设定超时时间时，最大值可设置为168小时（7天）。

> **说明** 2021年1月7日之前购买的独享调度资源组，请提交工单联系技术支持人员升级资源组，升级后才可支持超时时间设置。

重跑说明

您可以在时间属性中配置节点在特定情况下重跑，以及指定重跑时间及次数。

> **说明**
> - 使用重跑属性时，应尽量保证任务的幂等性（特殊任务除外），避免在任务出错重跑后，出现大量数据质量问题。例如，在ODPS SQL的开发过程中，使用 `insert overwrite` 语句来替代 `insert into` 语句。
> - DataWorks工作空间的**调度设置**页面，用于设置重跑相关参数的默认值，设置后，新创建的任务将使用该默认配置。详情请参见调度设置。

- **重跑属性**

  重跑属性不能为空，其支持的类型及应用场景如下表所示。

  > **说明** 单击**去设置默认重跑属性值**，即可设置重跑属性的默认值。

类型	应用场景
运行成功或失败后均可重跑	如果节点多次重跑不会影响结果，可选择使用该重跑类型。
运行成功后不可重跑，运行失败后可以重跑	如果节点运行一次成功后，重跑后会影响运行结果，而运行失败后重跑不会影响结果，可选择使用该重跑类型。
运行成功或失败后皆不可重跑	如果节点不管运行成功或失败，重跑后都会影响运行结果（例如，某些数据同步节点），可选择使用该重跑类型。 **说明** ○ 当选择该类型时，如果系统出故障，在故障恢复后，系统也不会自动重跑相应节点。 ○ 不支持配置**出错自动重跑**。

- **出错自动重跑**

  出错自动重跑（即任务运行失败后会自动触发重跑）的配置参数说明如下表所示。

参数	描述

参数	描述
重跑次数	周期任务调度执行失败的情况下，默认自动重跑的次数。 重跑次数最少配置为1（即任务出错后自动重跑1次），最多配置为10（即任务出错后会自动重跑10次）。您可以根据业务需要进行修改。
重跑间隔	默认每次重跑的间隔为30分钟，最小支持设置为1分钟，最大支持设置为30分钟。

> 说明
> - 当重跑属性设置为**运行成功或失败后皆不可重跑**时，则不会显示**出错自动重跑**属性，即任务出错不会自动重跑。
> - 您可以在**调度配置**页面，设置工作空间级别的默认重跑次数和重跑间隔。详情请参见调度设置。
> - 任务执行时，超过了超时时间导致的节点失败，自动重跑配置将不生效。

生效日期

调度节点在有效日期内生效并自动调度，超过有效期的任务将不会自动调度。此类任务为过期任务，您可以在运维大屏查看过期任务数量，并根据情况对其做下线等处理。查看运维大屏，详情请参见查看运维大屏

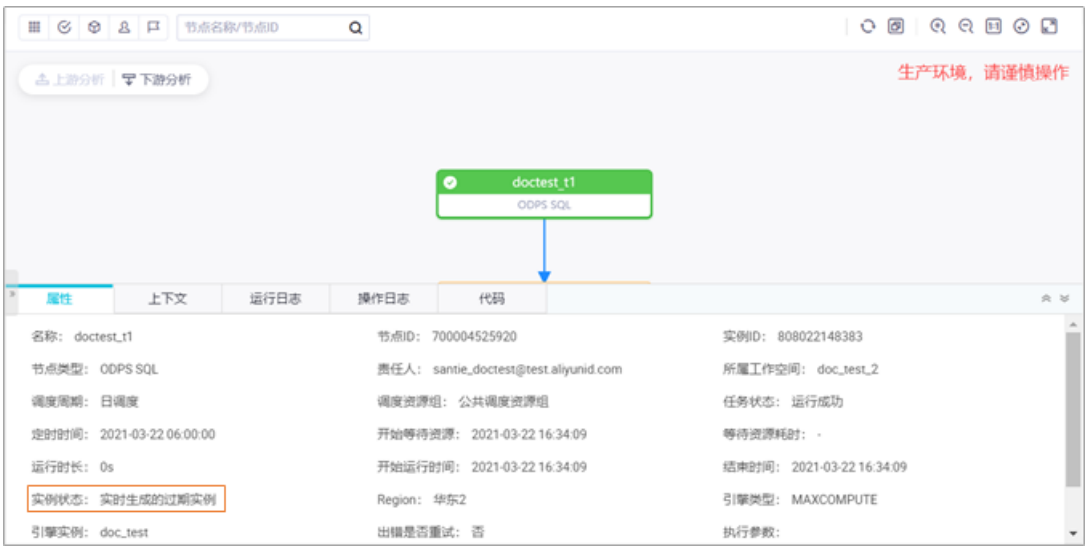
9.3.2. 实例生成方式：发布后即时生成实例

DataWorks会为您的节点自动生成实例，用于运行实例任务等操作。您可以在调度配置中指定实例在T+1次日生成或即时生成。指定发布后即时生成时，系统会立即生成实例，完成后您可进入运维中心查看实例生成状态。本文为您介绍即时转实例的规则及配置使用要点。

发布后即时生成实例规则

- 节点提交发布的时间早于23：30（23:30~24:00点期间为全量转实例期间）时，则DataWorks会即刻为您生成实例。
  - 任务定时运行时间在提交发布时间的10分钟之后（例如提交发布时间是18：00，定时运行时间是18：30），则DataWorks会正常生成实例运行任务，您可以在实例列表中找到对应的实例。

- 任务定时运行时间距离提交发布时间不足10分钟（例如提交发布时间是18：00，定时运行时间是18：05），则DataWorks会生成一个已完成的实例，节点实例状态为**实时生成的过期实例**，过期实例不会真实跑数据。



- 节点提交发布的时间晚于23：30（23:30~24:00点期间为全量转实例期间）时，则发布后即时生成实例功能不生效，您在实例列表中找不到对应的实例，需要等到提交发布后的第三天实例才生效。
- 节点如果是孤立节点时（即节点的调度依赖中，没有任何依赖的上游节点），则无法正常生成实例。

  例如，新增节点时，上游节点的**实例生成方式**配置为**T+1次日生成**，下游节点的**实例生成方式**配置为**发布后即时生成**时，下游节点实例即时生成后，上游节点实例还未生成，此时下游节点无法根据依赖关系找到上游节点实例，会变成孤立节点，无法自动调度。
- 已有的实例从**T+1次日生成**变更为**发布后即时生成**，会影响当天实例的产生情况。
  - 任务的定时时间（t1）在修改发布时间点（t2）的十分钟后：已生成的节点实例会被删除并替换为实时转出的实例。
  - 任务的定时时间（t1）是不在修改发布时间点（t2）的十分钟后：已生成的节点实例会保留。
- 组合节点（例如PAI节点、do-while节点、for-each节点等包含内部节点逻辑的节点）不支持将**实例生成方式**配置为**发布后即时生成**，即不支持实时转实例功能。

实例生成方式

目前有以下两种实例生成方式：

- **T+1次日生成**

  全量任务转实例：23：30之前提交发布的任务，第2天实例生效。23：30~00：00提交发布的任务，第3天实例生效。
- **发布后即时生成**

  实时转实例：发布代码即生成实例，生成实例时需满足实例生成规则，详情可参见发布后即时生成实例规则。

> **说明** 定时时间在发布时间点10分钟后的任务才会真实跑数据、产出数据。

常见使用场景：上游节点实例为T+1次日生成，下游节点实例为即时生成

实时转实例使用场景通常为上游节点实例的生成方式配置为T+1次日生成，下游节点实例的生成方式配置为发布后即时生成。以下图为例，上下游节点间的依赖关系如图所示。

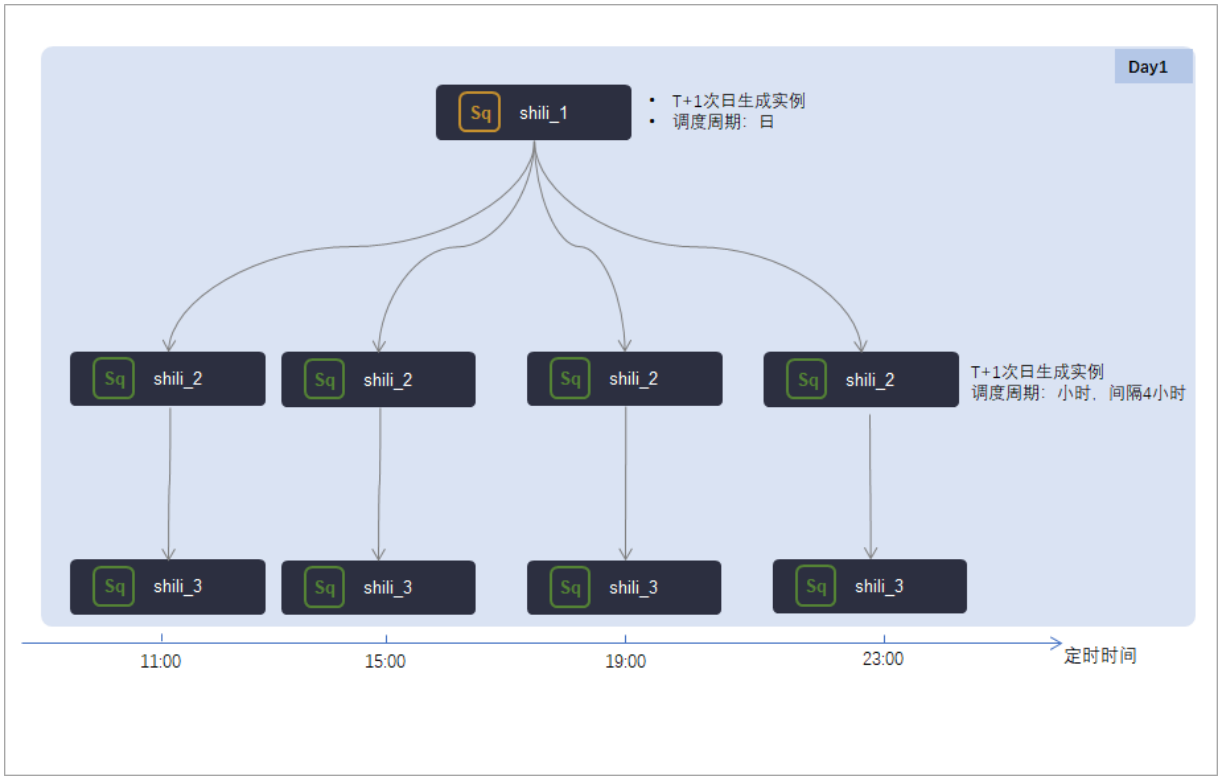
由于上游节点发布后第二天才会生成实例，而下游节点的实例为提交发布节点后即时生成，所以下游节点提交发布后，下游实时转实例任务是否正常执行，取决于上游节点当天的实例是否已经存在。细分场景及影响如下表所示。

细分场景	调度运行的影响	总结
上下游节点均为当天新增节点。 即下游节点提交发布时，上游节点的实例还未生成。	• 下游节点即刻产生实例，而上游节点实例在第2天才会产生，因此当天提交发布上下游节点后，下游节点的实例会变成孤立节点。 • 如果下游节点的**调度配置**中勾选了**依赖上一周期**，且**依赖项**选择为**本节点**，则即使第2天上游节点的实例成功产生了，但是下游节点由于跨周期自依赖到前1天的产生的下游上，从而导致整个任务被孤立且不被调度运行。	建议您修改上游节点的实例生成方式为**发布后即时生成**，则所有实例可正常生成，任务可正常调度运行。

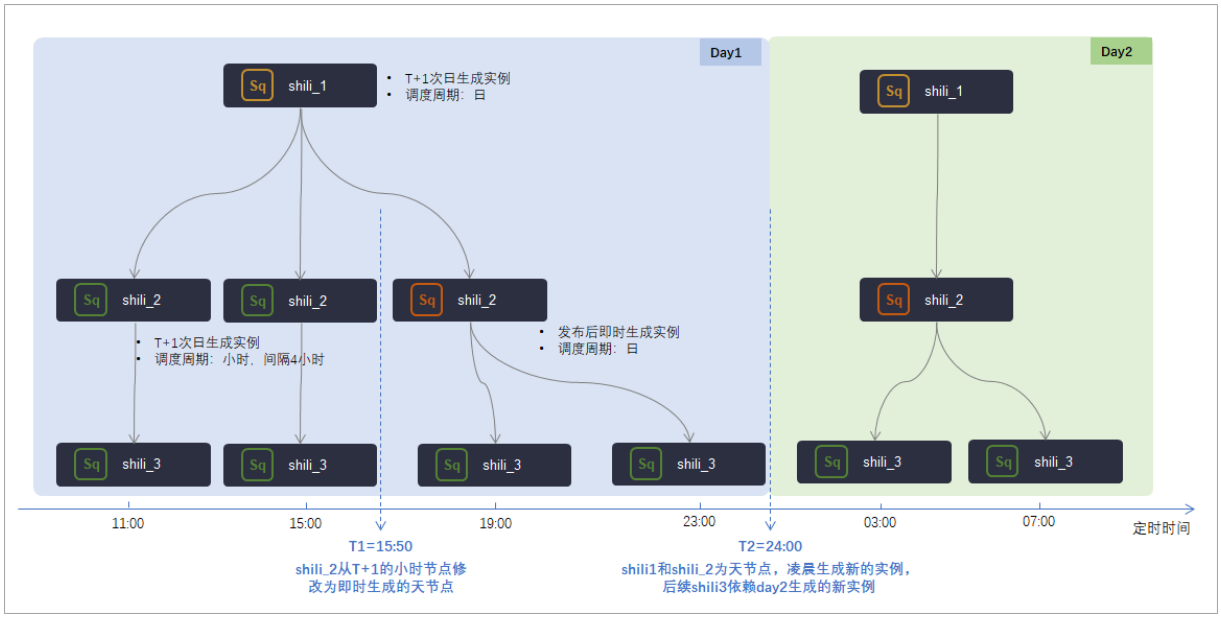
上游实例已经产生，下游节点为新增实时转实例的节点。
即下游节点提交发布时，上游节点已有成功生成的实例。
	• 下游节点在提交发布后即刻开始产生实例，产生的实例会依赖此前已有的上游节点的实例。
	• 如果下游节点的**调度配置**中：
	  ◦ **调度周期**为**小时**或**分钟**时，产生的下游节点的实例会依赖此前已有的上游节点的实例。
	  ◦ 勾选了**依赖上一周期**，且**依赖项**选择为**本节点**，则下游节点生成的第一个实例没有自依赖的上游实例，后续生成的实例可正常自依赖上一调度周期生成的实例，任务执行无影响。
		任何跨周期自依赖的调度是否成立，都需要以前一天该节点是否可以正常调度运行作为依据。

常见使用场景：节点从T+1次日生成实例的小时节点变更为即时生成实例的天节点

已提交发布的业务流程中，上游节点的实例为T+1次日生成，且调度周期为日，如下图Day1中的shili_1；下游节点的实例为T+1次日生成，调度周期为小时，如下图中Day1中的shili_2和shili_3。



当天节点实例已经正常产出并运行，运行一段时间后，在T1时刻将shili_2节点的调度周期从小时调度改为天调度，并且选择发布后及时生成实例。场景如下图所示。



变更后：

- 如上图所示，在T1所示时间点修改节点shili_2并提交发布后，则原本已生成的、用于T1时间点之后任务运行的小时节点实例会被删除，并且产生1个新的天节点实例，即提交发布前，shili_2为小时节点，发布后为天节点。
- 在T1提交发布后，shili_2节点的下游节点shili_3会依赖到新产生的天节点上。

- 如果shili_2的**调度配置**中勾选了**依赖上一周期**，且**依赖项**选择为**本节点**，则变更后，新产生的第一个天节点实例会依赖上个运行周期里的小时节点实例。

9.3.3. 调度周期：分钟调度

分钟调度即每天指定的时间段内，调度任务按`N*指定分钟`的时间间隔运行一次。

使用限制

分钟调度的时间间隔最小粒度只能设置为*5*分钟。

配置示例

- **配置路径**

  您需要进入数据开发节点的编辑页面，单击右侧导航栏的**调度配置**，在**调度配置 > 时间属性**区域配置节点的调度周期。

- **场景示例**

  目标任务每天`00:00 ~ 23:59`的时间段内，每隔5分钟调度一次，配置详情如下图所示。

  > **说明** corn表达式会根据您选择的时间自动生成，不可手动修改。

9.3.4. 调度周期：小时调度

小时调度即每天指定的时间段内，调度任务按`N*1小时`的时间间隔运行一次。例如，每天00:00~03:00的时间段内，每1小时运行一次。

使用限制

DataWorks仅支持配置调度任务在整点的时间段内进行小时调度。例如，您无法配置在00:05~23:59的时间段内，每隔1个小时运行一次。

注意事项

- 时间周期根据左闭右闭原则计算。例如，配置调度任务在0点~3点的时间段内，每隔1个小时运行一次。表示时间区间为[00:00，03:00]，间隔为1小时，调度系统每天将生成4个实例，实例定时时间分别在0点、1点、2点和3点，即0点、1点、2点和3点是实例的定时运行时间。
- 您可以设置在每天指定的时间段内，节点按指定时间间隔运行一次；也可以选择多个时间点，设置在每天指定的时间点运行。

- 周期调度配置的时间点为定时时间，任务实际运行时，可能会因为等待资源等原因导致实际运行时间与定时时间不一致。

配置示例

- **配置路径**

  您需要进入数据开发节点的编辑页面，单击右侧导航栏的**调度配置**，在**调度配置 > 时间属性**区域配置节点的调度周期。

- **场景示例**
  - **配置详情**

    目标任务每天 `00:00` ~ `23:59` 的时间段内，每隔6小时自动调度一次，配置详情如下图所示。

    > **说明** corn表达式会根据您选择的时间自动生成，不可手动修改。

  - **调度详情**

    调度系统每天将生成4个实例，并在实例的定时时间0点、6点、12点和18点运行，如下图所示。

9.3.5. 调度周期：日调度

日调度即调度节点每天在指定的定时时间运行一次。新建周期任务时，日调度默认的时间周期为每天0点运行一次。您可以根据需要自行指定运行时间点，例如，指定每天13点运行一次。

配置示例

- **配置路径**

  您需要进入数据开发节点的编辑页面，单击右侧导航栏的**调度配置**，在**调度配置 > 时间属性**区域配置节点的调度周期。

- **场景示例**

- **配置详情**
  - 假设导入、统计加工和导出任务，均为日调度任务。
  - 上述任务的运行时间为每天 `13:00` 点。
  - 统计加工任务依赖导入任务，导出任务依赖统计加工任务（即统计加工任务的依赖属性配置上游任务为导入任务）。

  根据上述场景，日调度任务的配置详情如下图所示。

  > **说明** corn表达式会根据您选择的时间自动生成，不可手动修改。

- **调度详情**

  调度系统会自动为任务生成实例并运行，如下图所示。

  > **说明**
  > - 调度节点执行需要满足如下条件：
  >   - 上游任务执行成功。
  >   - 节点的定时运行时间已到。
  >
  >   任何一个条件不满足，调度节点都无法执行，并且两个条件没有先后顺序。
  > - 默认调度时间是在 `00:00` ~ `00:30` 时间段随机生成。

9.3.6. 调度周期：周调度

周调度即调度任务每周的特定几天，在特定时间点自动运行一次。

注意事项

在非指定的调度时间内，为保证下游实例正常运行，节点会生成空跑的周期实例。即调度到该任务时，系统会直接返回成功，不会真正执行，也不会阻塞下游节点执行及占用资源。

配置示例

- **配置路径**

  您需要进入数据开发节点的编辑页面，单击右侧导航栏的**调度配置**，在**调度配置 > 时间属性**区域配置节点的调度周期。

- **场景示例**
  - **配置详情**

    目标任务配置在每周一、周五两天定时运行，则在周一、周五生成的实例会正常的调度执行，而周二、周三、周四、周六以及周日5天都是生成实例后直接设置为运行成功，配置详情如下图所示。

    > **说明** corn表达式会根据您选择的时间自动生成，不可手动修改。

  - **调度详情**

    调度系统会自动为任务生成实例并运行，如下图所示。

9.3.7. 调度周期：月调度

月调度即调度任务在每月的特定几天，在特定时间点自动运行一次。

注意事项

- 在非指定的调度时间内，为保证下游实例正常运行，节点会生成空跑的周期实例。即调度到该任务时，系

统会直接返回成功，不会真正执行，也不会阻塞下游节点执行及占用资源。

- 月调度支持将**指定时间**配置为**每月最后一天**，则每调度任务会在每个月的最后一天运行。

配置示例

- **配置路径**

  您需要进入数据开发节点的编辑页面，单击右侧导航栏的**调度配置**，在**调度配置 > 时间属性**区域配置节点的调度周期。

- **场景示例**
  - **配置详情**

    目标任务配置在每月最后一天运行。则每月最后一天生成的实例会正常的调度执行，其它日期每天都是生成空跑实例并直接设为运行成功，配置详情如下图所示。

    > **说明** corn表达式会根据您选择的时间自动生成，不可手动修改。

  - **调度详情**

    调度系统会自动为任务生成实例并运行，如下图所示。

9.3.8. 调度周期：年调度

年调度即调度任务在每年的特定几天，在特定时间点自动运行一次。

注意事项

在非指定的调度时间内，为保证下游实例正常运行，节点会生成空跑的周期实例。即调度到该任务时，系统会直接返回成功，不会真正执行，也不会阻塞下游节点执行及占用资源。

配置示例

- **配置路径**

您需要进入数据开发节点的编辑页面，单击右侧导航栏的**调度配置**，在**调度配置 > 时间属性**区域配置节点的调度周期。

- **场景示例**
  - **配置详情**

    目标任务配置在每年的一月、四月、七月、十月的1日和最后一日运行。则在上述指定日期生成的实例会正常的调度执行，其它日期每天都是生成空跑实例并直接设为运行成功，配置详情如下图所示。

    > **说明** corn表达式会根据您选择的时间自动生成，不可手动修改。

  - **调度详情**

    调度系统会自动为任务生成实例并运行，如下图所示。

9.4. 配置资源属性

周期任务的运行依赖于调度资源组，您可以在目标任务调度配置的资源属性区域，选择任务调度运行时需要使用的资源组。

配置资源组

进入数据开发节点的编辑页面，单击右侧导航栏的**调度配置**，在**资源属性**区域配置节点调度时使用的资源组。

1. 在**调度资源组**下拉框，选择任务调度时需要使用的资源组，默认配置为**公共调度资源组**。

   > **说明** 您可以进入**调度设置**页面，配置调度任务默认使用的资源组，详情请参见调度设置。

2. 单击所选资源组后方的**查看水位**，即可查看该资源组前一天不同时间段的使用率。您可以根据各个资源组的使用情况，为调度任务合理分配资源组。

> **说明**
> - 使用公共调度资源组时，高峰期任务排队风险较高，建议您调整任务定时时间以错峰调度或使用独享调度资源组。
> - 如果当前没有满足您需求的资源组，则可以单击**购买调度资源组**新增。新增独享调度资源组，详情请参见新增和使用独享调度资源组。

9.5. 配置调度依赖

9.5.1. 同周期调度依赖逻辑说明

DataWorks通过各个节点的调度依赖配置结果，有序的运行业务流程中各个节点，保障业务数据有效、适时地产出。本文为您介绍DataWorks的调度依赖实现流程与主要配置原则。

为什么要配置调度依赖

调度依赖就是节点间的上下游依赖关系，在DataWorks中，上游任务节点运行完成且运行成功，下游任务节点才会开始运行。

配置调度依赖后，可以保障调度任务在运行时能取到正确的数据（当前节点依赖的上游节点成功运行后，DataWorks通过节点运行的状态识别到上游表的最新数据已产生，下游节点再去取数）。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。

配置节点的调度依赖时，建议根据各个节点的表数据血缘关系来规划配置节点的上下游依赖，确保满足以下原则：

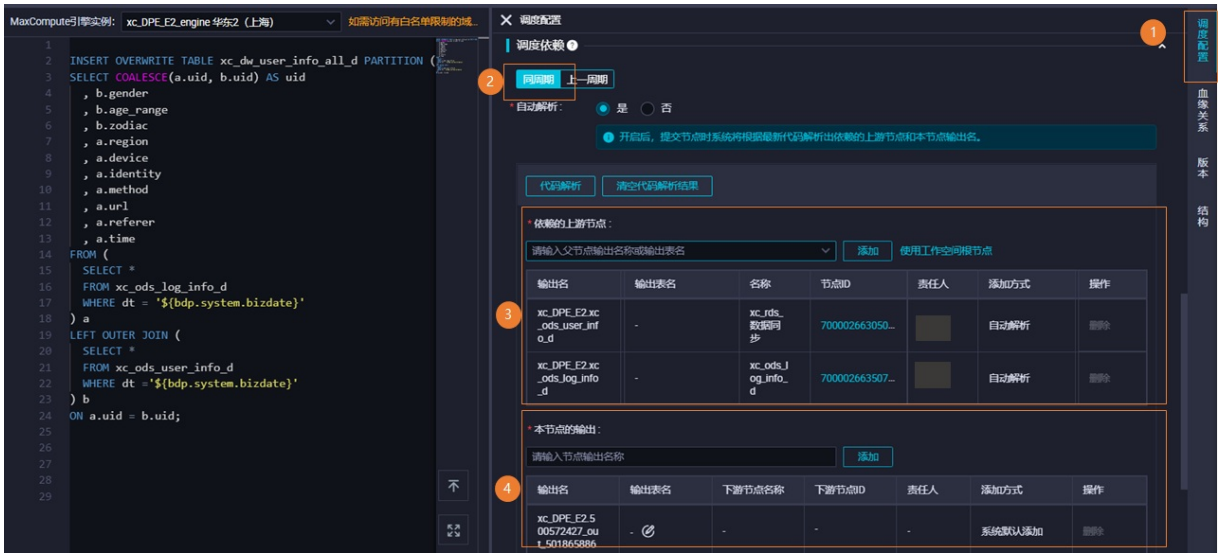
- 一张表的数据只由一个节点产出，且节点的产出表需配置为本节点的输出。

  > **说明**
  > - SQL任务会通过自动解析，将产出表作为本节点输出，无需手动配置。
  > - 离线同步任务需要手动配置，将产出表添加为本节点输出，格式为projectname.tablename。以便下游节点对该表进行数据清洗时，可以通过自动解析快速设置同步任务节点依赖关系。

- 上游节点的输出作为下游节点的输入，形成节点间的依赖关系。

> **说明** 对于没有表数据血缘关系的节点，可以根据节点运行的逻辑上下游关系规划配置节点的依赖关系，配置原则和配置后的结果与有血缘关系的节点一致。

DataWorks的调度依赖在各个节点的**调度配置**中进行配置，每个节点需要为其配置**依赖的上游节点**和**本节点的输出**。



DataWorks支持自动解析和手动配置的方式进行调度依赖配置。

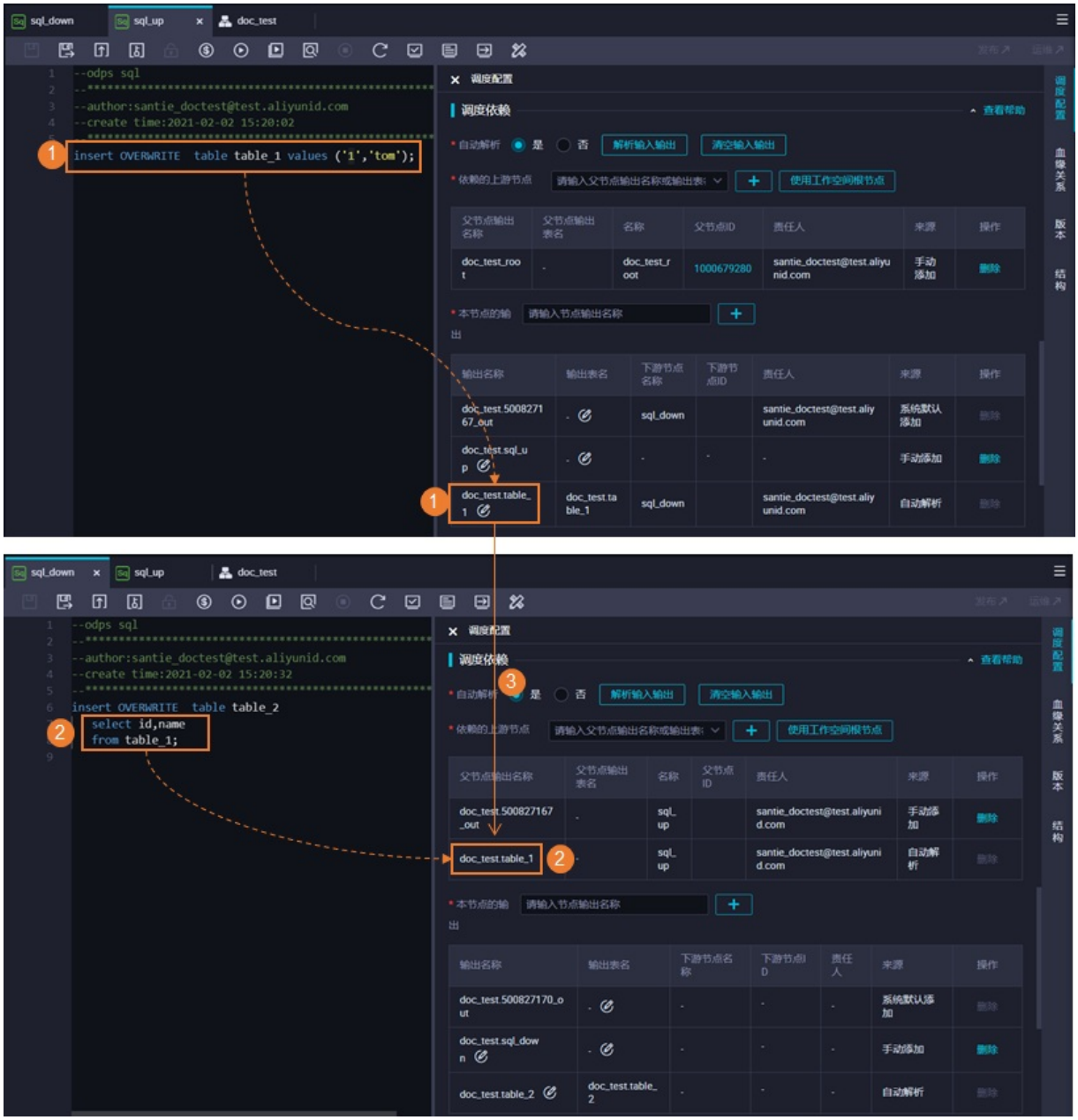
- 理想状态下，DataWorks可根据您规范化的节点任务代码开发，识别输入输出命令（如`select`、`insert`），根据代码识别表数据的血缘关系，以血缘关系为基座，通过自动解析自动为您配置好节点的调度依赖。
- 特殊场景下，例如本地上传的表，表数据无需周期性调度生成数据时，您可以手动增删节点的调度配置。

在提交节点时，DataWorks会检查节点的调度依赖与节点代码中的数据血缘关系是否一致，如果出现不一致的提示，您需要根据实际情况查看是否需要修改调度依赖配置。

自动解析

对于SQL类的节点，DataWorks可根据节点中的任务代码，自动解析出当前节点的上下游依赖关系，并自动为您在节点的调度配置中添加好对应的**本节点的输出**或**依赖的上游节点**。

自动解析添加调度依赖的原理



- 节点代码中存在输出命令时，如`insert`、`create`命令，DataWorks会自动解析，将输出表添加至节点的**本节点的输出**。
- 节点代码中存在输入命令时，如`select`命令，DataWorks会自动解析，将输入表添加至节点的**依赖的上游节点**。
- 将上游节点的输出添加为下游节点的输入，以表数据的血缘上下游关系为基座，形成节点间的上下游依赖关系。

自动解析的结果原则上与表数据的血缘关系是一致的，您在提交节点时，DataWorks会检测节点的调度依赖配置结果与表数据血缘关系是否一致，当出现不一致提示时，您需要根据实际情况选择处理方式：

- 节点中没有查询非周期性产生数据的表时，您需要检查调度依赖配置是否正确。
- 节点有查询非周期性产生数据的表时，您需要通过手动配置的方式，删除此依赖。

代码开发要求与原则

自动解析完全依据您的任务节点中代码自动识别，因此您在进行数据开发时，建议严格遵循DataWorks的代码开发要求和节点创建要求：

- 代码开发要求：一张表数据由一个节点产出，一个节点只产出一张表。
- 节点创建要求：建议节点名称与产出表的表名称保持一致。
- 调度配置要求：节点的产出表需配置为本节点的输出。

手动配置

DataWorks支持在节点的代码开发过程中，手动修改节点的**依赖的上游节点**、**本节点的输出**。当通过自动解析生成的节点调度依赖配置与实际应用不符时，您可通过手动配置进行修改。

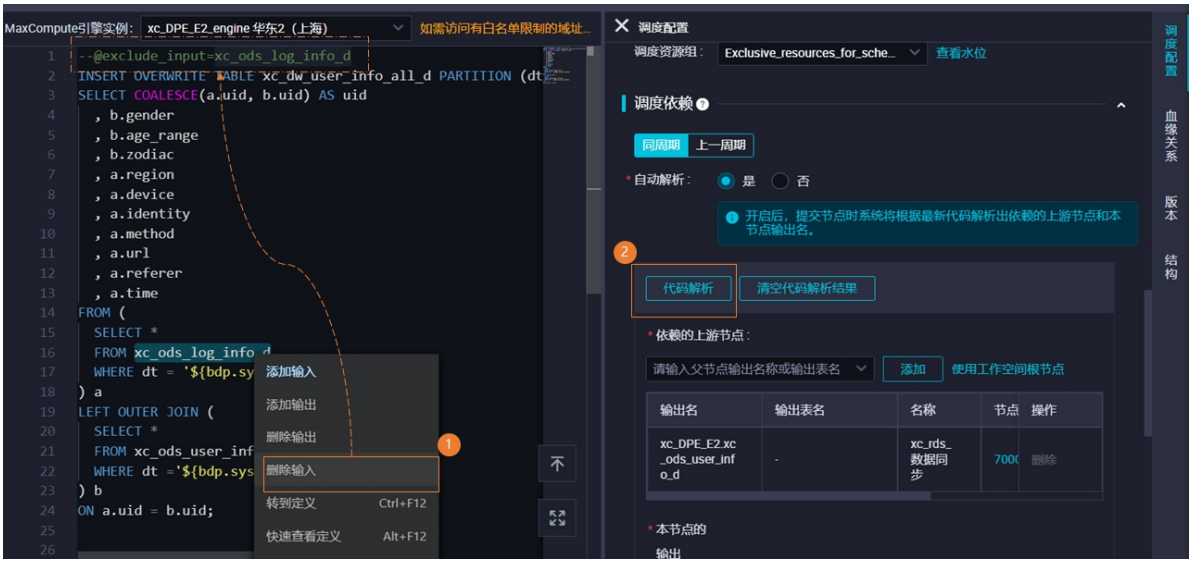
应用场景

由于DataWorks的调度依赖主要保障的是调度节点定时更新的表数据，通过节点调度依赖保障下游取数没有问题，所以不是DataWorks平台上调度更新的表，平台无法监控。当存在非周期性调度生产数据的表，有节点select这类表数据时，您需要手动删除通过select自动生成的依赖的上游节点配置。非周期性调度生产数据的表包括：

- 从本地上传到DataWorks的表
- 维表
- 非DataWorks调度产出的表
- 手动任务产出的表

配置方式

- 代码编辑页面删除



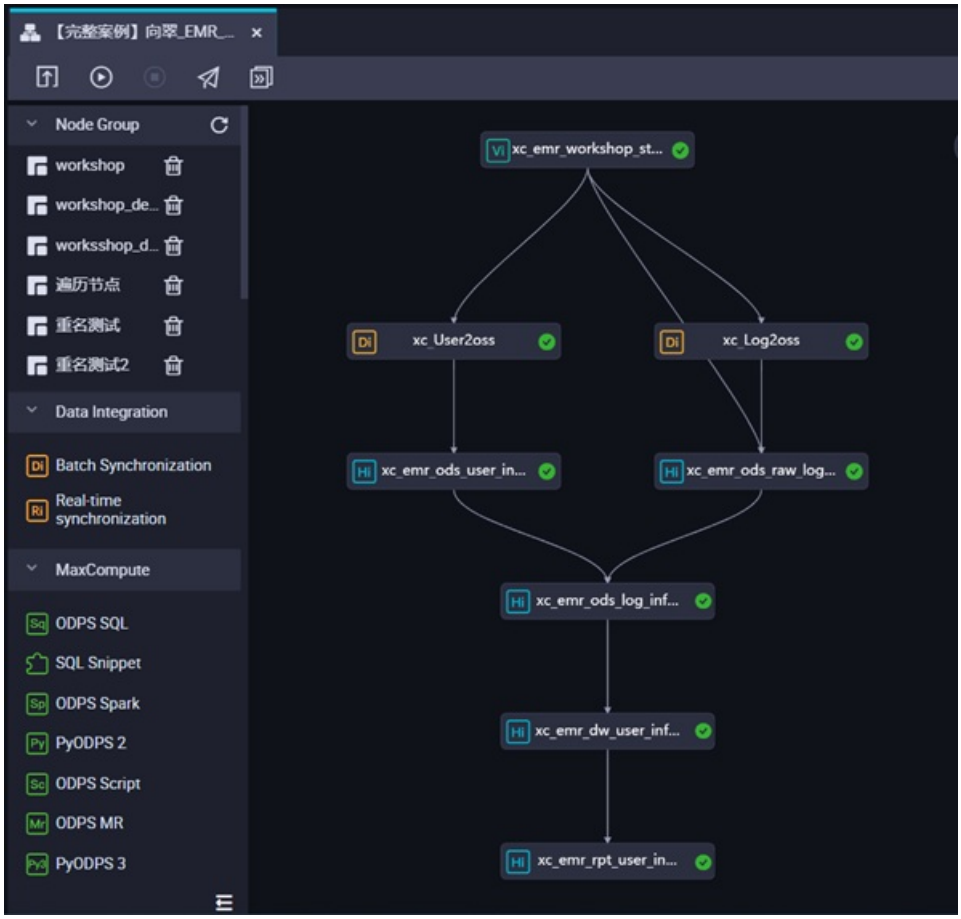
如上图所示，您可以在select了非周期性产出表的节点代码编辑页，右键相应的表名，进行删除输入的操作。您也可以在代码的最上方添加一条规则的注释，操作完成后自动解析将不会解析该依赖。

- 调度配置页面删除

如上图所示，您可以在select了非周期性产出表的节点调度配置的页面中，将自动解析开关选择为否，然后手动删除对应的**依赖的上游节点**。

拉线配置

DataWorks支持在业务流程的页面，直接通过连线的方式，指定各个节点的上下游关系。拉线完成后DataWorks根据您的拉线结果自动为您在各个节点中添加调度依赖配置。



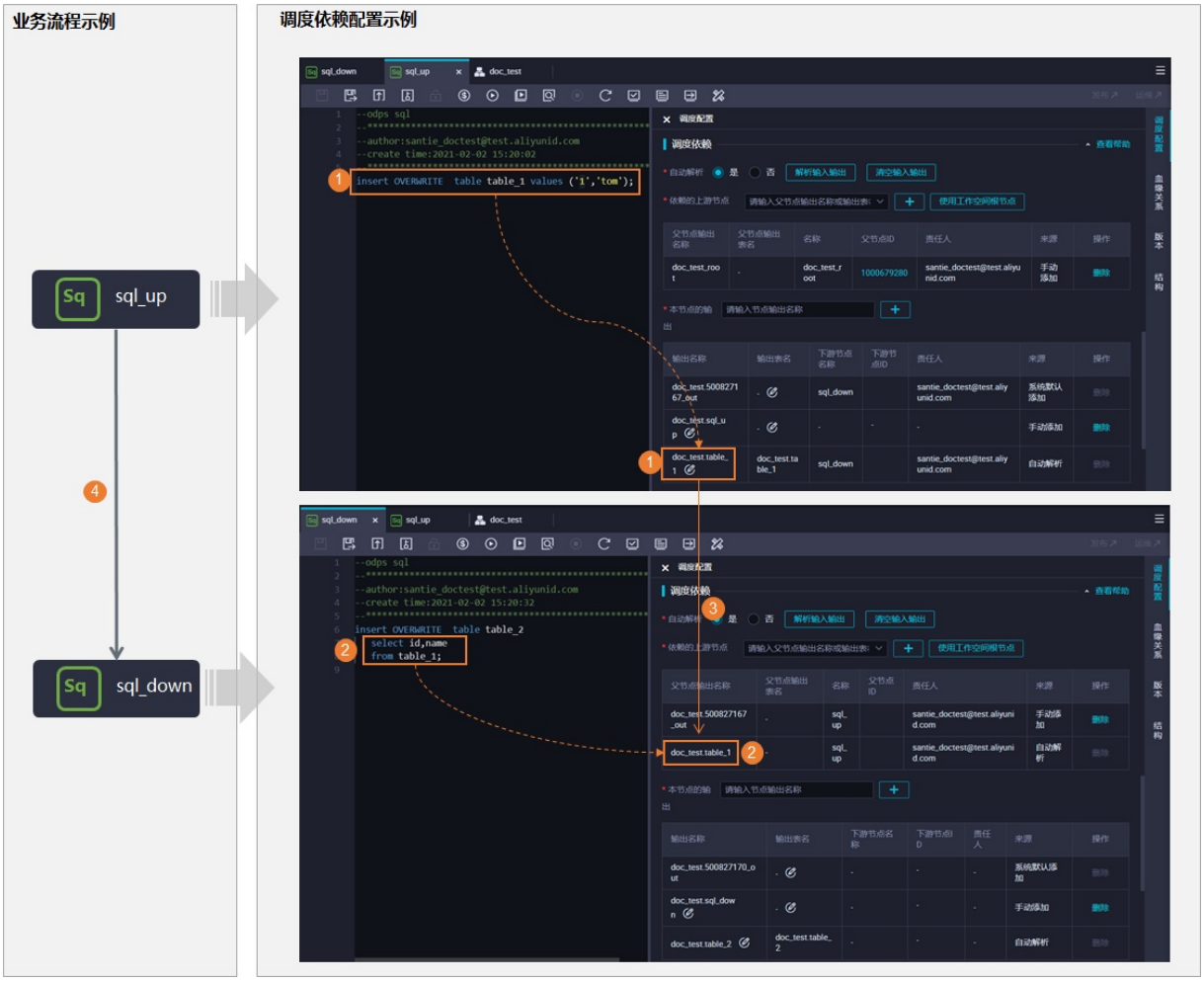
各个节点创建完成后，DataWorks会自动为所有节点添加一个名称后缀为_out的**本节点的输出**，拉线指定上下游依赖时，DataWorks将自动生成的后缀为_out的输出添加为下游节点的输入中。

适用场景

当您创建完成业务流程后，您可根据业务规划，将各个节点按照节点的逻辑顺序，在业务流程页面通过拉线的方式配置好各个节点的依赖关系。后续在代码开发过程中，通过自动解析和手动修改的方式添加或修改各个节点的依赖关系，保障整体业务流程中所有节点的依赖关系是正确的。

案例说明

下面以一个具体案例为例，说明调度依赖配置后的实现流程与原理。



如上图所示：

- 节点输出需要添加到节点的**本节点的输出**。有输出命令时，例如 `insert` 命令时，DataWorks会自动解析，将输出表添加至节点的**本节点的输出**。
- 节点输入需要添加到节点的**依赖的上游节点**。有输入命令时，例如 `select` 命令时，DataWorks会自动需要将输入表添加至节点的**依赖的上游节点**。
- 将上游节点的输出添加为下游节点的输入，以表数据的血缘上下游关系为基座，形成节点间的上下游依赖关系。

上下游关系形成后，后续在业务流程周期性调度的过程中，会根据上下游顺序，优先运行上游节点。上游节点运行完成且运行成功后，开始启动下游节点运行。

通过上述流程可见，各个节点进行依赖调度配置的关键配置原则：

- 有上下游关系的节点，上游节点的**本节点的输出**一定要作为下游节点的**依赖的上游节点**，形成节点间的上下游依赖关系。
- 下游节点的**依赖的上游节点**中的**父节点输出名称**和**父节点ID**要唯一（从另一方面也就要求了任一节点的节点输出名称要唯一），否则下游节点无法通过这两个信息找到正确的上游节点，取用上游节点数据。

调度依赖配置指导

通用场景的调度依赖配置可参见配置同周期调度依赖。

其他典型场景的调度依赖配置可参见：

- 场景1：包含离线同步节点的业务流程，如何配置调度依赖
- 场景2：依赖上一周期的结果时，如何配置调度依赖
- 场景3：如何配置跨业务流程、跨工作空间的调度依赖

常见问题

- 提交节点报错：当前节点依赖的父节点输出名不存在
- 提交节点时提示：输入输出和代码血缘分析不匹配
- 依赖关系

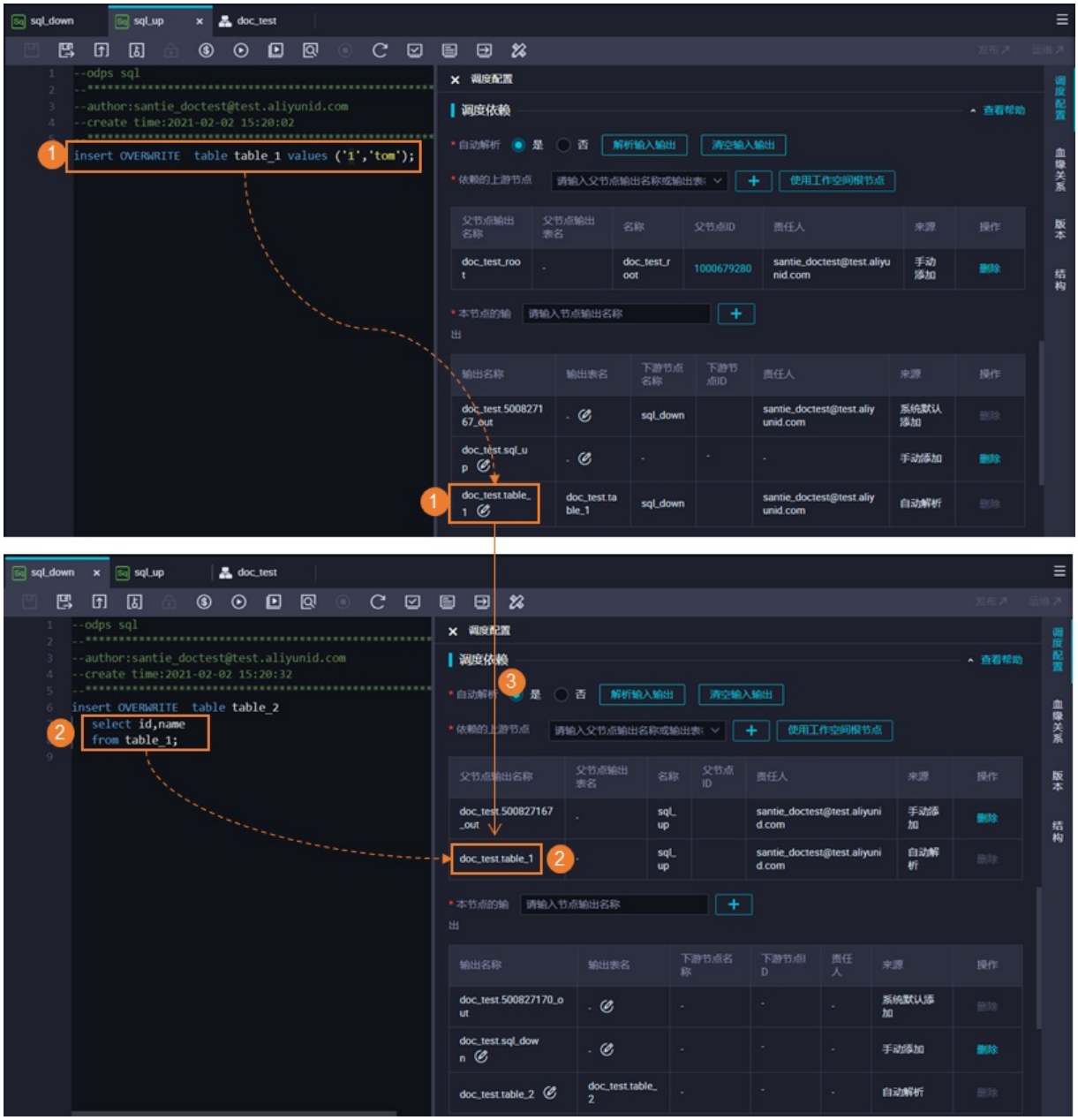
9.5.2. 配置同周期调度依赖

调度依赖关系是您构建有序业务流程的根本，只有正确构建任务依赖关系，才能保障业务数据有效、适时地产出。本文为您介绍调度依赖的配置指导。

背景信息

调度依赖部分的配置主要包含**依赖的上游节点**和**本节点的输出**，具体说明如下：

- **依赖的上游节点**：定义了本节点的上游节点，在上游节点运行成功后本节点才具备运行条件。
- **本节点的输出**：作为其他节点与本节点建立依赖关系的媒介，您可以通过输出名找到目标节点并将其设置为依赖的上游节点。



DataStudio中，为周期任务配置依赖关系的方式包括：

- 在业务流程画布中，使用鼠标拖拽的方式在节点之间进行连线。详情请参见调度依赖配置指导：鼠标拖拽。
- 在调度配置面板中，搜索节点名称或者表名称，手动将某个节点添加为当前节点的上游依赖。您也可以借助**代码解析**功能，并基于系统的解析结果进行修改。详情请参见调度依赖配置指导：手动配置。
- 在调度配置面板中，开启自动解析功能，提交任务时系统会根据代码解析出本节点的上游依赖，并自动完成添加。详情请参见调度依赖配置指导：自动解析。
- 在调度配置面板中，使用自动推荐功能。系统会根据表的血缘信息推荐上游依赖节点。

> **说明** 表的血缘信息更新时间为 `t+1` 天，可能存在推荐上游节点滞后的情况。

配置完成后，您可以预览该节点的依赖关系，详情请参见预览依赖关系。

常见的典型配置可参见典型业务场景配置指导。

调度配置完成后，提交节点时，DataWorks会检查节点的调度依赖与节点代码中的数据血缘关系是否一致，详情可参见调度配置完成后处理。

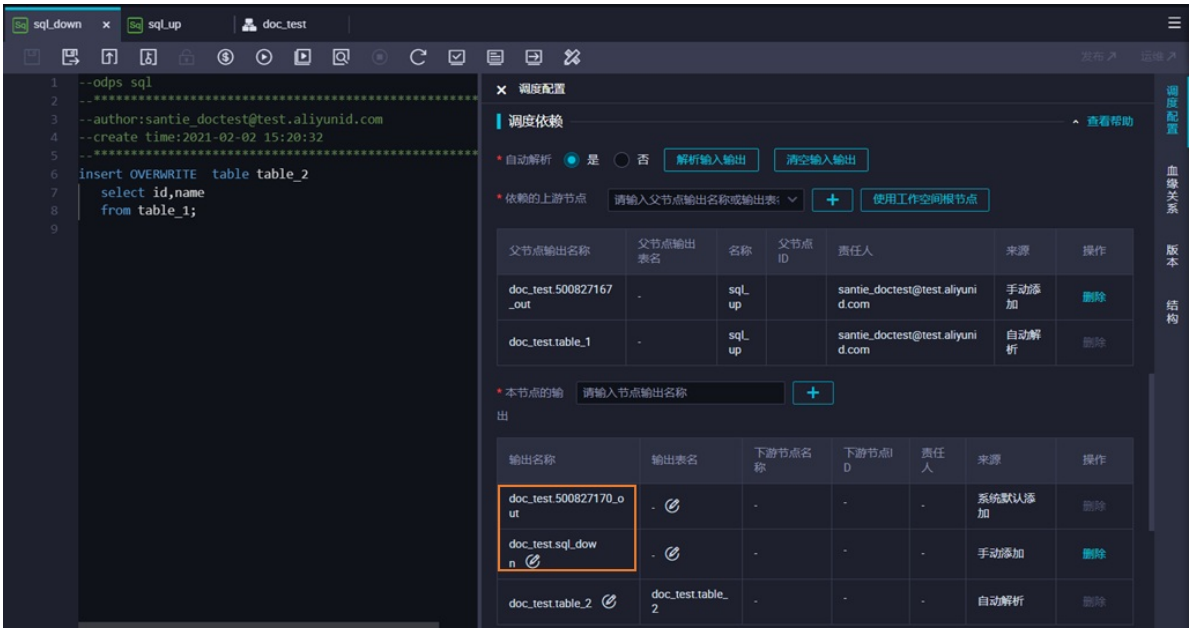
通用配置原则

DataWorks的调度依赖配置包括**依赖的上游节点**和**本节点的产出**：

- **本节点的输出**指当前节点的输出，输出节点名称是全局唯一的，在整个阿里云账号内不允许重复。

  创建节点后，DataWorks自动为各节点生成两个**本节点的输出**的配置信息，其中：

  - 一个**本节点的输出**名称的后缀为*******_out**
  - 一个**本节点的输出**名称为 `projectname.nodename`



- **依赖的上游节点**指当前节点依赖的上游节点，配置后，DataWorks会通过配置的上游节点输出名或输出表名进行找到依赖的上游节点。

  如果您通过手动搜索上游输出名添加，则搜索器会根据已提交至调度系统中的节点的输出名来进行搜索。

  搜索支持模糊匹配，即输入关键词，即可显示所有包含关键词的节点。当节点显示**对应节点目前被冻结**时，请勿使用该节点作为依赖的上游节点，以免影响任务的正常运行。

DataWorks支持自动解析和手动配置的配置方式，无论使用哪种方式配置依赖关系，调度配置的总逻辑不变：

- 一张表的数据只由一个节点产出，且节点的产出表需配置为本节点的输出。

  > 说明
  > - SQL任务会通过自动解析，将产出表作为本节点输出，无需手动配置。
  > - 离线同步任务需要手动配置，将产出表添加为本节点输出，格式为projectname.tablename。以便下游节点对该表进行数据清洗时，可以通过自动解析快速设置同步任务节点依赖关系。

- 上游节点的输出作为下游节点的输入，形成节点间的依赖关系。

更多节点的调度依赖的逻辑原理说明，可参见同周期调度依赖逻辑说明。下文为您详细介绍调度依赖的原理及配置方式。

> **说明** 2019年1月10日之前创建的工作空间，存在数据问题，需要提交工单申请修改。2019年1月10日之后创建的工作空间，则不受影响。

调度依赖配置指导：自动解析

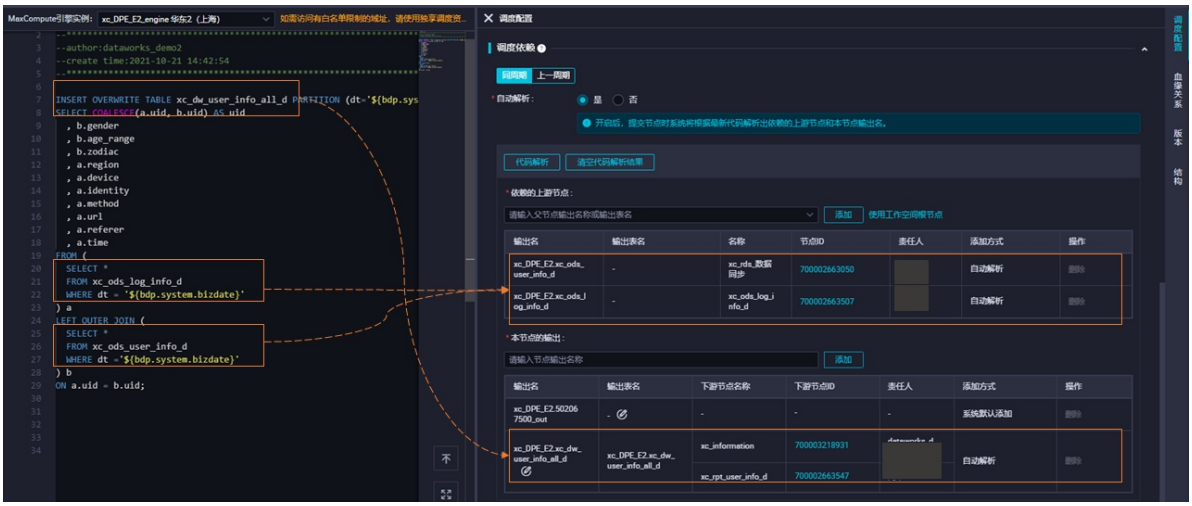
- 应用场景

  DataWorks支持根据任务节点中的SQL命令，自动解析出表数据的血缘关系，以表数据的血缘关系为基座，为节点自动添加**本节点的输出**或**依赖的上游节点**，自动解析高效便捷，适用于绝大部分场景。

> **说明** 离线同步任务暂不支持自动解析设置节点依赖关系。当同步任务产出一张表时，您需要手动将添加为节点的输出，格式为projectname.tablename。以便下游节点对该表进行数据清洗时，可以通过自动解析快速设置节点依赖关系。

- 实现原理

  下图为自动解析依赖关系的原理。



  - SELECT一张表，该表将自动解析为本节点依赖的上游。
  - INSERT一张表，该表将自动解析为本节点的输出。

  如果出现的多个INSERT、SELECT，则会自动解析出多个输出、输入名称。

- 配置方法

  自动解析通过SQL代码命令自动识别配置，无需您手动配置。自动配置的原则如下表所示。

节点类型	代码命令	自动解析	调度依赖配置规则
ODPS节点	- CREATE - INSERT	当节点代码中出现此类输出命令时，会自动为节点添加一条**本节点输出**配置内容。	为节点自动添加的**本节点输出**命名规则为：*odps_project_name.table_name*。 其中： *odps_project_name*：为当前节点所在的DataWorks项目名称。 *table_name*：为输出命令后的表名称。
	SELECT	当节点代码中出现此命令时，会自动为节点添加一条**依赖的上游节点**配置内容。	为节点自动添加的**依赖的上游节点**命名规则为：*project_name.table_name*。 其中： *project_name*：SELECT语句中，from命令后的表所在节点的项目名称。 *table_name*：SELECT语句中from命令后的表名称。

节点类型	代码命令	自动解析	调度依赖配置规则
非ODPS的SQL节点	○ ALTER ○ CREATE ○ UPDATE ○ INSERT	当节点代码中出现此类输出命令时，会自动为节点添加一条**本节点输出**配置内容。	各类型节点自动添加的**本节点输出**命名规则为： ○ EMR：*workspace_name.db_name.table_name* ○ ADBPG：*workspace_name.db_name.schema_name.table_name* ○ ADBMySQL：*workspace_name.db_name.schema_name.table_name* ○ Hologres：*workspace_name.db_name.schema_name.table_name* 其中： ○ *workspace_name*：为当前节点所在的DataWorks项目名称。 ○ *db_name*：为当前节点所在的数据库名称。 ○ *schema_name*：为当前节点的schema名称。 ○ *table_name*：为输出命令后的表名称。
	SELECT	当节点代码中出现此命令时，会自动为节点添加一条**依赖的上游节点**配置内容。	为节点自动添加的**依赖的上游节点**命名规则为：*project_name.table_name*。 其中： ○ *project_name*：SELECT语句中，from命令后的表所在节点的项目名称。 ○ *table_name*：SELECT语句中from命令后的表名称。
离线同步节点	离线同步节点不支持自动解析，需要手动添加节点的调度依赖配置。 **说明** 当同步任务产出一张表时，需要手动将添加为节点的输出，格式为projectname.tablename。以便下游节点对该表进行数据清洗时，可以通过自动解析快速设置节点依赖关系。		

- 注意事项
  - 代码开发要求

    自动解析完全依据您的任务节点中代码自动识别，因此您在进行数据开发时，建议严格遵循DataWorks的代码开发要求和节点创建要求：
    - 代码开发要求：一张表数据由一个节点产出，一个节点只产出一张表。
    - 节点创建要求：建议节点名称与产出表的表名称保持一致。
    - 调度配置要求：节点的产出表需配置为本节点的输出。
  - 不支持自动解析的场景
    - 离线节点、AnalyticDB for PostgreSQL节点、AnalyticDB for MySQL节点、EMR节点不支持通过自动解析添加节点的调度依赖，这些节点的产出表需要手动添加为本节点的输出。
    - SQL代码中的临时表（例如在工作空间配置中指定t_开头的表为临时表）不支持自动解析，不会被自动解析为本节点的输出或依赖的上游节点。

- 不规范使用的处理逻辑
  - 如果SQL语句中的一个表名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SQL语句中的一个表名被多次引用或被多次产出，则解析时只解析一个调度依赖关系。
- 提交节点时，出现调度依赖配置不一致情况

  开启自动解析后，为保障节点数据产出无误，提交节点时，系统将基于代码中表的血缘关系自动解析当前节点的输入与输出。您也可以根据实际需求修改节点的输入、输出结果。

  如果提交节点时，当前版本的调度依赖（自动解析结果+您在**调度配置 > 调度依赖**区域自行修改的输入输出结果）与开发环境或生产环境节点的调度依赖不一致时，将出现输入输出变更提示（当前最新版本与上个版本比较，新增或删除了哪些输入或输出）。您可以选择是否**使用新的解析结果**，基于当前节点最新版本的调度依赖继续提交该节点，节点提交后，最新的解析结果将自动添加至**调度配置 > 调度依赖**区域。

  > **说明** 节点提交时，若发现节点当前调度依赖解析与生产或开发环境节点调度依赖关系配置存在差异，请确认该节点当前的调度依赖是否符合业务需要，避免由于依赖关系变更导致产出数据出现问题。若当前节点存在众多下游任务时，可能会产生较大影响，请明确业务场景后再谨慎操作。
  >
  > 例如，当前节点提交时与生产环境该节点的调度配置比较，发现缺少了输入名A（即**上游节点输出名**为A），此时，您需要确认当前节点的调度依赖是否配置正常。若该节点代码中配置了依赖A表的数据，但未将产出A表数据的节点作为当前节点依赖的上游，则可能会出现A表数据未产出，当前节点便开始执行，最终导致当前节点产出的表数据出现问题。

- 其他注意事项

  节点的SQL命令中，如果查询了非周期性生成数据的表（例如维表、从本地上传到DataWorks的表等表），自动解析会将此表添加为本节点的**依赖的上游节点**。但是通过这个**依赖的上游节点**找不到生成此非周期性生成数据表的节点，会导致调度错误，您需要手动将自动解析出来的**依赖的上游节点**删除。

调度依赖配置指导：手动配置

- 应用场景

  DataWorks支持在节点的代码开发过程中，手动修改节点的**依赖的上游节点**、**本节点的输出**。当通过自动解析生成的节点调度依赖配置与实际应用不符时，您可通过手动配置进行修改。

  常见的应用场景包括：

  - 删除非周期性调度生产数据的表的自动解析配置结果

    由于DataWorks的调度依赖主要保障的是调度节点定时更新的表数据，通过节点调度依赖保障下游取数没有问题，所以不是DataWorks平台上调度更新的表，平台无法监控。当存在非周期性调度生产数据的表，有节点select这类表数据时，您需要手动删除通过select自动生成的依赖的上游节点配置。非周期性调度生产数据的表包括：

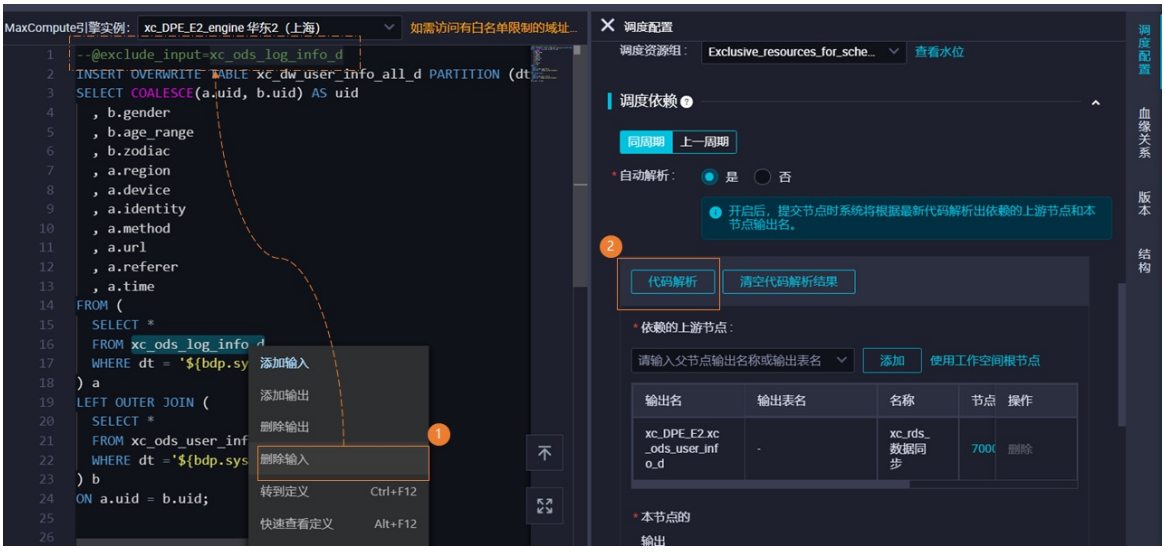
    - 从本地上传到DataWorks的表
    - 维表
    - 非DataWorks调度产出的表
    - 手动任务产出的表

  - 为不支持自动解析的部分节点，手动添加产出表为本节点的输出

    离线节点、AnalyticDB for PostgreSQL节点、AnalyticDB for MySQL节点、EMR节点不支持通过自动解析添加节点的调度依赖，这些节点的产出表需要手动添加为本节点的输出。

- 配置方法

  - 代码编辑页面删除

    如上图所示，您可以在select了非周期性产出表的节点代码编辑页，右键相应的表名，进行删除输入的操作。您也可以在代码的最上方添加一条规则的注释，操作完成后自动解析将不会解析该依赖。

- 调度配置页面删除

如上图所示，您可以在select了非周期性产出表的节点调度配置的页面中，将自动解析开关选择为否，然后手动删除对应的**依赖的上游节点**。

- 注意事项
  - 在调度配置页面配置时，将自动解析的开关选择为**否**后，建议您先单击**解析输入输出**，系统将使用与自动解析原理一致的方式，为您自动识别并添加好调度依赖关系，您可根据实际情况在此基础上进行删减操作。
  - 单击**清空输入输出**，您可以将自动解析识别添加的调度依赖关系一键删除，已手动添加的调度依赖关系不会被删除。
  - 单击**自动推荐**，系统将会基于本工作空间的SQL血缘关系，为您推荐产出当前节点输入表的其它所有SQL节点。您可以根据实际情况，选择推荐列表中的任务，配置为当前节点的依赖的上游节点。

> **说明** 由于需要提交发布至生产环境并真实产出该表数据的节点，才会被解析出来，所以自动解析推荐的节点有T+1的延迟。
>
> 被推荐节点需要在前一天提交至调度系统，第二天的数据产出之后，才可以被自动推荐功能识别。

调度依赖配置指导：鼠标拖拽

- 应用场景

  DataWorks支持在业务流程的页面，直接通过连线的方式，指定各个节点的上下游关系。拉线完成后DataWorks根据您的拉线结果自动为您在各个节点中添加调度依赖配置。

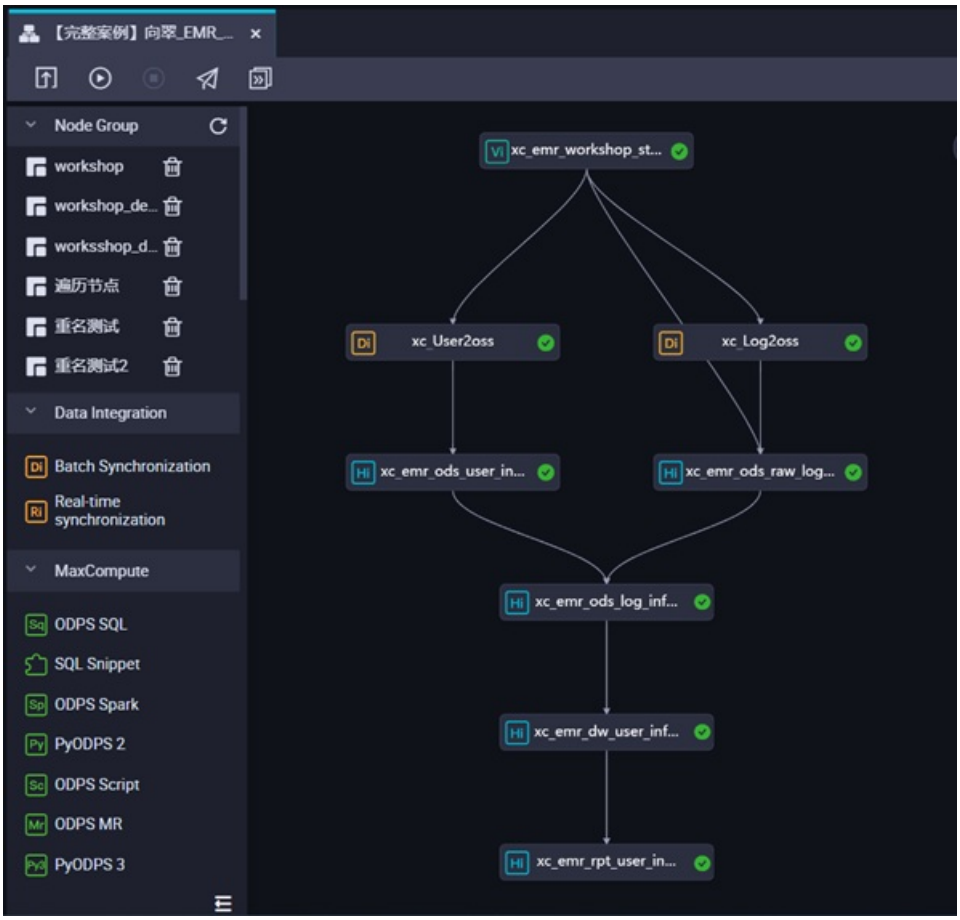
  当您创建完成业务流程后，您可根据业务规划，将各个节点按照节点的逻辑顺序，在业务流程页面通过拉线的方式配置好各个节点的依赖关系。后续在代码开发过程中，通过自动解析和手动修改的方式添加或修改各个节点的依赖关系，保障整体业务流程中所有节点的依赖关系是正确的。

- 实现原理

  拉线指定上下游依赖时，DataWorks将自动生成的后缀为*******_out**的输出添加为下游节点的输入中。

- 配置方法

  在业务流程编辑页面，通过连线的方式将节点连接起来。



预览依赖关系

当前节点的调度依赖配置完成后，您可以单击**预览依赖**，通过**任务依赖**和**实例依赖**维度，查看节点的上下游依赖关系，以便当节点的上下游依赖不符合预期时及时调整。

> **说明**
>
> - 根据当前调度配置生成的依赖关系预览图，与生产环境的实际依赖关系可能存在差异，仅供参考。
> - 目前仅支持**开发**、**运维**、**项目所有者**、**项目管理员**角色预览节点的依赖关系。如果您需要预览依赖关系，可为对应用户授权相关角色，详情请参见角色及成员管理：空间级。
> - 目前仅支持查看当前节点的一级上游和一级下游。
> - 单击**预览依赖**后，如果当前调度依赖未保存，您需要在**请注意**对话框单击**确认**，查看最新依赖关系。
> - **实例依赖**适用于预览产生多个周期实例的任务场景对应的依赖关系。例如，预览*小时任务依赖分钟任务*的依赖关系。
> - 当依赖的上游节点为已保存状态时，当前节点预览的依赖关系才会符合预期。

- 选择预览方式。

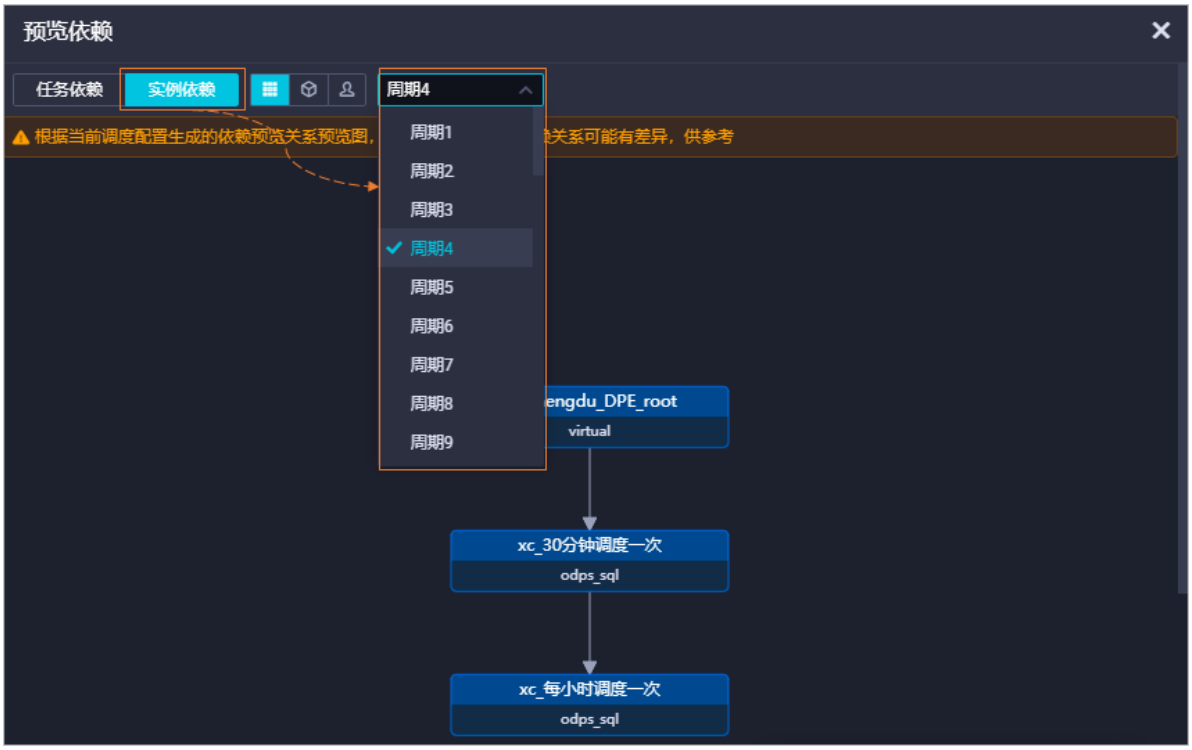
您可以选择**不聚合**、**按工作空间聚合**、**按责任人聚合**等方式预览依赖关系。聚合方式详情请参见DAG图功能介绍。

下图以预览**任务依赖**为例，为您展示不同聚合方式的预览效果。

单击目标节点即可查看节点的基本信息。

- 预览多周期实例任务的依赖关系。

  对于多周期任务，您可以使用**实例依赖**选择不同周期，查看相应依赖关系。



典型业务场景配置指导

- 场景1：包含离线同步节点的业务流程，如何配置调度依赖
- 场景2：依赖上一周期的结果时，如何配置调度依赖
- 场景3：如何配置跨业务流程、跨工作空间的调度依赖

调度配置完成后处理

各个节点在完成调度配置后，提交节点时，DataWorks会检查节点的调度依赖与节点代码中的数据血缘关系是否一致，如果出现不一致的提示，您需要根据实际情况查看是否需要修改调度依赖配置。详情可参见提交节点时提示：输入输出和代码血缘分析不匹配。

常见问题

- 提交节点报错：当前节点依赖的父节点输出名不存在
- 提交节点时提示：输入输出和代码血缘分析不匹配
- 依赖关系

9.5.3. 配置上一周期调度依赖

DataWorks的调度依赖中，依赖上一周期，是指本次节点的周期实例运行依赖某个节点的上一周期实例运行，即节点当前周期实例是否运行取决于其所依赖的节点上一周期实例是否正常运行。本文为您介绍上一周期依赖的配置说明及依赖形式。

配置说明

您需要进入数据开发节点的编辑页面，单击右侧导航栏的**调度配置**，在**调度配置 > 调度依赖**区域配置节点的依赖关系。DataWorks的调度依赖支持配置为依赖同周期和依赖上一周期两种方式。

两种调度依赖的区别如下表所示。

依赖方式	业务逻辑及使用场景	依赖关系展示
依赖同周期	当前节点依赖的上游表数据为某一节点当天产出的表数据（即当前节点代码中使用 `SELECT` 语句操作的表，为某一节点当天产出的表数据），此时，为了保障当前节点运行时能够成功取上游表数据，则需要为当前节点设置**同周期**依赖于产出上游表数据的节点。	在运维中心查看节点依赖关系时，同周期的依赖关系会以实线展示。进入运维中心查看任务依赖关系，详情请参见运维中心概述。
依赖上一周期	当前节点依赖的上游表数据为某一节点上一周期产出的表数据（即当前节点代码中使用 `SELECT` 语句操作的表，为某一节点上一周期产出的表数据），此时，为了保障当前节点运行时能够成功取上游表数据，则需要为当前节点设置**上一周期**依赖于产出上游表数据的节点。	在运维中心查看节点依赖关系时，依赖上一周期的依赖关系会以虚线展示。进入运维中心查看任务依赖关系，详情请参见运维中心概述。

依赖形式

依赖上一周期支持的依赖形式如下表所示。

依赖形式	节点依赖关系	业务场景
依赖上一周期：本节点	本节点本次实例运行，依赖于本节点上一周期的实例运行结果。即本次节点是否运行，取决于本节点上一周期的实例是否运行成功。	本节点本次实例运行，取决于本节点上一周期业务数据的产出情况。

依赖形式	节点依赖关系	业务场景
依赖上一周期：一级子节点	本节点本次实例运行，取决于下游节点上一周期的实例运行情况。即本次节点是否运行，取决于该节点的下游节点在上一周期的实例是否运行成功。 例如：节点A包含B、C、D三个下游节点，依赖一层子节点是指节点A依赖B、C、D三个节点在上一周期的运行结果。当B、C、D三个节点上一周期均运行成功时，本次节点A才会启动运行。	本节点本次实例的运行，依赖于该节点的下游节点在上一周期对本节点上一周期结果表（即本节点输出表）数据的清洗结果是否成功。
依赖上一周期：其他节点	本节点本次实例的运行，依赖于其他节点在上一周期的实例运行结果。即本次节点是否运行，取决于其依赖的其他节点在上一周期实例是否运行成功。 **说明** 依赖的其他节点需要您手动输入节点ID。	本节点本次实例运行，在业务逻辑上需要依赖其它业务的数据，但本节点中不包含涉及其他业务数据的相关操作。

您还可以为**依赖上一周期：本节点**和**依赖上一周期：其他节点**依赖形式配置**沿用上游的空跑属性**，详情请参见是否沿用上游的空跑属性。

配置完成后，您可以预览该节点的依赖关系，详情请参见预览依赖关系。

典型应用场景，详情请参见应用场景案例。

依赖上一周期：本节点

- **节点依赖关系**

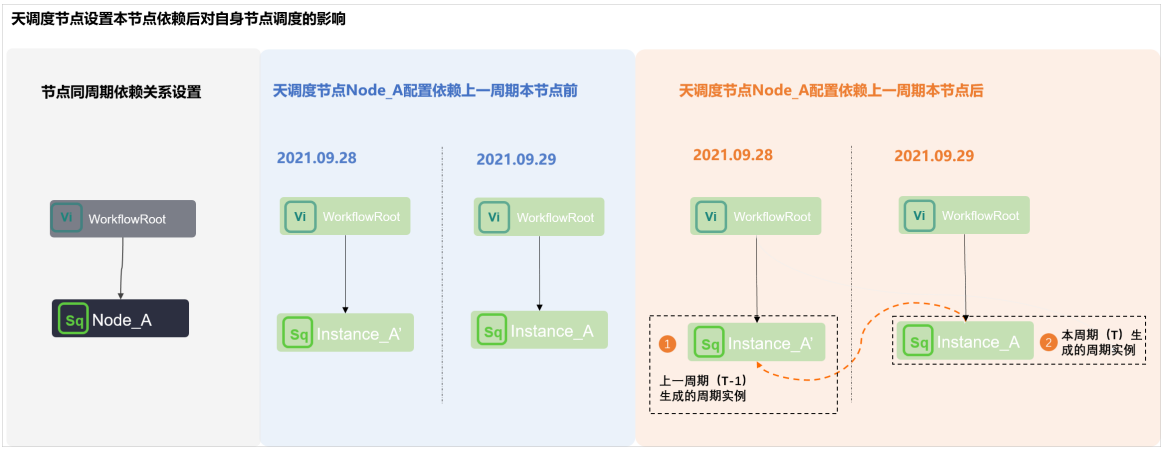
  本节点本次实例运行，依赖于本节点上一周期的实例运行结果。即本次节点是否运行，取决于本节点上一周期的实例是否运行成功。

- **业务场景**

  本节点本次实例运行，取决于本节点上一周期业务数据的产出情况。

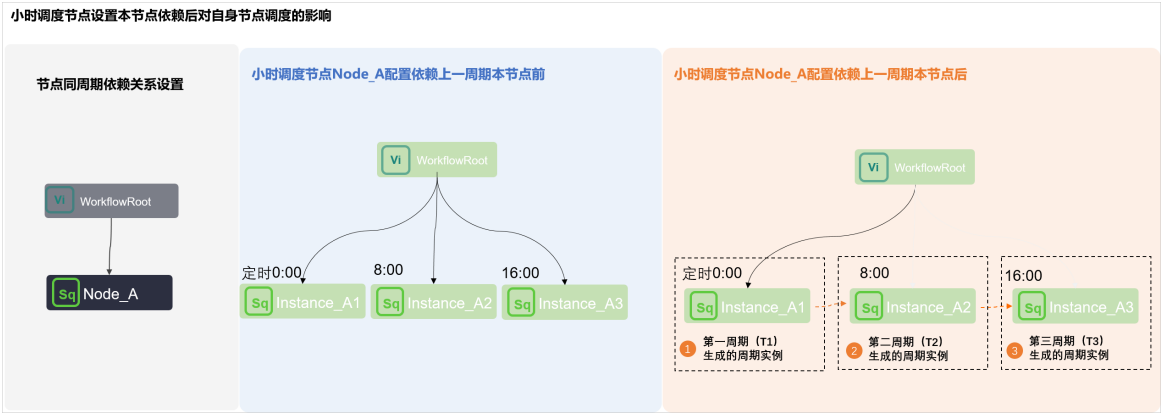
- **设置本节点依赖后对当前节点调度的影响**

- 场景示例一（天调度）：
  - 假设WorkflowRoot、Node_A均为天调度节点。
  - Node_A设置了本节点依赖。
  - Node_A在本周期（T）生成的周期实例名称为 `Instance_A` 。
  - Node_A在上一周期（T-1）生成的周期实例名称为 `Instance_A'` 。



配置**依赖上一周期：本节点**后， `Instance_A` 的运行依赖于 `WorkflowRoot` 、 `Instance_A'` 的运行结果，当上述两个周期实例均运行成功后， `Instance_A` 才会启动运行。

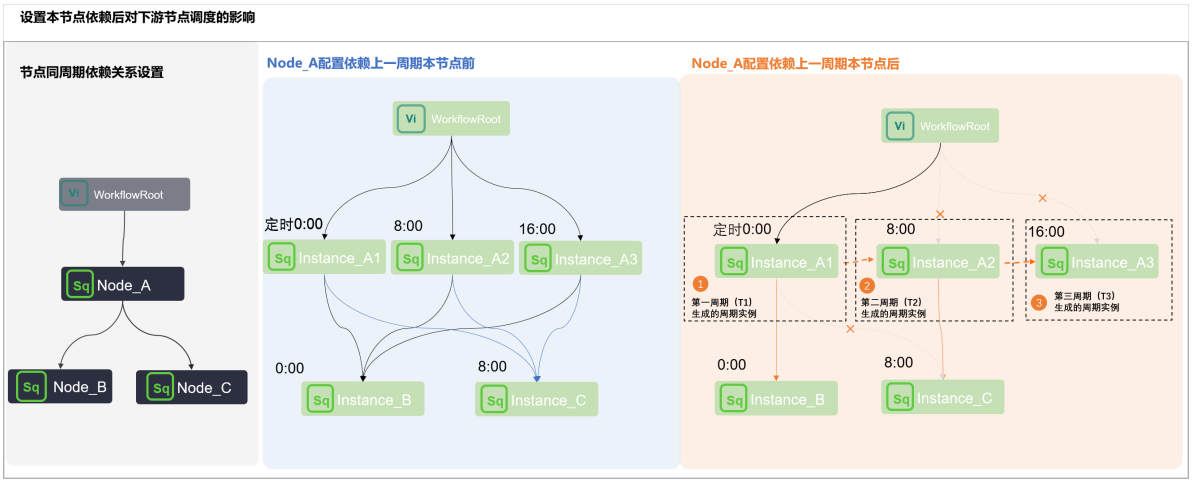
- 场景示例二（小时、分钟调度）：
  - 假设WorkflowRoot为天调度虚拟节点，Node_A为小时调度节点。
  - Node_A从 `00:00` 点开始至 `23:59` 点，每8小时调度一次，则其定时调度时间为 `00:00` （第一周期T1）、 `08:00` （第二周期T2）、 `16:00` （第三周期T3）。
  - Node_A节点在T1、T2、T3生成的周期实例名称分为 `Instance_A1` 、 `Instance_A2` 、 `Instance_A3` 。
  - Node_A设置了本节点依赖。



  - 当小时调度节点Node_A依赖天调度虚拟节点WorkflowRoot，小时调度节点不设置本节点依赖时，则小时调度节点当天的所有周期实例（即 `Instance_A1` 、 `Instance_A2` 、 `Instance_A3` ）在满足运行条件的情况下，可以同时被上游虚拟节点调度。
  - 当小时调度节点Node_A依赖天调度虚拟节点WorkflowRoot，小时调度节点设置本节点依赖后，则小时调度节点运行当天的周期实例，仅第一个周期实例（即T1的 `Instance_A1` ）依赖上游虚拟节点，其他周期实例则依赖自己的上一个小时实例（即 `Instance_A2` 依赖 `Instance_A1` ， `Instance_A3` 依赖 `Instance_A2` ）。仅当依赖的上一个周期实例运行成功后，当前周期实例才会启动运行。

> **说明** 本场景以小时任务示例，分钟任务调度逻辑类似。

- **设置本节点依赖后对下游节点调度的影响（小时、分钟任务）**
  - 假设WorkflowRoot为天调度虚拟节点，Node_A为小时调度节点，Node_B、Node_C为天调度节点，且Node_B、Node_C为Node_A的子节点。
  - Node_A从 `00:00` 点开始至 `23:59` 点，每8小时调度一次，则其定时调度时间为 `00:00` （第一周期T1）、 `08:00` （第二周期T2）、 `16:00` （第三周期T3）。
  - Node_B每天 `00:00` 运行，Node_C每天 `08:00` 点运行。
  - Node_A节点在T1、T2、T3生成的周期实例名称分为 `Instance_A1` 、 `Instance_A2` 、 `Instance_A3` 。
  - Node_B、Node_C生成的周期实例名称分别为 `Instance_B` 、 `InstanceC` 。
  - Node_A设置了本节点依赖。



- 当小时调度节点Node_A依赖天调度虚拟节点WorkflowRoot，小时调度节点不设置本节点依赖时，各下游节点运行情况如下：
  - Node_A当天的所有周期实例 `Instance_A1` 、 `Instance_A2` 、 `Instance_A3` ，同时被上游虚拟节点调度。
  - Node_A下游的周期实例 `Instance_B` 、 `InstanceC` 依赖小时任务当天所有的周期实例运行，即仅当 `Instance_A1` 、 `Instance_A2` 、 `Instance_A3` 全部运行成功后， `Instance_B` 、 `InstanceC` 才会启动运行。
- 当小时调度节点Node_A依赖天调度虚拟节点WorkflowRoot，小时调度节点设置本节点依赖后，各下游节点运行情况如下：
  - Node_A当天的所有周期实例均依赖于其上一个周期的实例。即T1的 `Instance_A1` 依赖虚拟节点WorkflowRoot、T2的 `Instance_A2` 依赖T1的 `Instance_A1` 、T3的 `Instance_A3` 依赖T2的 `Instance_A2` 。仅当依赖的上一周期实例运行成功后，当前实例才会运行。
  - Node_A下游的周期实例 `Instance_B` 、 `InstanceC` 依赖距离自己的定时运行时间最近的实例。

    `00:00` 点运行的 `Instance_B` 会在T1的 `Instance_A1` 运行成功后执行，此时 `InstanceC` 不会执行。

    `08:00` 点运行的 `InstanceC` 会在T2的 `Instance_A2` 运行成功后执行。

依赖上一周期：一级子节点

- **节点依赖关系**

  本节点本次实例运行，取决于下游节点上一周期的实例运行情况。即本次节点是否运行，取决于该节点的下游节点在上一周期的实例是否运行成功。

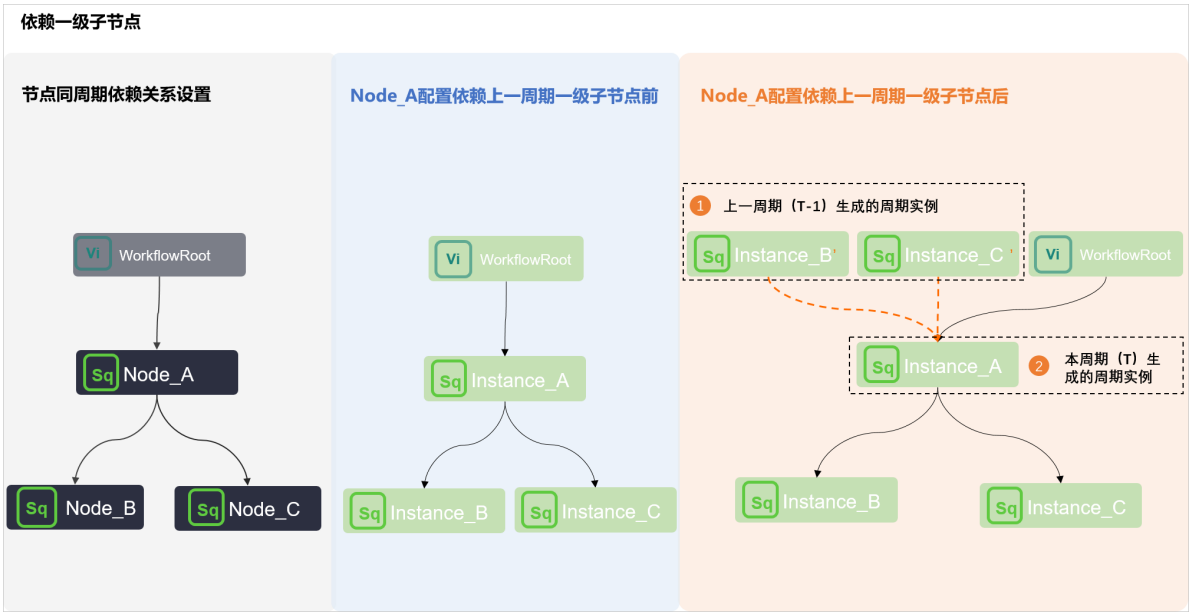
  例如：节点A包含B、C、D三个下游节点，依赖一层子节点是指节点A依赖B、C、D三个节点在上一周期的运行结果。当B、C、D三个节点上一周期均运行成功时，本次节点A才会启动运行。

- **业务场景**

  本节点本次实例的运行，依赖于该节点的下游节点在上一周期对本节点上一周期结果表（即本节点输出表）数据的清洗结果是否成功。

- **场景示例**
  - 假设WorkflowRoot、Node_A、Node_B、Node_C均为天调度节点。
  - Node_B、Node_C为Node_A的一级子节点。

- Node_A、Node_B、Node_C节点在本周期（T）生成的周期实例名称分别为`Instance_A`、`Instance_B`、`Instance_C`。
- Node_A、Node_B、Node_C节点在上一周期（T-1）生成的周期实例名称分别为`Instance_A'`、`Instance_B'`、`Instance_C'`。



配置**依赖上一周期：一级子节点**后，本周期（T）`Instance_A`的运行依赖于`WorkflowRoot`、上一周期（T-1）`Instance_B'`、上一周期（T-1）`Instance_C'`的运行结果，当上述三个周期实例均运行成功后，`Instance_A`才会启动运行。

依赖上一周期：其他节点

- **节点依赖关系**

  本节点本次实例的运行，依赖于其他节点在上一周期的实例运行结果。即本次节点是否运行，取决于其依赖的其他节点在上一周期实例是否运行成功。

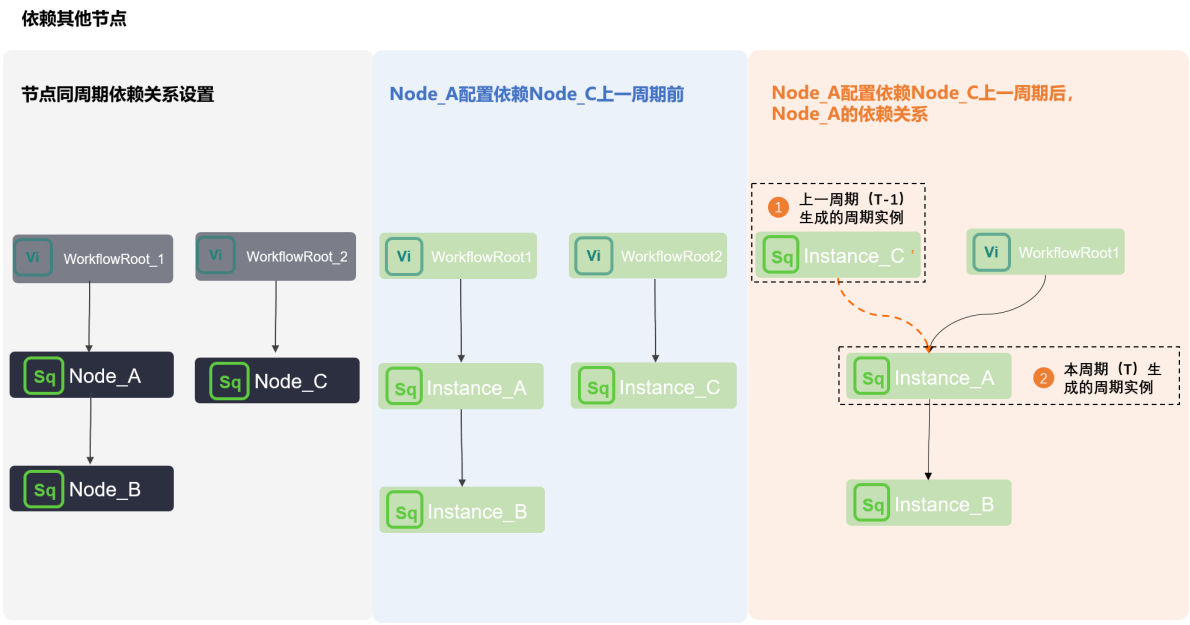
  > **说明** 依赖的其他节点需要您手动输入节点ID。

- **业务场景**

  本节点本次实例运行，在业务逻辑上需要依赖其它业务的数据，但本节点中不包含涉及其他业务数据的相关操作。

- **场景示例**
  - 假设WorkflowRoot_1、WorkflowRoot_2、Node_A、Node_B、Node_C节点均为天调度节点。
  - Node_A、Node_B、Node_C属于不同的业务流程。

    Node_A、Node_B为WorkflowRoot_1的子节点，Node_C为WorkflowRoot_2的子节点。
  - Node_A设置**依赖上一周期：其他节点**依赖Node_C。
  - Node_A、Node_B、Node_C节点在本周期（T）生成的周期实例名称分别为`Instance_A`、`Instance_B`、`Instance_C`。
  - Node_A、Node_B、Node_C节点在上一周期（T-1）生成的周期实例名称分别为`Instance_A'`、`Instance_B'`、`Instance_C'`。



配置**依赖上一周期：其他节点**后，本周期（T）`Instance_A`的运行依赖于`WorkflowRoot_1`、上一周期（T-1）`Instance_C'`的运行结果，当上述两个周期实例均运行成功后，`Instance_A`才会启动运行。

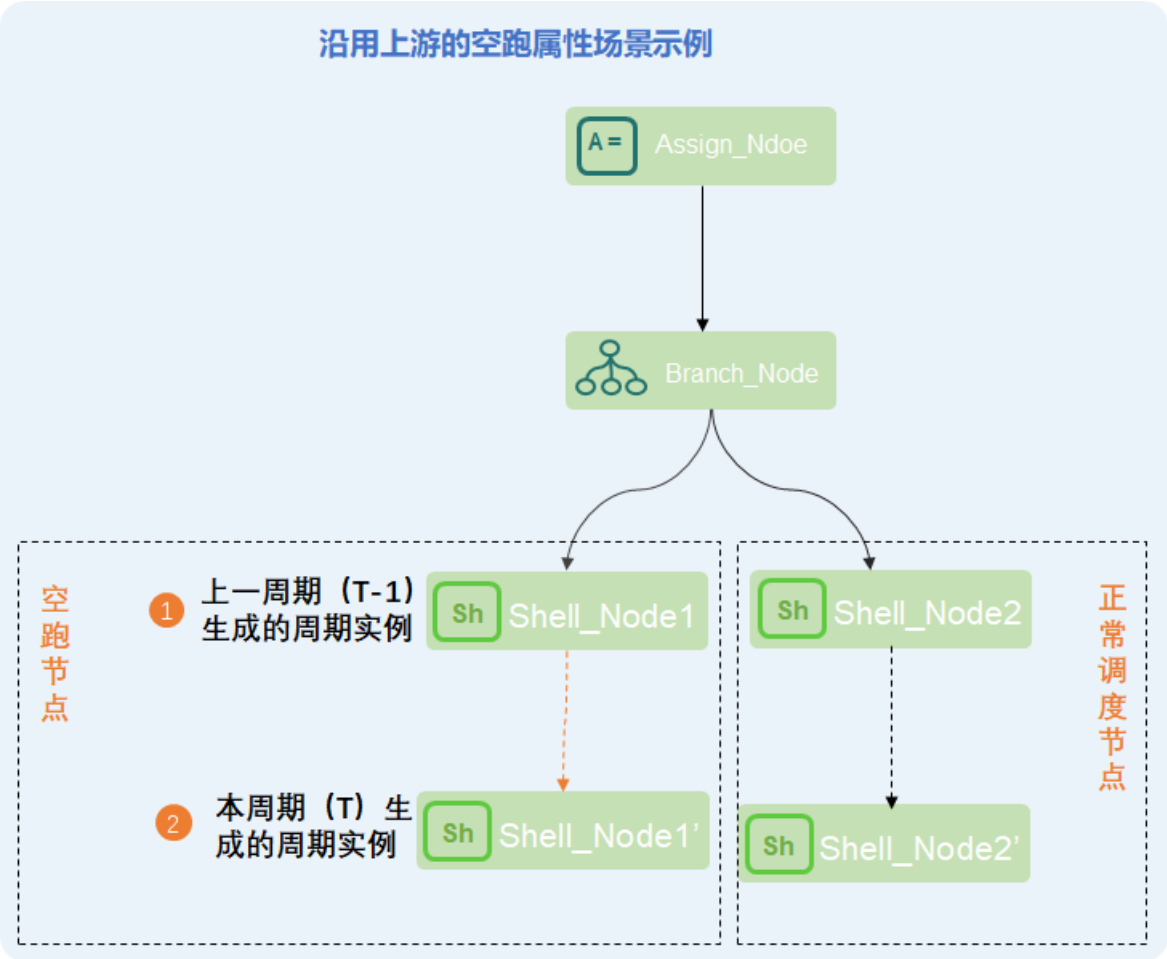
是否沿用上游的空跑属性

- **应用场景**

  通常，某些节点包含两个下游节点，在任务执行过程中，下游节点只有一个节点会正常运行，另一个节点则会被置为空跑状态。此时，当下游的空跑节点配置了依赖自身的上一周期，则该节点的空跑属性会不断向下传递至其子节点，导致出现该节点持续空跑的情况。当您不需要子节点延续被依赖节点的空跑属性时，可以在调度依赖中设置**沿用上游的空跑属性**为**否**。

  > **说明** 普通节点上一周期的空跑属性不会延续至下游节点，仅**分支节点**空跑时，其空跑属性会延续至下游节点。

- **场景示例**
  - 假设Assign_Ndoe为赋值节点，Branch_Node为分支节点，Shell_Node1、Shell_Node2为Branch_Node的下游节点。该节点均为天调度节点。
  - 实际运行时，Shell_Node1被置为空跑，Shell_Node2正常运行。
  - Shell_Node1节点设置了依赖自身的上一周期。
  - 本周期（T）Shell_Node1节点生成的周期实例名称为`Shell_Node1'`。
  - 上一周期（T-1）Shell_Node1节点生成的周期实例名称为`Shell_Node1`。



本周期（T）的周期实例 `Shell_Node1'` ，会依赖上一周期（T-1）的周期实例 `Shell_Node1` 运行，下游节点会延续上游节点的空跑属性，导致Shell_Node1节点永远空跑。您可以在调度依赖中配置**沿用上游的空跑属性**为**否**，从而使下游节点不被上一周期该节点的空跑属性影响。

预览依赖关系

当前节点的调度依赖配置完成后，您可以单击**预览依赖**，通过**任务依赖**和**实例依赖**维度，查看节点的上下游依赖关系，以便当节点的上下游依赖不符合预期时及时调整。

> **说明**
>
> - 根据当前调度配置生成的依赖关系预览图，与生产环境的实际依赖关系可能存在差异，仅供参考。
> - 目前仅支持**开发**、**运维**、**项目所有者**、**项目管理员**角色预览节点的依赖关系。如果您需要预览依赖关系，可为对应用户授权相关角色，详情请参见角色及成员管理：空间级。
> - 目前仅支持查看当前节点的一级上游和一级下游。
> - 单击**预览依赖**后，如果当前调度依赖未保存，您需要在**请注意**对话框单击**确认**，查看最新依赖关系。
> - **实例依赖**适用于预览产生多个周期实例的任务场景对应的依赖关系。例如，预览*小时任务依赖分钟任务*的依赖关系。
> - 当依赖的上游节点为已保存状态时，当前节点预览的依赖关系才会符合预期。

- 选择预览方式。

  您可以选择**不聚合**、**按工作空间聚合**、**按责任人聚合**等方式预览依赖关系。聚合方式详情请参见DAG图功能介绍。

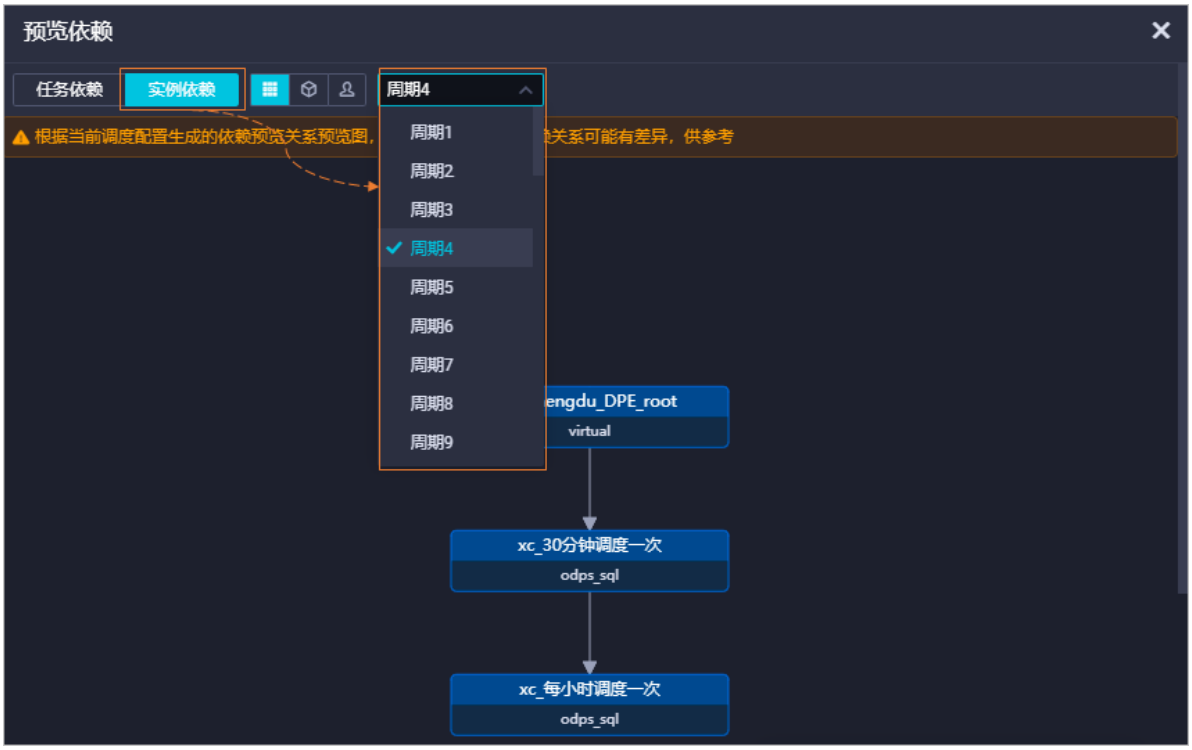
  下图以预览**任务依赖**为例，为您展示不同聚合方式的预览效果。

  单击目标节点即可查看节点的基本信息。

- 预览多周期实例任务的依赖关系。

  对于多周期任务，您可以使用**实例依赖**选择不同周期，查看相应依赖关系。



应用场景案例

依赖上一周期的典型应用场景案例，详情请参见场景2：依赖上一周期的结果时，如何配置调度依赖。

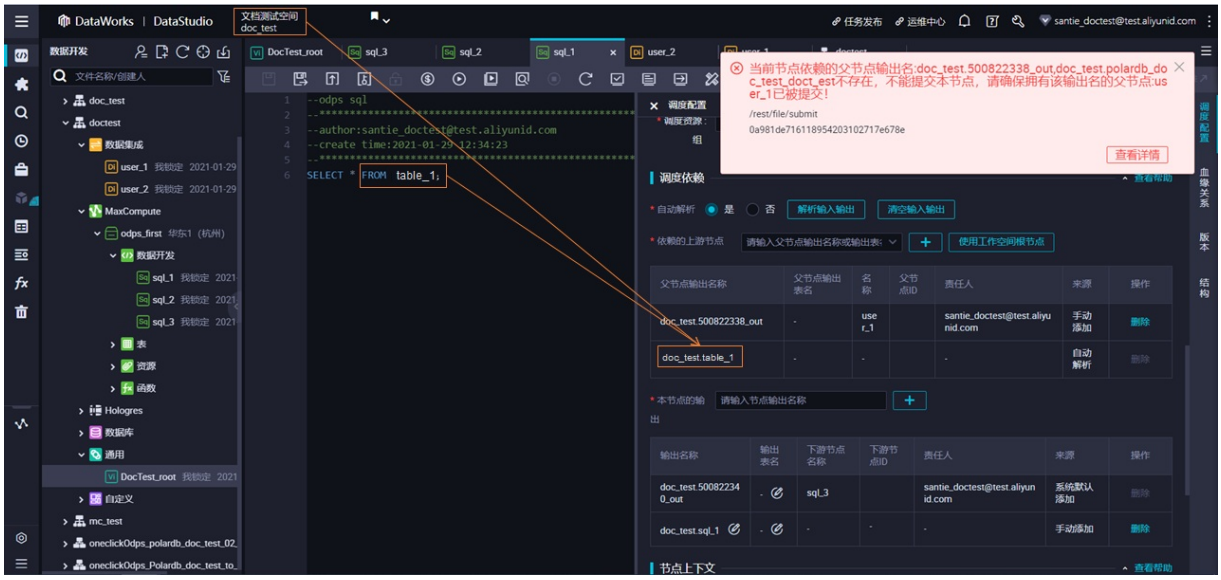
9.5.4. 典型应用场景案例

9.5.4.1. 场景1：包含离线同步节点的业务流程，如何配置调度依赖

DataWorks的离线同步节点不支持通过自动解析自动添加调度依赖，包含离线同步节点的业务流程，如果下游节点依赖离线同步节点产生的表，您需手动添加产出表到离线同步节点的输出中，下游节点查询离线同步节点数据时，自动解析可以通过表快速找到产出该表数据的离线同步节点。

易错点

如果您没有将离线同步节点的产出表手动添加到离线同步节点的输出中，自动解析无法找到此离线同步节点，提交引用此离线同步节点的SQL节点时，会出现如下错误提示。

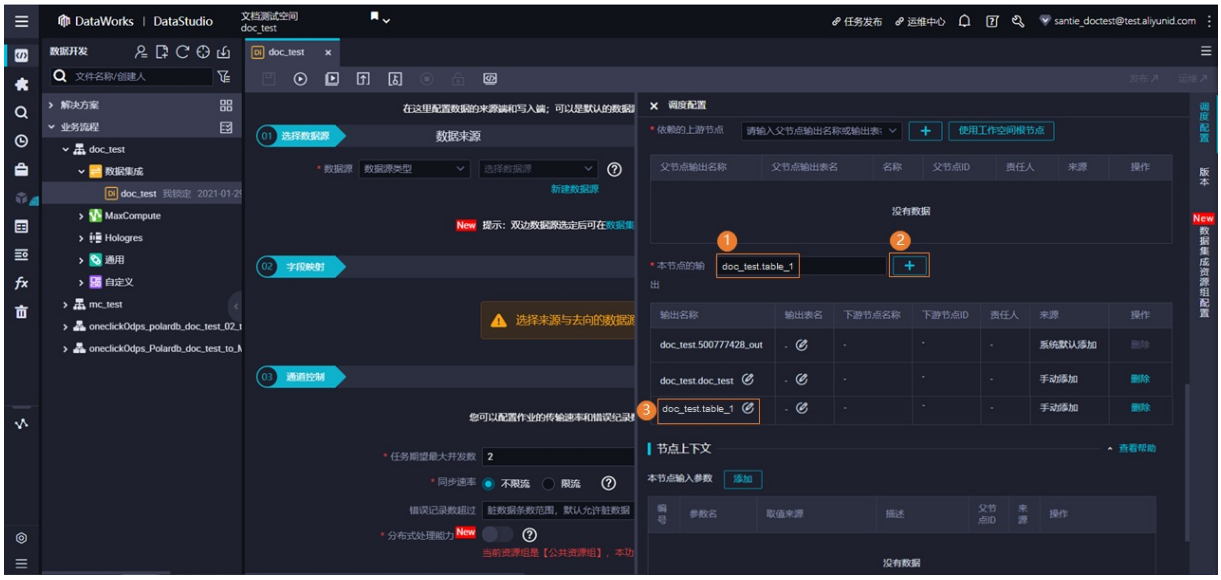


出现此种错误主要是下游节点中自动解析出来的上游依赖，无法匹配找到对应的上游离线节点，详细原因分析可参见易错原因详解。为避免此类易错点，建议包含离线同步节点的业务流程中，调度依赖配置参考以下两种方式进行配置：

- 配置方法1：手动将离线节点产出表添加为本节点产出
- 配置方法2：离线节点名称与产出表名称保持一致

配置方法1：手动将离线节点产出表添加为本节点产出

通过上述报错原因分析可见，如果要避免此类错误需保障下游节点自动解析出来的上游依赖，被添加至上游节点的本节点的输出中。因此，在完成上述几个步骤后，您可以在离线节点的调度配置页面，手动将产出表添加为节点输出，如下图所示。



配置方法2：离线节点名称与产出表名称保持一致

根据上述流程描述可知：

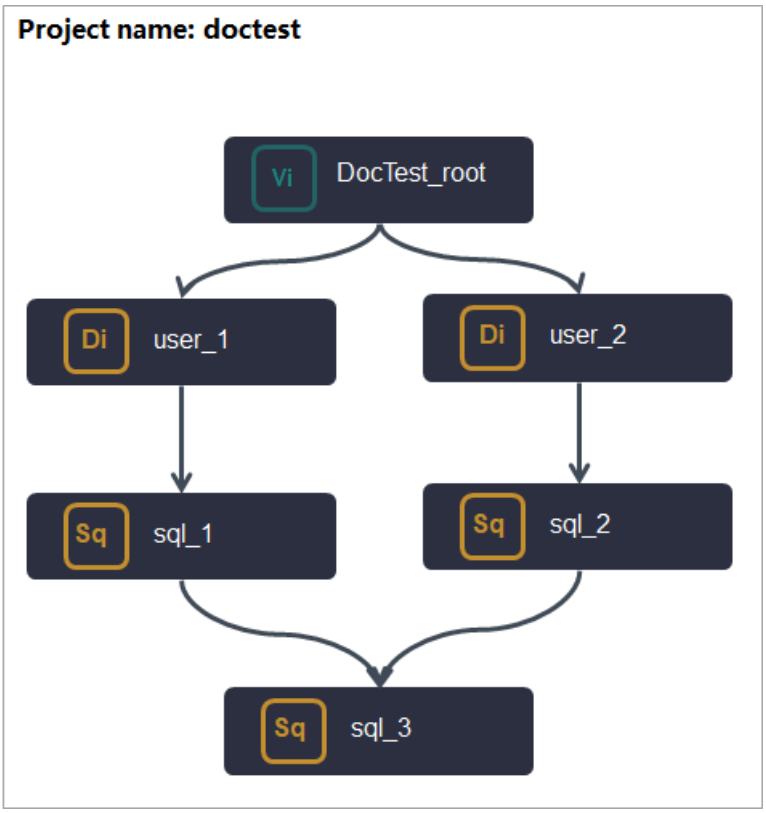
- 创建离线同步节点时，会为节点自动生成一个命名规则为 `projectname.nodename` 的**本节点的输出**。
- SQL节点引用离线节点产出表时，会为SQL节点自动生成一个命名规则为 `projectname.tablename` 的**依赖的上游节点**。
- 为避免报错，需要保障SQL节点中**依赖的上游节点**名称和离线节点的**本节点的输出**名称一致。

因此，您可以将离线节点的节点名称（nodename）和离线节点产出的表名称（tablename）保持一致，此种情况下即可满是上述要求，提交节点时不会报错。

> **说明** 自动生成的 `projectname.nodename` 的**本节点的输出**是在创建节点时生成的，节点创建完成后，如果修改节点名称，已自动生成的 `projectname.nodename` 的**本节点的输出**不会随之改变名称。因此本方法仅适用与创建离线节点时使用，后续修改节点名称与产出表名称一致的场景无法解决本文说明的问题。

易错原因详解

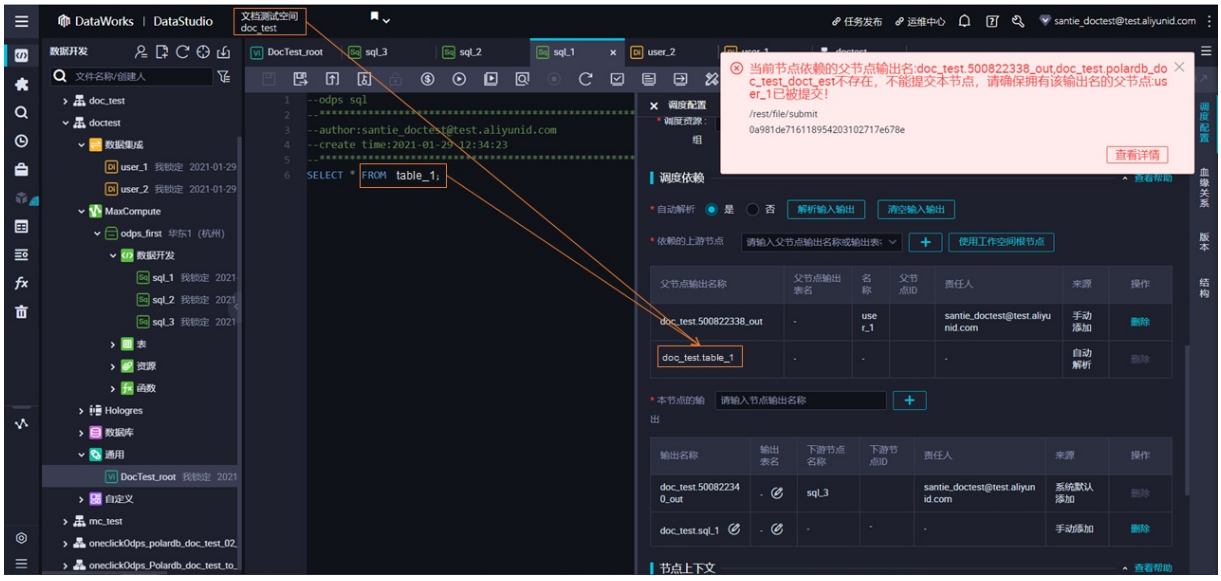
包含离线同步节点的业务流程，以下图为例，常见节点创建和依赖配置操作如下：



步骤序号	步骤详情	调度依赖配置结果

步骤序号	步骤详情	调度依赖配置结果
1	根据业务流程规划，创建各节点。 以上图为例，即创建虚拟节点、离线同步节点、ODPS节点。	在DataWorks中创建节点后，DataWorks自动为各节点生成两个**本节点的输出**的配置信息，其中一个本节点的输出名称的后缀为**_out**，一个本节点的输出名称为 `projectname.nodename` 。 以上图中的离线节点user_1为例，创建节点后，节点即有： • 一个名称为 `*******_out` 的本节点的输出。 • 一个名称为 `doctest.user_1` 的本节点的输出。
2	根据业务流程规划，为各个节点拉线，明确各节点的运行逻辑顺序上的上下游依赖关系。	在业务流程页面为各节点拉线后，DataWorks根据拉线的结果，自动为各个父节点添加依赖的配置信息。 以上图中的ODPS节点sql_1为例，拉线完成后，离线节点user_1在sql_1的上游，DataWorks自动将user_1的名称为 `*******_out` 的输出添加为sql_1的**依赖的上游节点**。
3	为各节点开发任务代码。	在各个节点中开发任务代码时，DataWorks会根据业务代码自动解析，通过代码中的部分输入、输出命令，自动为节点添加**本节点的输出**或**依赖的上游节点**。 以上图中的ODPS节点sql_1为例，如果sql_1节点需要取用离线节点user_1产出的表 `table_1` 的数据时，例如出现了类似 `select * from table_1` 这类的语句，DataWorks会自动为sql_1添加一条**依赖的上游节点**，且自动添加的父节点输出名称的命名规则为 `projectname.tablename` ，本示例即为 `doctest.table_1` 。

完成上述步骤后，如果您没有关注到离线节点无法自动解析，为离线节点自动将输出表添加为节点的**本节点的输出**，您直接提交业务流程节点时，系统会报错，提示您依赖的父节点输出名不存在。



出现此种错误的原因是：

- 由于离线节点不支持自动解析，所以离线节点的产出表table_1没有被自动添加为离线节点的**本节点的输**

出。即离线节点user_1没有 `doctest.table_1` 这条输出。

- 离线节点的下游节点sql_1因为自动解析，添加了一条命名规则为 `projectname.tablename` 的**依赖的上游节点**，本示例即为 `doctest.table_1` ，但是由于 `doctest.table_1` 没有作为user_1的输出，所以sql_1中添加的这条父节点依赖无法匹配到user_1的节点ID。
- 提交sql_1节点时，系统检测到sql_1有 `doctest.table_1` 这个上游依赖，但是因为这条上游依赖没有关联到节点ID，系统无法通过这条依赖找到对应的上游节点，出现报错，提示依赖的父节点输出名不存在。

9.5.4.2. 场景2：依赖上一周期的结果时，如何配置调度依赖

依赖上一周期是指依赖某个节点的上一周期实例，即跨周期依赖某节点上一周期实例是否正常执行。

DataWoks支持以下三种跨周期依赖形式：

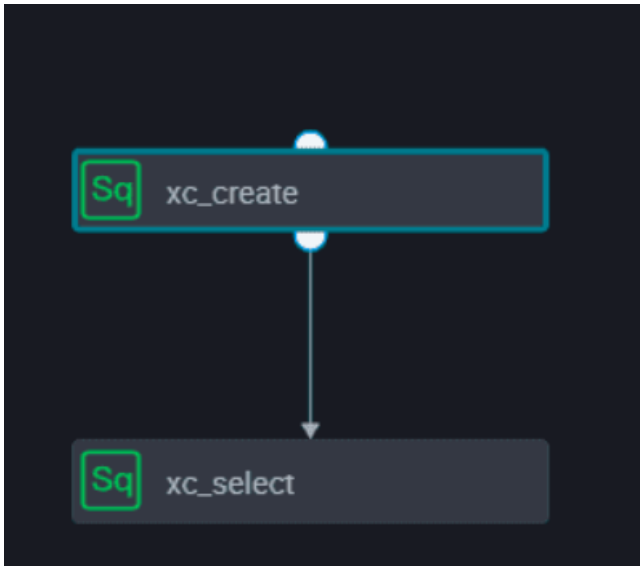
- 一层子节点
  - 节点依赖关系：依赖当前节点的下游。例如节点A存在B、C、D三个下游节点，依赖一层子节点是指节点A依赖B、C、D三个节点的上一周期，即本次节点是否运行取决于上一周期下游节点是否运行成功。
  - 业务场景：本次（本周期）节点的运行，依赖下游节点上一周期对本节点上一周期结果表（即本节点输出表）的数据清洗结果是否成功。如果您需要查看下游节点对当前节点数据的清洗结果是否符合预期，可以对下游节点产出的结果表配置数据质量规则。
- 本节点
  - 节点依赖关系：跨周期自依赖（依赖当前节点的上一周期），即本次节点是否运行取决于上一周期本节点是否运行成功。
  - 业务场景：本周期节点运行依赖上一周期该节点业务数据的产出情况。如果需要查看节点数据清洗结果是否符合预期，可以对节点产出的结果表配置数据质量监控规则。
- 自定义：手动输入需要依赖的其他节点，此处需要输入节点ID。如果存在多个节点，需要使用英文逗号（,）分隔，例如12345,23456：
  - 节点依赖关系：手动输入需要依赖的节点，本周期节点运行取决于自定义依赖的节点上一周期该是否运行成功。
  - 业务场景：业务逻辑上需要依赖其它业务的数据正常产出，但本节点中没有操作该业务数据。

依赖上一周期和依赖本周期的区别：在运维中心中查看节点依赖关系时，所有跨周期依赖的节点都会以虚线的形式展示。

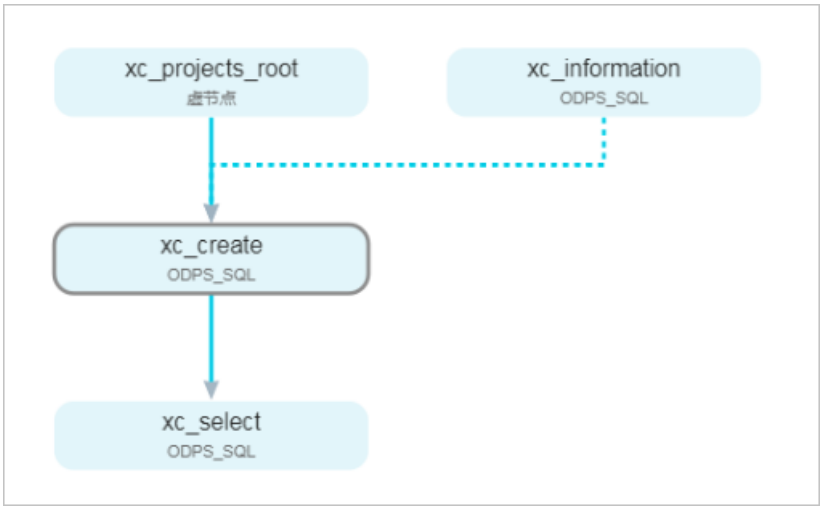
下线节点时需要删除节点依赖关系，需要删除的依赖关系包括跨周期依赖（①）和同周期依赖（②）。

您可以根据业务需求选择需要依赖的上游节点的周期。通常同周期依赖和跨周期依赖只需要选择一个，自动解析默认依赖上游同周期。如果需要修改，请删除同周期依赖，再添加跨周期依赖。详情请参见调度依赖逻辑说明。

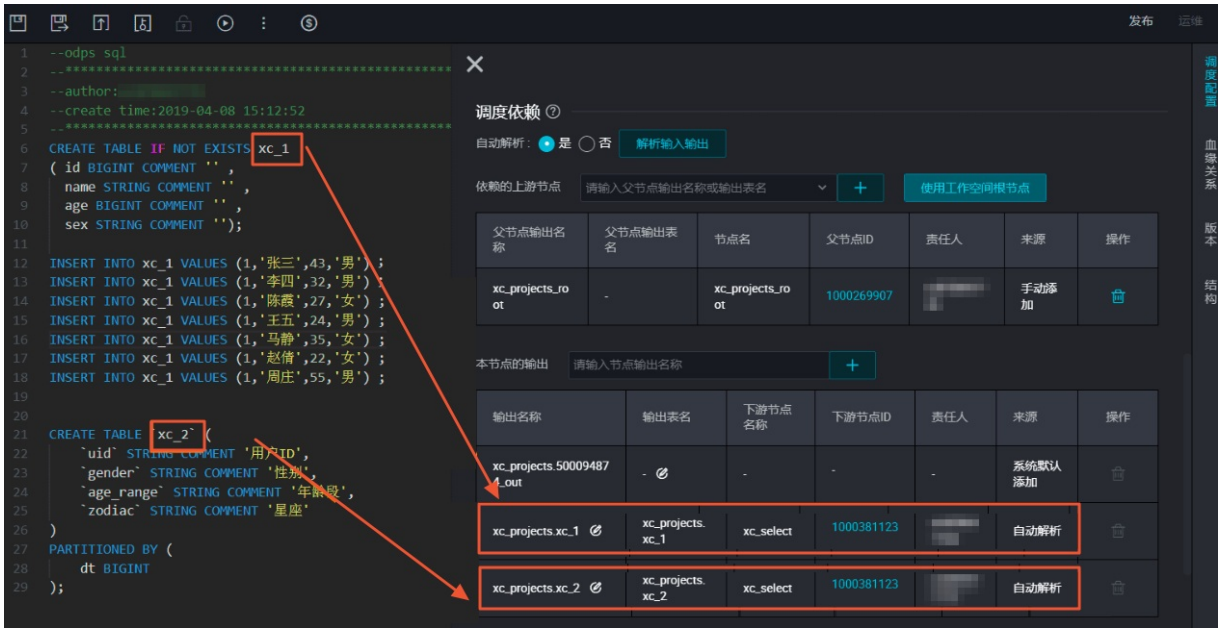
下图为业务流程节点的依赖关系。



运维中心页面为您展示业务流程的依赖关系。

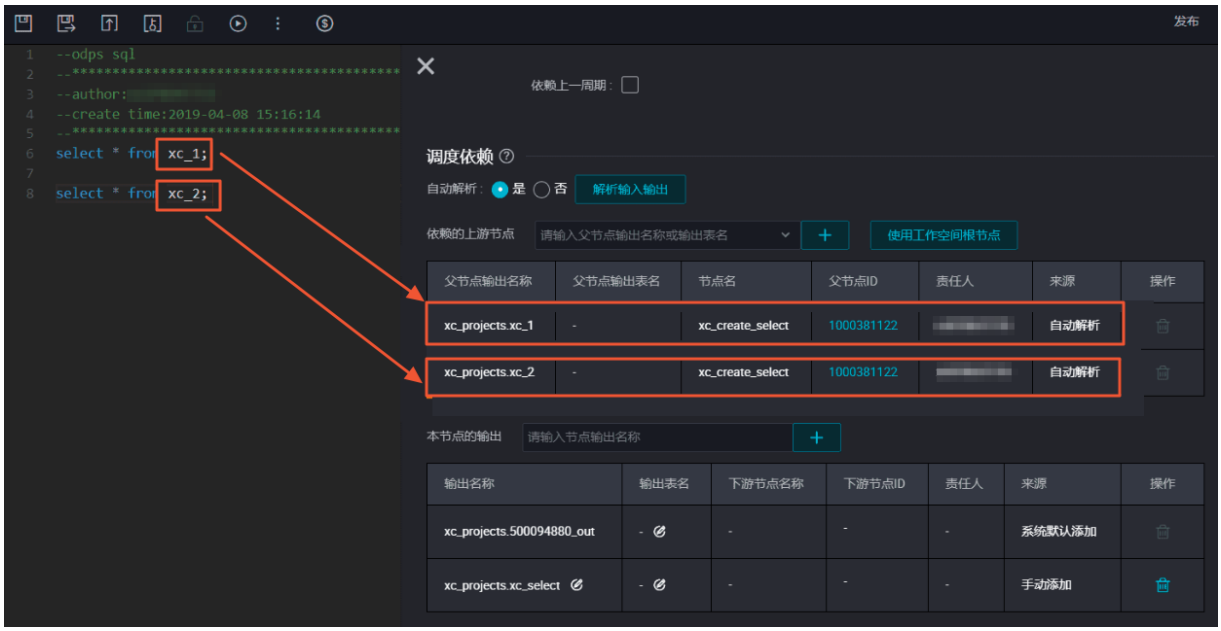


以配置xc_create节点代码为例。



如上图中的SQL节点内容所示，xc_create节点创建xc_1、xc_2两张表（或产出两张表的数据），并将xc_1、xc_2作为本节点的输出。

以配置xc_select节点代码为例。



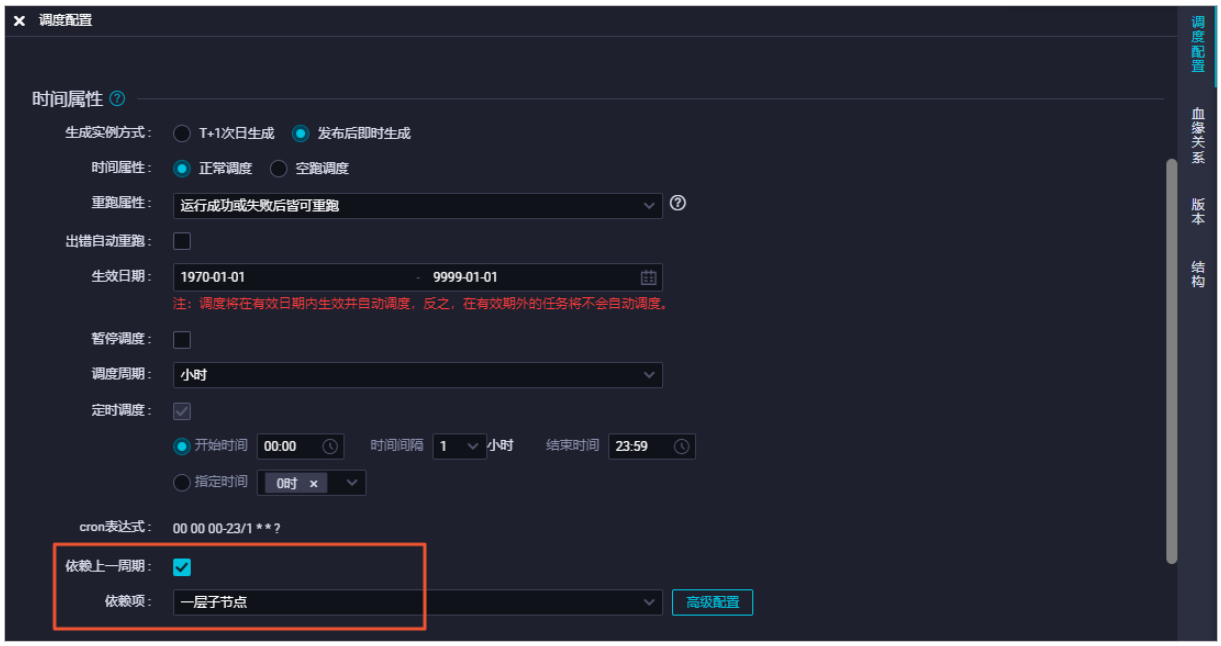
如上图中的SQL节点内容所示，xc_select节点查询xc_create节点中的表数据，通过自动解析功能，自动将xc_create节点解析为本节点依赖的上游。

依赖上一周期：一层子节点

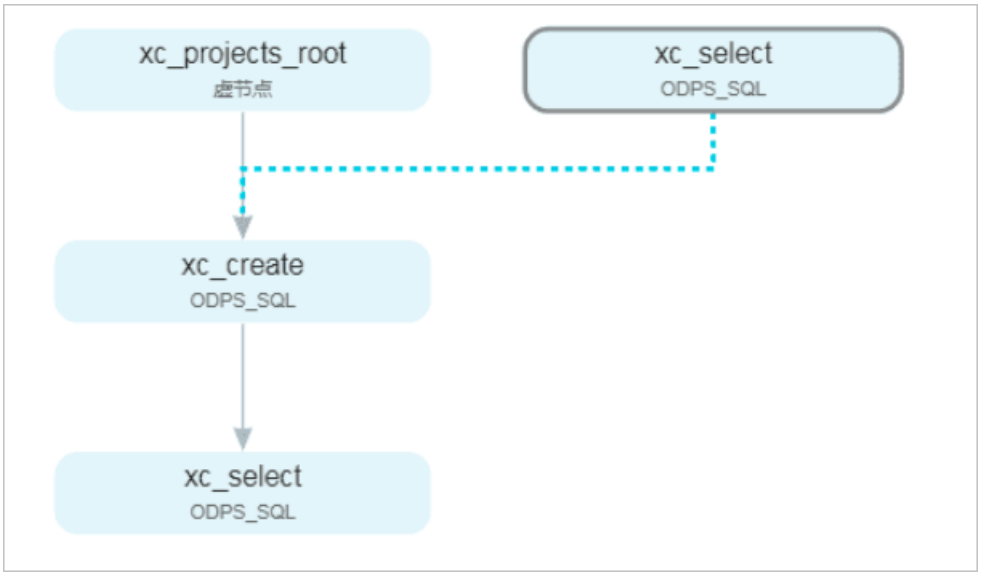
节点依赖：依赖当前节点的下游。例如，下游节点的上一周期是否运行成功例如节点A存在下游节点B、C、D三个节点，依赖一层子节点是节点A依赖B、C、D三个节点的上一周期。

业务场景：本周期该节点是否进行数据清洗取决于下游节点上一周期对本节点的结果表（即本节点输出表）数据清洗的结果。如果下游节点的上一周期运行成功，本周期的节点实例开始运行，否则将不能运行。

xc_create选择依赖一层子节点。



运维中心页面为您展示各节点的依赖关系。



依赖上一周期：本节点

节点依赖：本周期节点是否运行取决于上一周期本节点是否运行成功。如果上一周期本节点未完成，将阻碍本周期节点运行。

业务场景：本次节点是否进行数据清洗取决于上一周期本节点数据清洗情况。此处设置节点为小时调度以便查看。

您可以进入**运维中心 > 周期任务运维 > 周期实例**页面，查看节点的依赖情况。

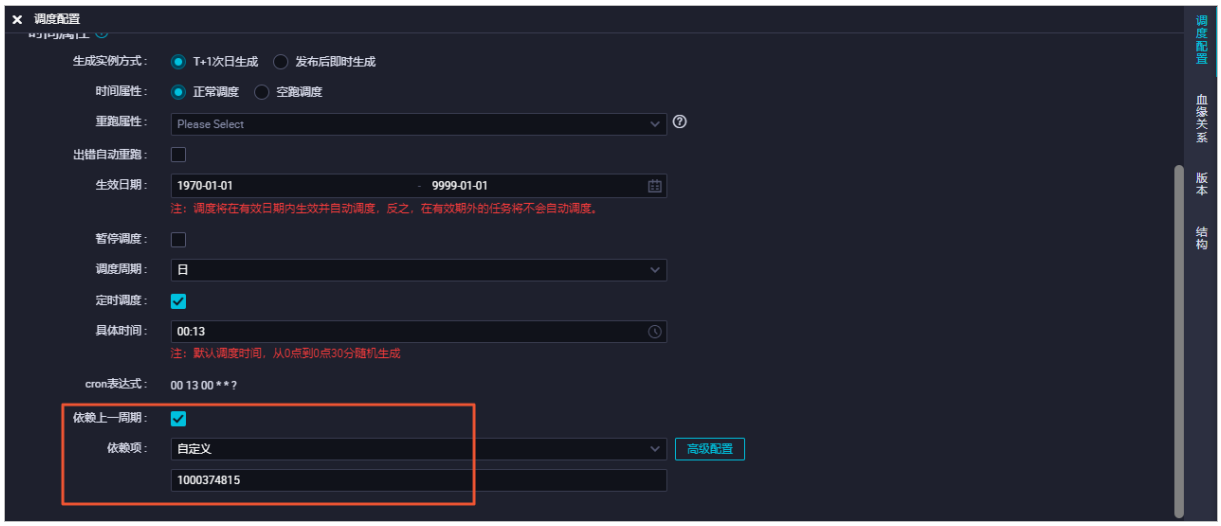
> **说明** 小时节点设置自依赖（依赖上一周期：本节点）的情况下，如果本节点上一周期实例未成功运行，则该节点下一个小时实例也不会执行。
>
> 例如，每小时调度的任务，如果第一个实例执行失败了或者未运行，则当天该节点的其它小时实例也不会运行。

依赖上一周期：自定义节点

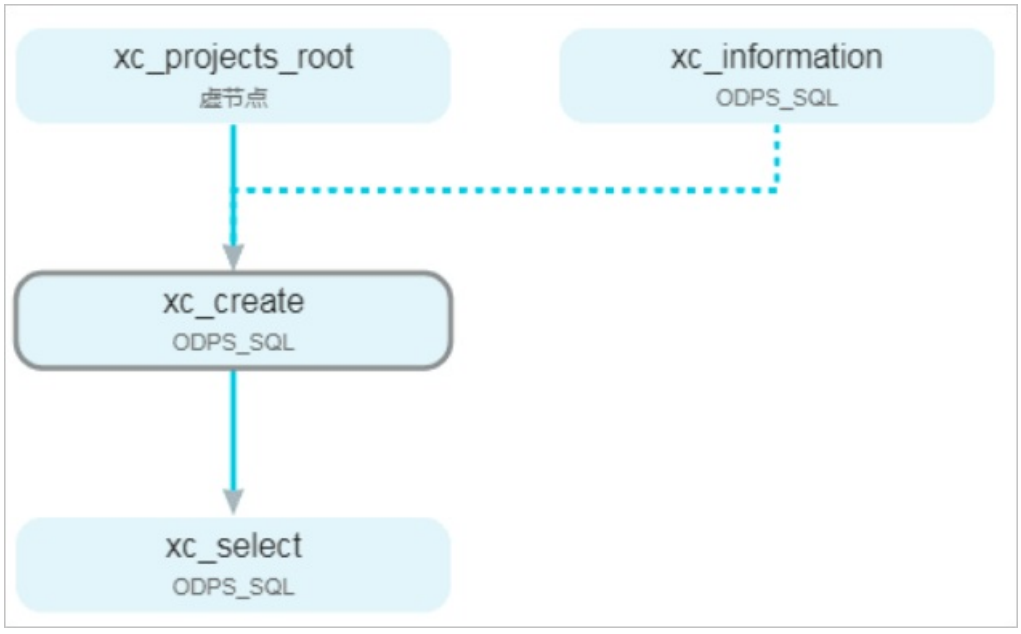
节点依赖：代码中没有用到1000374815节点的产出表，但业务上需要依赖该1000374815节点的上一周期是否正常产出数据。从节点关系来说，xc_create节点需要依赖1000374815节点的上一周期。

业务场景：业务逻辑上需要依赖1000374815节点正常产出的业务数据，但本节点（xc_create）中没有操作该业务数据（没有select1000374815节点产出的结果表）。

xc_create节点依赖的上游自定义选择1000374815节点。



您可以进入**运维中心 > 周期任务运维 > 周期实例**页面，查看节点的依赖情况。

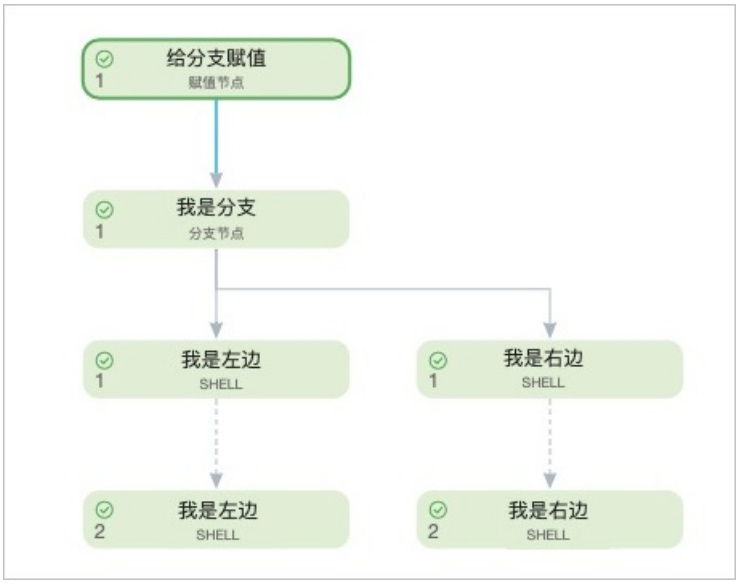


依赖上一周期高级配置

考虑到某些分支节点的下游有两个节点，通常只有一个节点被选中，另外一个节点会被置为空跑，同时该空跑属性会不断向下传导至其子节点的情况，DataWorks新增了**上游节点空跑属性不进行跨周期传导**的调度特性。

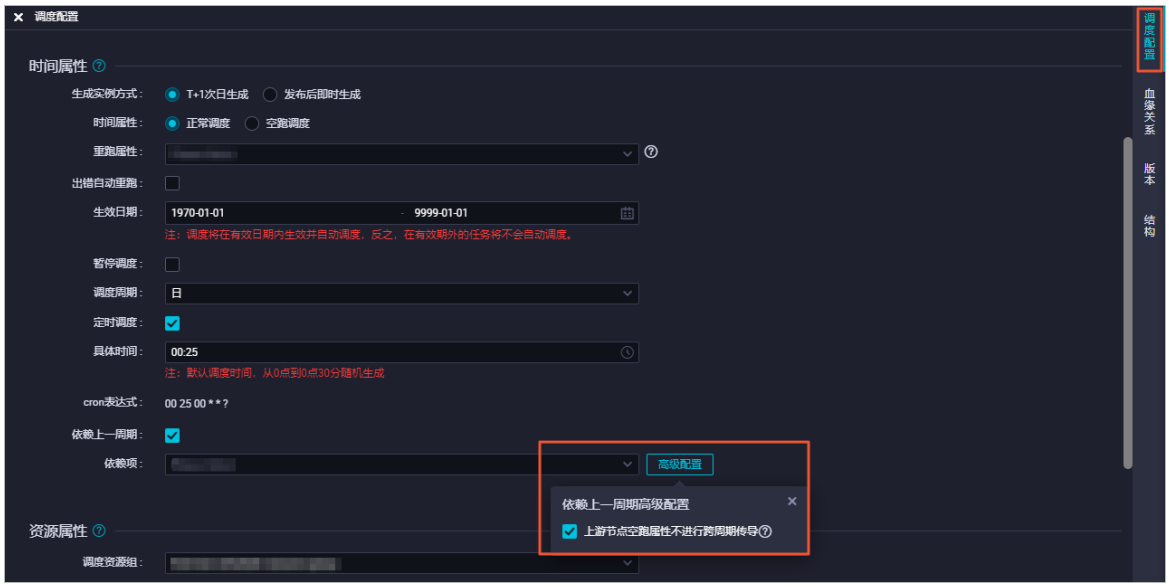
但如果下游分支节点中，某一个分支节点依赖自身的上一周期，同时上一周期节点未被选中，则该节点会永远空跑。

例如，节点（我是左边）被置为空跑，则其下游节点也会被置为空跑。



为了满足下一周期的节点是否运行由下一周期的分支节点决定，而不是上一周期的空跑属性来决定的需求，您可以进行以下操作：

1. 单击节点编辑页面右侧的**调度配置**。
2. 在**时间属性**区域，选中**依赖上一周期**。
3. 单击**高级配置**。
4. 选中**上游节点空跑属性不进行跨周期传导**，该任务将不被上一周期分支节点的空跑属性影响。



> **说明** 普通节点上一周期的空跑属性不适用该选项，仅分支节点未被选中导致的空跑属性会被影响。

跨周期依赖的典型场景

- 实时场景一：

- 场景描述：天任务依赖小时任务，但不想等24个小时任务实例运行结束才运行天任务，希望天任务能够在定时时间12:00时运行。
- 解决方案：上游小时任务配置为**依赖上一周期 > 本节点**，设置下游天任务的定时调度时间为12:00，且天任务无需设置跨周期依赖。

  待上游小时任务定时时间12点的实例运行成功后，下游天任务便会运行。

- 实时场景二：
  - 实时场景：天任务依赖小时任务昨天的数据。
  - 解决方案：下游天任务配置为**依赖上一周期 > 自定义**，输入上游小时任务的ID。
- 实时场景三：
  - 实时场景：小时任务依赖天任务，当上游天任务运行结束，下游小时任务多个周期定时时间已到，导致小时任务多周期并发调起，该如何处理？
  - 解决方案：下游小时任务配置为**依赖上一周期 > 本节点**。
- 实时场景四：
  - 实时场景：本节点依赖自己上一周期产出的数据，如何确认上一周期产出的时间？
  - 解决方案：本节点配置为**依赖上一周期 > 本节点**。

9.5.4.3. 场景3：如何配置跨业务流程、跨工作空间的调度依赖

本文为您介绍跨业务流程、跨工作空间场景下，如何设置节点的调度依赖。

跨业务流程配置调度依赖

包含多个分支结果的业务流程如果要实现跨业务流程依赖，您需要使用虚拟节点对多个分支节点进行汇总，再手动将该汇总节点的输出作为下游业务流程统筹根节点的输入，以此方式实现跨业务流程依赖。

> **说明** 虚拟节点属于控制类节点，它是不产生任何数据的空跑节点，通常作为业务流程统筹节点的根节点，或作为业务流程中多个分支节点的汇总输出节点使用。一个业务流程存在多个分支结果时，您需要新建一个虚拟节点（例如，`业务流程_end_虚拟节点`），`业务流程_end_虚拟节点`依赖上游多个分支结果，当`业务流程_end_虚拟节点`执行成功，则表示该业务流程执行完成。

当包含多个分支结果的业务流程需要实现跨业务流程依赖时，则可以使用虚拟节点配置上下游依赖关系。示例如下。

- 创建两个业务流程：业务流程1、业务流程2，业务流程1作为业务流程2的上游。
- 上游业务流程1创建如下虚拟节点。
  - `业务流程1_start_虚拟节点`：统筹起始节点，作为上游业务流程1中多分支节点的统筹起始节点。
  - `业务流程1_end_虚拟节点`：汇总输出节点，用于对上游业务流程1的多分支节点进行汇总输出。
- 下游业务流程2创建如下虚拟节点。
  - `业务流程2_start_虚拟节点`：统筹起始节点，下游业务流程2中多分支节点的统筹起始节点。
  - `业务流程2_end_虚拟节点`：汇总输出节点，用于对下游业务流程2的多分支节点进行汇总输出。
- 上下游业务流程的依赖关系：配置`业务流程1_end_虚拟节点`的输出作为`业务流程2_start_虚拟节点`的输入，从而实现跨业务流程调度依赖。

> **说明** DataWorks的依赖关系是通过将上游节点的输出配置为下游节点的输入，以此形成节点依赖，您可以使用**鼠标拖拽**、**手动配置**、**自动解析**三种方式配置节点的依赖关系。本示例通过在下游节点`业务流程2_start_虚拟节点`的**依赖的上游节点**配置区域，手动输入上游节点`业务流程1_end_虚拟节点`的输出，从而形成节点依赖。
>
> - 创建业务流程，详情请参见创建业务流程。
> - 创建虚拟节点，详情请参见虚拟节点。
> - 配置调度依赖，详情请参见配置同周期调度依赖。

跨工作空间配置调度依赖

DataWorks支持同区域下的工作空间进行跨工作空间依赖，根据调度依赖原理，通过将上游节点的输出作为下游节点的输入，以此形成节点依赖，实现跨工作空间的调度依赖。例如，将工作空间A中节点A的输出添加为工作空间B中节点B的输入，即可实现跨工作空间依赖。配置方法与通用场景的调度依赖配置相同，详细操作可参见配置同周期调度依赖。

> **说明** 对于部分早期创建的工作空间，标准模式工作空间依赖简单模式工作空间可能无法支持，请提交工单申请修复。

9.6. 配置节点上下文

节点上下文用于支持参数在上游节点和下游节点之间传递，本文为您介绍如何定义、使用节点上下文中的输入参数和输出参数。

背景信息

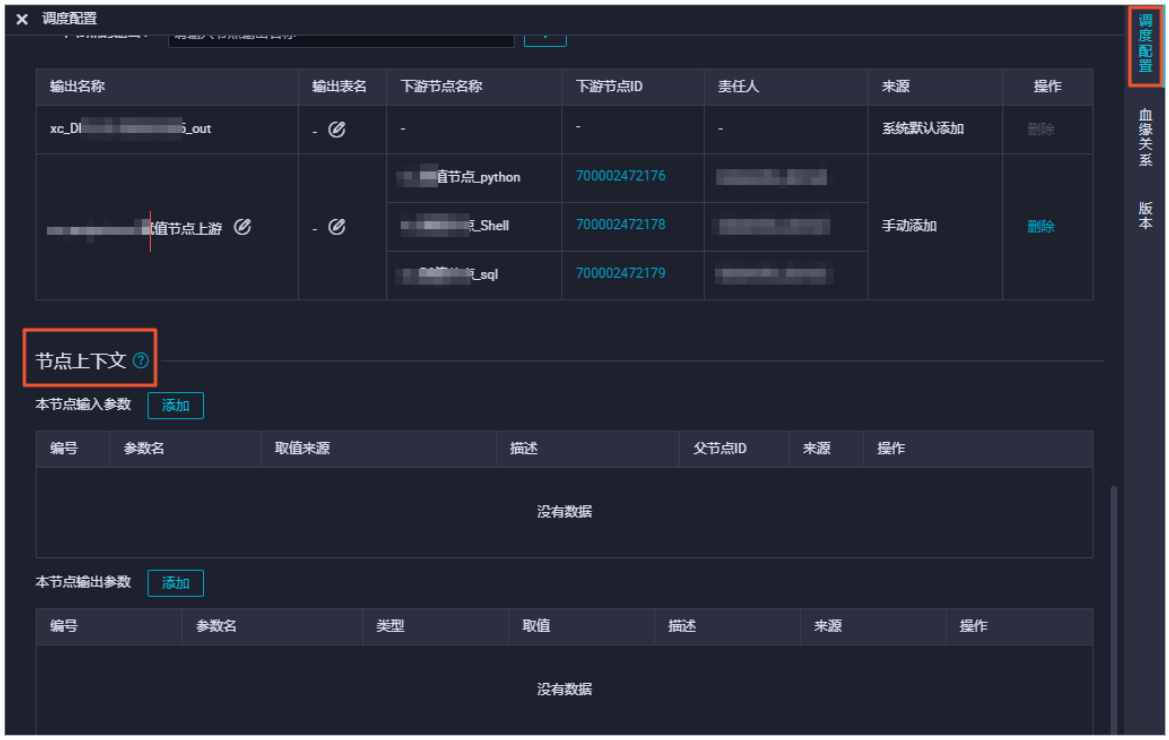
在上游节点定义输出参数及其取值后，在下游节点定义输入参数（取值引用上游节点的输出参数），即可在下游节点中使用此参数获取上游节点传递过来的取值。

注意事项

节点上下文参数仅用于上游节点的**节点上下文**输出参数作为下游节点的**节点上下文**输入参数，无法直接将上游节点的查询结果传递到下游，如果您需要将上游节点的查询结果传递到下游节点，可以使用赋值节点，详情可参见文档：赋值节点。部分节点支持赋值参数的功能，赋值参数使用与赋值节点行为大体一致，您可以参考赋值节点使用文档来进行**赋值参数**的配置，关于赋值参数的配置可参考下文：赋值参数。

配置节点上下文

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**进入数据开发**。
4. 在**数据开发**面板，双击打开相应节点的编辑页面。
5. 单击右侧的**调度配置**，在**节点上下文**区域配置**本节点输入参数**和**本节点输出参数**。
   各类型节点的调度参数配置，详情请参见SQL类型节点及离线同步节点。

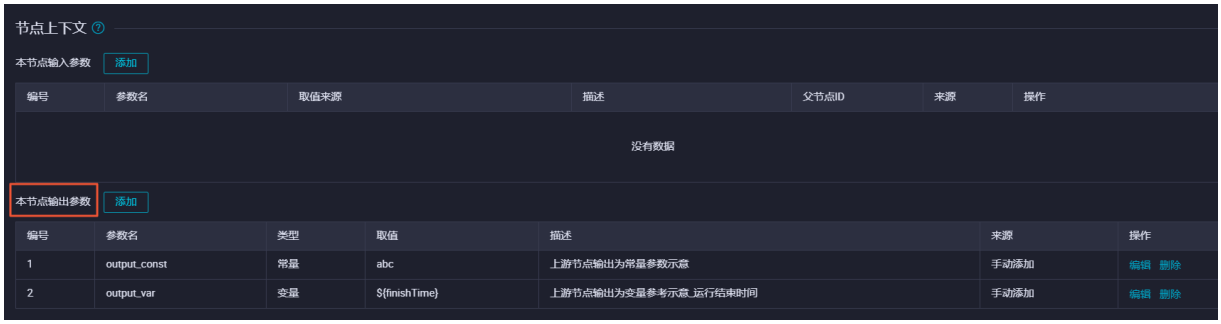


输出参数

您可以在节点上下文中定义**本节点输出参数**，输出参数的取值分为**常量**和**变量**两种类型。

完成输出参数的定义并提交当前节点后，即可在下游节点中引用，作为下游节点的输入参数的取值。

> **说明** 不支持在当前节点编写代码的方式，对定义的输出参数进行赋值。



字段	含义	描述
编号	编号为系统控制，自动增加。	无

字段	含义	描述
参数名	定义的输出参数名称。	无
类型	参数类型。	包括**常量**和**变量**。
取值	输出参数的取值。	取值类型包括常量和变量： • 常量为固定字符串。 • 变量包括系统支持的全局变量、调度内置参数、自定义参数${...}和自定义参数$[...]。
描述	参数的简要描述。	无
来源	当前参数的来源。	包括**系统默认添加**、**自动解析**和**手动添加**。
操作	提供**编辑**和**删除**两种操作。	当存在下游节点依赖时，不支持编辑和删除。在下游节点添加对上游节点引用之前，请谨慎检查，确保上游输出定义正确。

赋值参数

如果您需要将一个任务的查询结果作为参数传递给下游任务进行引用，请在上游节点（目前支持的节点类型）的编辑页面，单击右侧的**调度配置**。在该节点的**节点上下文 > 本节点输出参数**区域，单击**添加赋值参数**，一键添加输出的赋值参数。目前支持的节点包括EMR Hive、EMR Spark SQL、ODPS Script、Hologres SQL、AnalyticDB for PostgreSQL和MySql节点等。

> **说明** 您需要购买DataWorks标准版及以上版本，才可以使用**添加赋值参数**功能。

您单击**添加赋值参数**后，赋值参数会传递上游节点生产的查询结果。如果产生结果为空，不会阻塞本节点运行，但下游引用的节点可能会失败。

下游节点需要在输入参数中添加上游节点的赋值参数，在代码中通过二维数组的方式引用。其使用方式和赋值语言为ODPS SQL的赋值节点一致，详情请参见赋值节点。

输入参数

节点的输入参数用于定义对其依赖的上游节点的输出的引用，并可以在节点内部使用，使用方式与其它参数一致。

输入参数的定义和使用如下：

- 定义输入参数
  i. 在**调度依赖**中添加依赖的上游节点。

  ii. 在**节点上下文**中，添加输入参数定义，取值选择引用上游节点的输出参数。

字段	含义	说明
编号	编号为系统控制，自动增加。	无
参数名	定义的输入参数名称。	无
取值来源	参数的取值来源，引用上游节点取值。	取值是上游节点输出参数的具体取值。
描述	参数的简要描述。	无
父节点ID	父节点ID。	自动从上游节点解析得到。
来源	当前参数的来源。	包括**系统默认添加**、**自动解析**和**手动添加**。
操作	提供**编辑**和**删除**两种操作。	无

- 使用输入参数

在节点中使用定义的输入参数的方法和其它系统变量一致，引用方式为 `${输入参数名}` 。下图为在Shell节点中进行引用。

```
#!/bin/bash
#********************************************************************#
##author:
##create time:2019-04-16 18:13:52
#********************************************************************#

echo 'input_from_up_const:' ${input_from_up_const}
echo 'input_from_up_var:' ${input_from_up_var}

```

系统支持的全局变量

- 系统变量

系统变量	说明
${projectId}	项目ID。
${projectName}	MaxCompute项目名。
${nodeId}	节点ID。
${gmtdate}	实例定时时间所在天的00:00:00，格式为yyyy-MM-dd 00:00:00。
${taskId}	任务实例ID。
${seq}	任务实例序号，代表该实例在当天同节点实例中的序号。
${cyctime}	实例定时时间。
${status}	实例的状态：成功（SUCCESS）、失败（FAILURE）。
${bizdate}	业务日期。
${finishTime}	实例结束时间。
${taskType}	实例运行类型：正常（NORMAL）、手动（MANUAL）、暂停（PAUSE）、空跑（SKIP）、未选择（UNCHOOSE）、周月空跑（SKIP_CYCLE）。
${nodeName}	节点名称。

- 其它参数设置请参见调度参数概述。

9.7. 常见问题

背景信息

依赖的父节点输出名不存在

提交报错依赖的父节点输出名不存在。

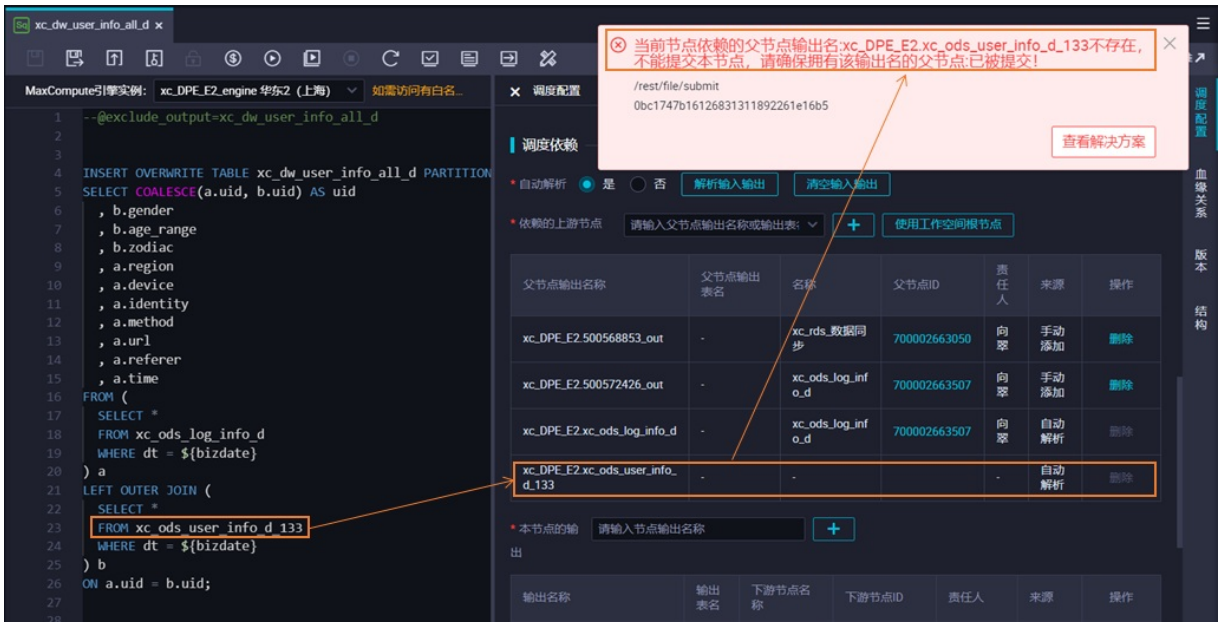
1.

节点输出相同

9.7.1. 提交节点报错：当前节点依赖的父节点输出名不存在

问题现象

提交节点时，系统报错，提示当前节点依赖的父节点输出名不存在。



以上图为例，出现此类报错说明，系统无法通过本节点配置的这条**父节点输出名称**的依赖关系，找到产出表 `xc_ods_user_info_d_133` 的上游节点。

> **说明** 出现此报错，说明调度依赖配置里的**节点输出名不存在**（即没有节点将这个节点输出名配置为本节点的输出），并不是指表不存在。如果表存在，且由某个节点产出，但是没有将这个表添加为节点的输出，也会出现此类报错。

可能原因1：没有节点产出这个表

- 可能原因

  出现此种情况的原因之一是：确实没有节点产出这个表。

  对于大部分场景，DataWorks可通过自动解析，自动将产出表添加为节点的本节点的输出，但是对于非周期性生成的表，不支持使用自动解析。非周期性调度生产数据的表包括：

  - 从本地上传到DataWorks的表
  - 维表
  - 非DataWorks调度产出的表
  - 手动任务产出的表

当有节点SELECT非周期性调度生成数据的表时，就会出现上述报错。

- 解决方案

  您需要手动删除包含非周期性生成数据的表相关的依赖配置。本示例中，即您需要手动将**父节点输出名称**为 `xc_ods_user_info_d_133` 的调度依赖配置删除。

  手动删除调度依赖的具体操作可参见调度依赖配置指导：手动配置。

可能原因2：有节点产出该表数据，但是该表没有添加为该节点的输出

- 可能原因

  出现此种情况的另外一个可能的原因是：有节点产出该表数据，但是该表没有添加为该节点的输出。

  对于大部分场景，DataWorks可通过自动解析，自动将产出表添加为节点的本节点的输出，但是对于一些特殊的节点，DataWorks不支持使用自动解析。离线节点、AnalyticDB for PostgreSQL节点、AnalyticDB for MySQL节点、EMR节点不支持通过自动解析添加节点的调度依赖，这些节点的产出表需要手动添加为本节点的输出。

  当有节点SELECT这类节点生成的表，且产出这个表的没有手动添为节点的产出时，就会出现上述报错。

- 解决方案

  您需要手动将表添加为产出该表节点的输出。本示例中，即您需要手动将 `xc_ods_user_info_d_133` 添加为产出这个表的**本节点输出**。

  手动添加调度依赖的具体操作可参见调度依赖配置指导：手动配置。

为了避免依赖关系配置错误导致数据出现问题，DataWorks会在提交节点时进行表数据血缘关的输入输出和调度配置输入输出比较，如果不一致会给您提示，详情可参见提交节点时提示：输入输出和代码血缘分析不匹配。

不是DataWorks每天调度产出的表数据，是不需要设置节点依赖关系的，这类表依赖可以删除，删除后提交节点时，会出现血缘关系与调度依赖配置不一致的提示，您可确认一下，是否除了删除的非周期性生成数据表的依赖关系外，没有其他血缘与调度配置不一致的地方，没有的话可以强制提交节点。

可能原因3：存在同名的节点输出

- 可能原因

  出现此种情况的另外一个可能的原因是：有多个节点的**本节点产出**名称一样。此原因可能由两种场景造成：

  - 有多个节点产出了同一张表。

    如果有多个节点产出了同一个表，当有节点SELECT这张表时，系统无法找到准确且唯一的产出这张表的节点，提交节点时会出现上述报错。

  - 同个项目中存在同名的节点。

    由于节点创建时，系统会自动为节点添加两个**本节点产出**，其中一个名称命名规则为 `projectname.nodename`，如果同一个项目下如果有两个同名节点，这两个节点自动生成的**本节点产出**名称一样，提交会出现上述报错。

- 解决方案

  需严格按照代码开发规范和界面命名建议进行整改：

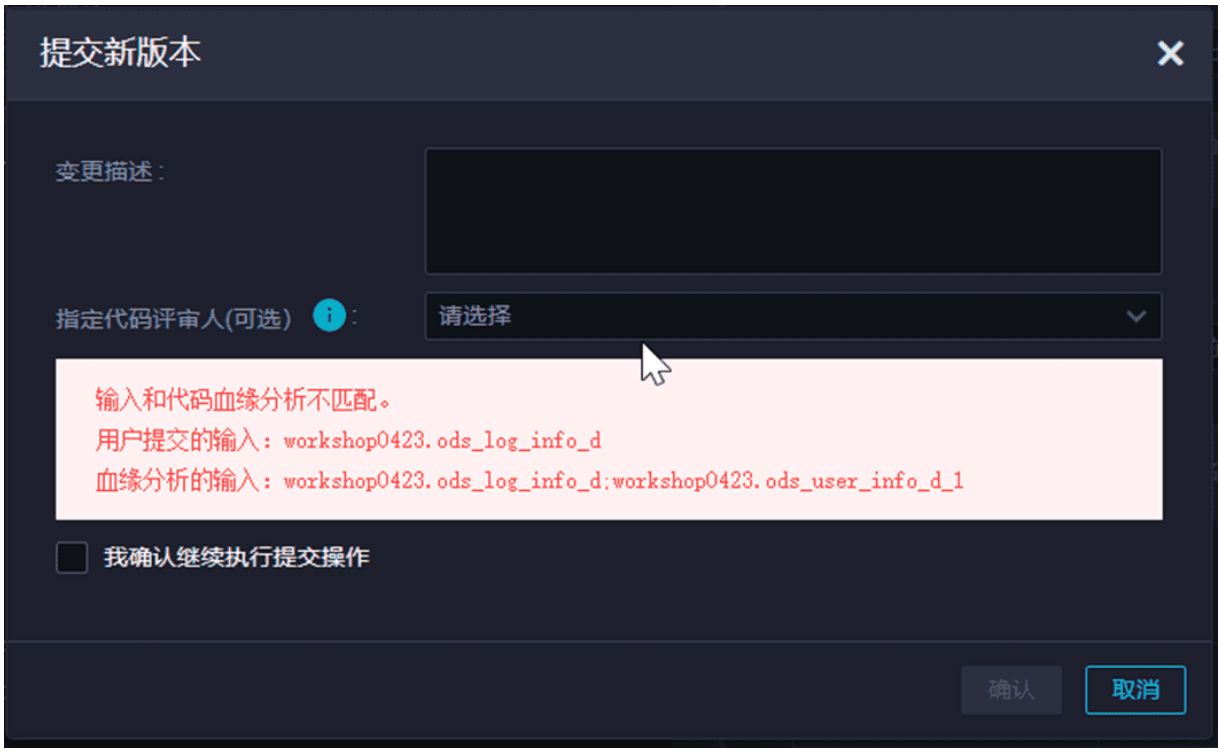
  - 一张表由一个节点产出，节点的产出表需添加为本节点的产出。
  - 同项目中的节点命名不重复。

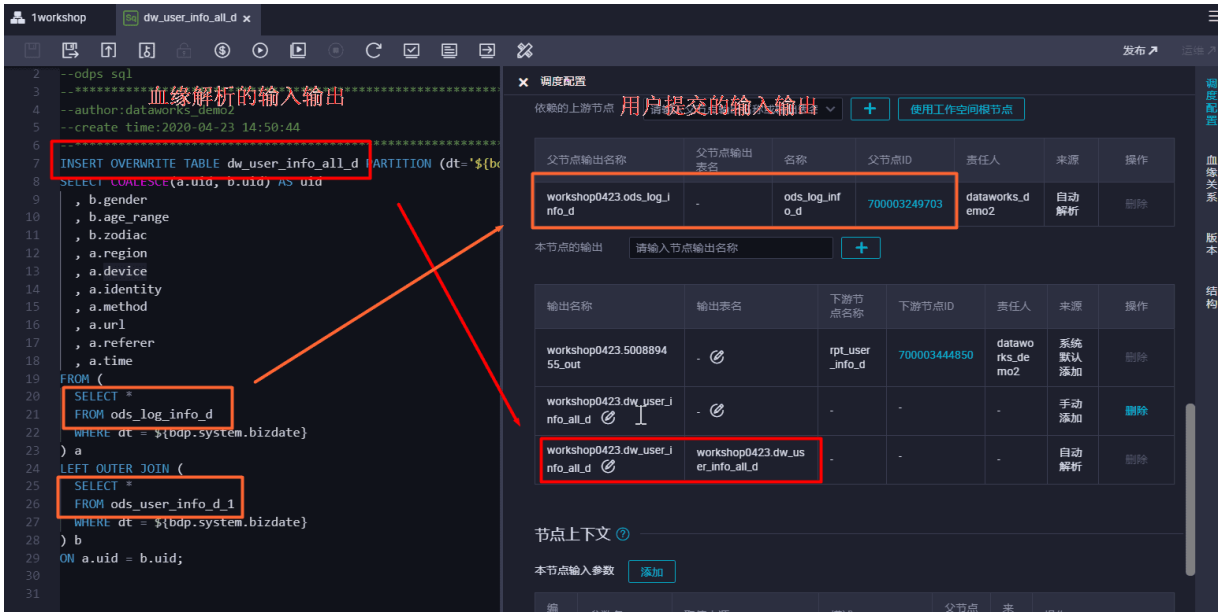
  整改后，需确保不存在多个节点的**本节点产出**名称一样。

9.7.2. 提交节点时提示：输入输出和代码血缘分析不匹配

问题现象

提交节点时，系统出现提示：输入输出和代码血缘分析不匹配。





可能原因

当代码中SELECT的表与节点的依赖的父节点配置不一致，或代码中INSERT、CREATE的表与节点的本节点的输出不一致时，会出现该提示。

以上图为例，说明：

- 您提交的节点代码中有SELECT名称为table2的数据，但是table2并没有配置为节点的依赖的父节点。
- 您提交的节点中有将doc_test配置为节点的本节点的输出，但是节点代码中并没有INSERT或CREATE名称为doc_test的表。

解决方案

- 非周期性生成数据的表可以忽略提示直接提交。

  由于DataWorks的调度依赖主要保障调度节点定时更新的表数据，所以非DataWorks平台上调度更新的表，平台无法监控。当节点代码中SELECT非周期性调度生产的表数据时，您需要删除通过SELECT自动生成的依赖的上游节点配置。非周期性调度生产数据的表包括：

  - 从本地上传到DataWorks的表
  - 维表
  - 非DataWorks调度产出的表
  - 手动任务产出的表

- 对于周期性生成数据的表，您需要仔细检查表数据的血缘关系与调度依赖关系是否一致。

  如果您不检查直接强制提交节点，可能会导致以下影响：

  - 例如，代码中SELECT一张表A，并且表A是个调度节点每天定时产出的表（即表A不是非周期性生成数据的表），如果没有将表A添加为本节点的依赖的父节点，形成依赖关系的时，某次生成表A的节点没有执行成功的话，下游节点取表A的数据即取用的表A上一次运行结果的数据，可能会有问题。
  - 例如，在代码中CREATE或INSERT一张表B，没有将表B作为本节点的输出，则如果有节点SElECT表B，自动解析会自动将表B作为节点的输入，形成依赖关系，但是系统无法通过这个依赖关系找到产出表B的节点，提交节点时会报错：当前节点依赖的父节点输出名不存在。详情可参见提交节点报错：当前节点依赖的父节点输出名不存在。

10.调试及提交发布任务

10.1. 调试与查看任务

10.1.1. 调试代码片段：快捷运行

DataWorks的快捷运行功能，帮助您在节点编辑页面，快速运行选中的代码片段。您可以通过该功能测试代码片段编写是否正确。本文为您介绍如何快捷运行目标代码。

前提条件

已创建ODPS SQL节点并编写任务代码，详情请参见创建ODPS SQL节点。

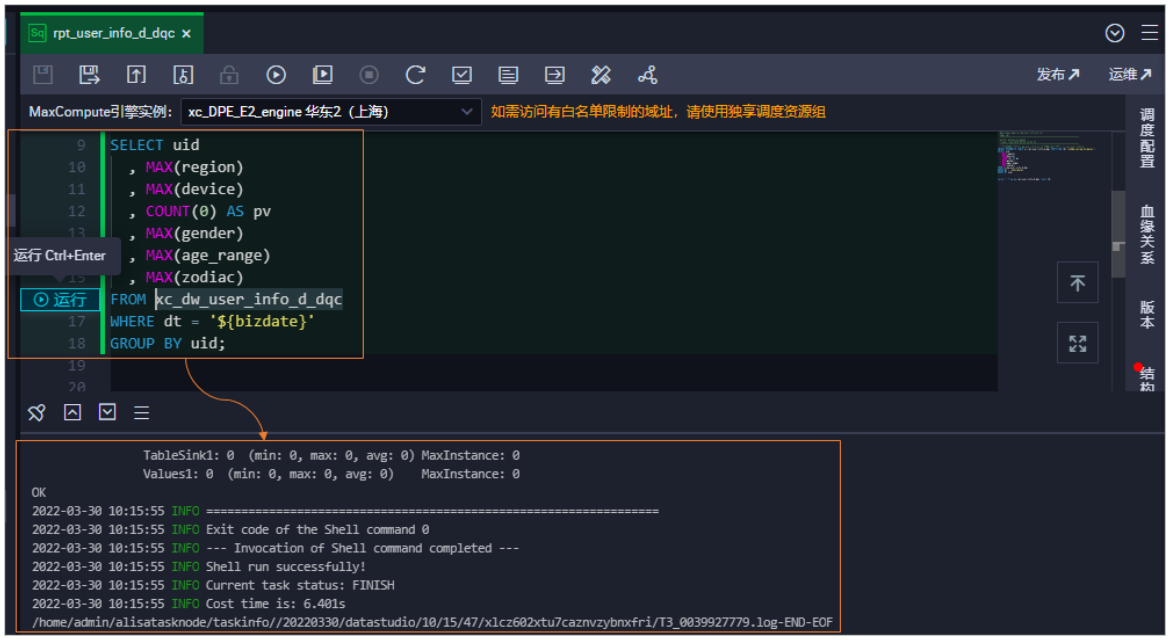
使用限制

- 目前仅支持ODPS SQL节点使用快捷运行功能。
- 仅非运行状态的节点支持使用该功能。如果节点的任务代码处于运行状态，则在代码行左侧将不会显示快捷运行（运行）图标。

操作步骤

1. 进入数据开发。
   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**数据开发**，进入**DataStudio**页面。
2. 在**数据开发**或**手动业务流程**功能模块的目录树，或通过**临时查询**功能，查找目标节点，双击进入节点编辑页面。

   数据开发及**手动业务流程**目录树结构介绍，详情请参见数据开发功能索引。使用临时查询，详情请参见创建临时查询。
3. 快捷运行所选代码并查看运行结果。



i. 选中目标代码。

在节点编辑页面的SQL代码区域，鼠标定位至目标代码行，系统会自动识别该行代码所属的完整代码片段。

ii. 运行代码。

> 说明
> - 仅非运行状态的节点支持使用该功能。如果节点的任务代码处于运行状态，则在代码行左侧将不会显示快捷运行（运行）图标。
> - 快捷运行功能使用的资源组说明如下：
>   - 快捷运行当前代码片段所使用的资源组，为最近一次运行（包括快捷运行运行、运行、高级运行）节点代码时使用的资源组。
>   - 若当前节点为首次运行，则您需要根据业务情况选择所使用的调度资源组。如果没有合适的资源组，您可以参考新增和使用独享调度资源组新建。
>   - 若您需要修改当前节点运行时使用的资源组，则请使用高级运行功能。
> - 快捷运行的代码片段如果包含变量，则首次运行时，您需要为变量赋值，赋值后，系统会保存变量的此次赋值。后续运行中，如果您需要修改变量的赋值，则请使用高级运行功能。更多变量的赋值详情，请参见调度参数概述。

您可以通过如下两种方式运行代码：

- 单击代码行左侧快捷运行（运行）图标。
- 使用快捷键运行。
  - Windows系统：`Ctrl + Enter`。
  - Mac系统：`Cmd + Enter`。

运行完成后，您可以根据运行结果判断目标代码是否符合预期，及时修正有误代码。

10.1.2. 创建临时查询

临时查询用于在本地测试代码的实际情况与期望值是否相符或排查代码错误。

背景信息

临时查询无需提交、发布和设置调度参数。如果您需要使用调度参数，请在**数据开发**或**业务流程**中创建节点。

新建文件夹

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. 在左侧导航栏，单击**临时查询**。

您可以单击左下方的图标，展开或折叠左侧导航栏。

3. 鼠标悬停至+新建图标，单击**文件夹**。
4. 在**新建文件夹**对话框中，输入**文件夹名称**，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。
> - DataWorks支持存在多级文件夹目录，您可以保存当前文件夹至其它已创建好的文件夹中。

5. 单击**提交**。

创建临时查询节点

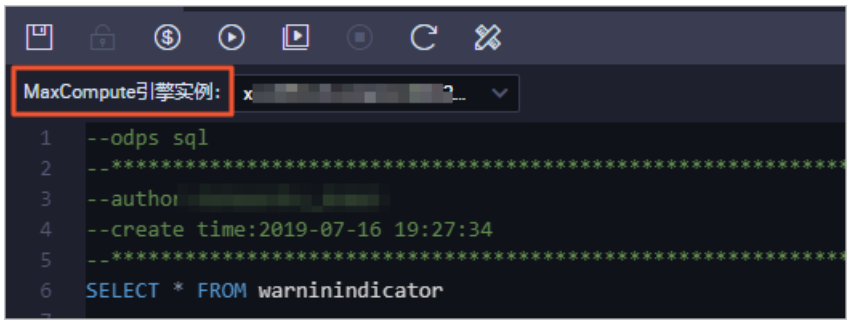
您可以在**临时查询**页面新建EMR Hive、ODPS SQL、EMR Spark SQL、EMR Presto、EMR Impala、Shell、AnalyticDB for PostgreSQL、AnalyticDB for MySQL和Data Lake Analytics节点。

本文以新建ODPS SQL节点为例：

1. 在**临时查询**页面，右键单击文件夹名称，选择**新建节点 > ODPS SQL**。
2. 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。

3. 单击**提交**。
4. 在节点的编辑页面，输入SQL查询语句。



> **说明**
> - 如果当前工作空间绑定多个MaxCompute计算引擎，请选择需要的MaxCompute引擎实例后，再运行查询语句。
> - 如果您选中的MaxCompute引擎实例使用的是按量计费默认资源组，则可以在运行语句前，单击工具栏中的图标，预估此次运行产生的费用（实际费用请以账单为准）。

5. 单击工具栏中的图标，查看运行结果。

10.1.3. 运行历史

运行历史面板为您展示最近三天您在数据开发界面运行过的所有任务记录，单击相应的任务，即可查看运行日志。

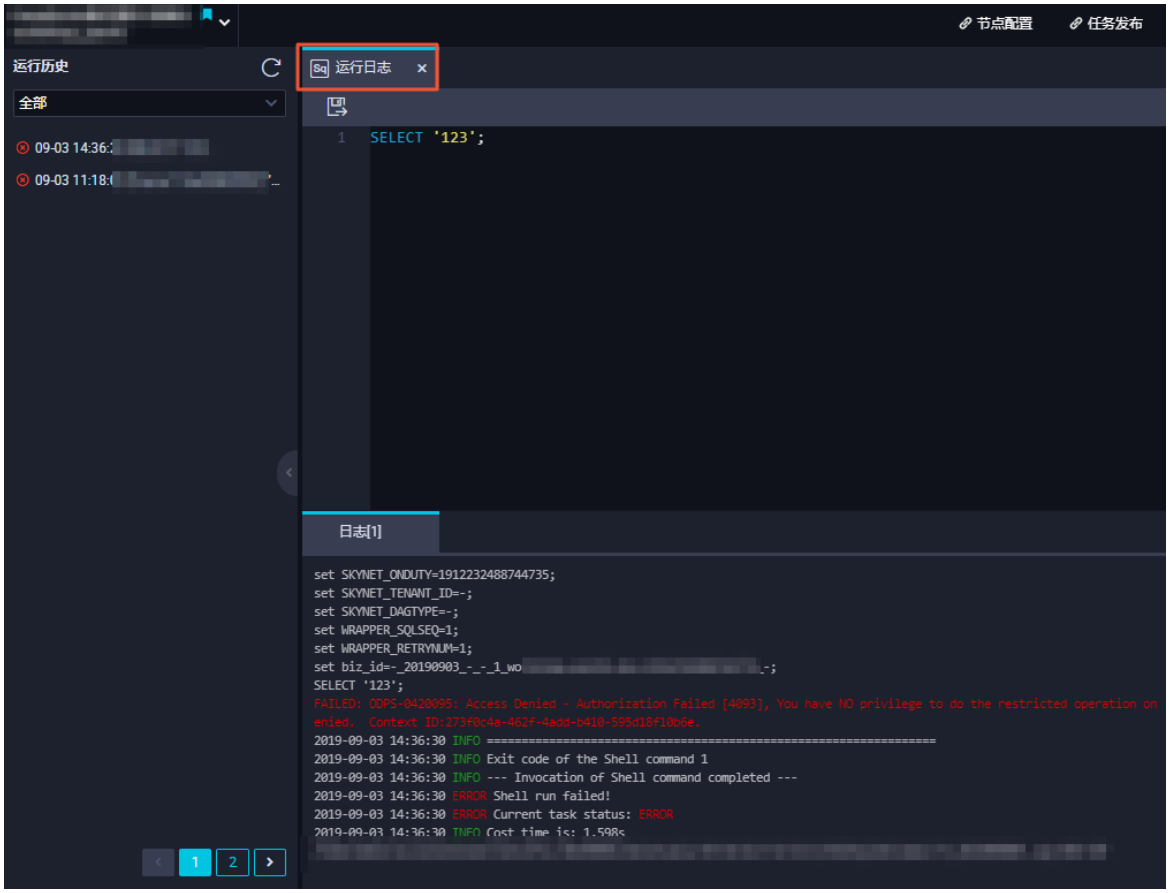
> **说明** 此界面只展示个人最近三天的运行历史。

查看运行历史

1. 登录DataWorks控制台，单击相应工作空间后的**进入数据开发**。
2. 单击左侧导航栏中的**运行历史**，切换至运行历史面板（默认展示全部状态）。

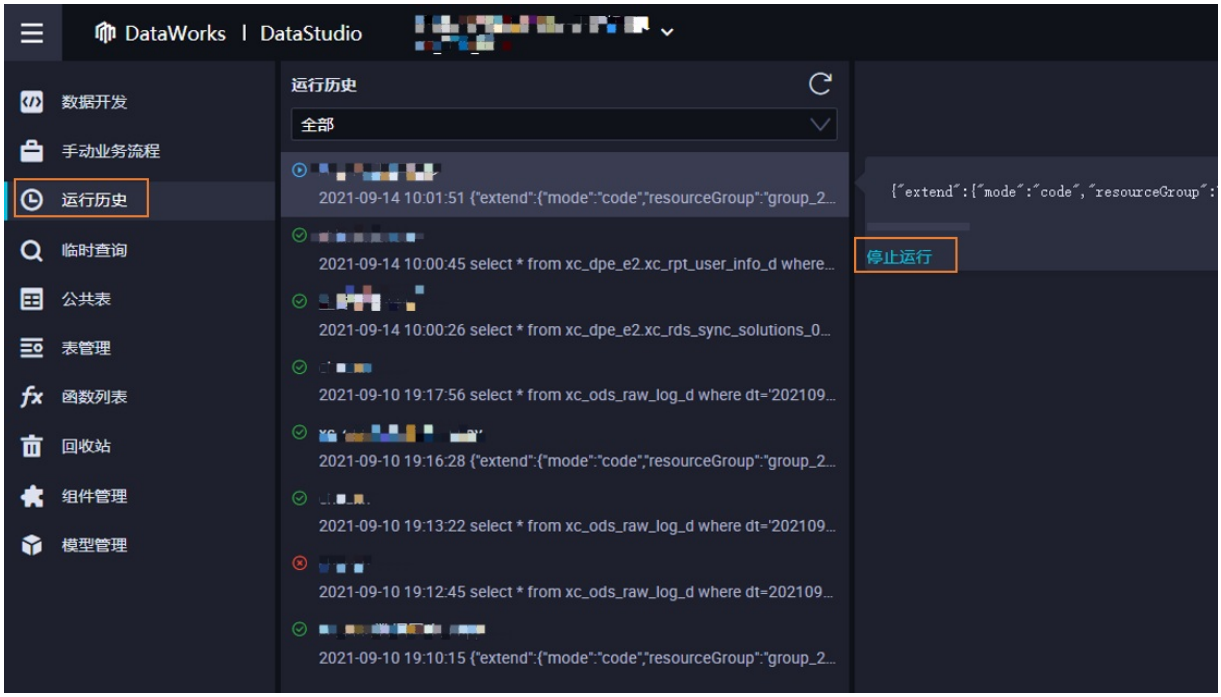
从状态列表中，选择需要查看的相关状态的任务。

3. 单击需要查看的运行记录，即可在右侧查看运行日志。



停止运行中的任务

您可以过滤运行中的节点，找到需要停止运行的任务，中止该任务运行。



另存为临时文件

如果需要保存运行记录中的SQL语句，您可以单击**保存**，将运行过的SQL记录另存为临时文件。

在**新建节点**对话框中，选择**节点类型**和**目标文件夹**，并填写**节点名称**后，单击**提交**即可。

10.1.4. 使用流程参数

当整个业务流程需要对同一个变量统一赋值或替换其参数值时，您可以使用流程参数功能。本文以替换手动业务流程中所有的ReplaceMe参数为ReplaceMe123，为您介绍如何使用流程参数。

使用限制

- 手动业务流程中的ODPS SQL、Shell和数据同步节点支持全局参数，且需要使用特定的格式。例如，全局参数为x=y1，不同类型节点的替换及引用方式如下：
  - 对于ODPS SQL节点，需要双击打开节点后，单击右侧导航栏的**属性**。在**基础属性**中输入参数x=aaa，节点在执行时才会正确替换为x=y1。代码中以${x}的方式来进行引用。
  - 对于Shell节点，需要双击打开节点后，单击右侧导航栏的**属性**。在**基础属性**中输入参数$x，节点在在执行时才会正确替换为y1。代码中以$1的方式来进行引用。
  - 对于数据同步节点，需要双击打开节点后，单击右侧导航栏的**属性**。在**基础属性**中输入参数-p"-Dx=aaa"，节点在在执行时才会正确替换为-p"-Dx=y1。代码中以${x}的方式来进行引用。
- 调度的业务流程仅支持ODPS SQL节点使用流程参数。
- 使用流程参数前，请先配置好单个节点的参数，确保单个节点运行无误。
- 当流程参数的赋值与单个节点参数的赋值不一致时，流程参数的赋值会覆盖节点的参数赋值。
- 输入参数时，请注意区分大小写。

配置流程参数

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. 在数据开发页面左侧导航栏，单击**手动业务流程**。
3. 在**手动业务流程**目录树，双击目标手动业务流程，进入手动业务流程编辑页面。
4. 在手动业务流程编辑页面，单击右侧导航栏的**流程参数**。
5. 在**流程参数**对话框中，输入**参数名称**为ReplaceMe，**参数值或表达式**为ReplaceMe123。

6. 单击手动业务流程编辑页面工具栏的图标，保存该配置。

ODPS SQL节点获取流程参数

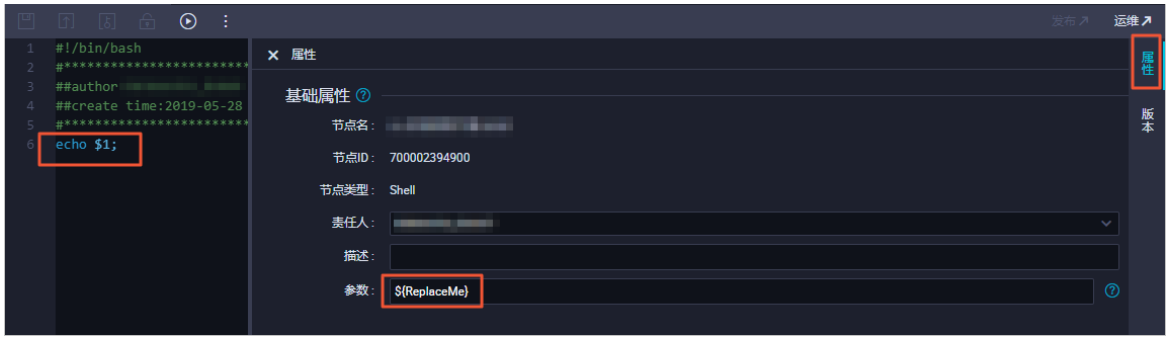
1. 在数据开发页面左侧导航栏，双击**手动业务流程**。
2. 进入**手动业务流程**的编辑页面，双击目标ODPS SQL节点，进入节点的编辑页面。
3. 在节点编辑页面，单击右侧导航栏的**属性**，在**参数**区域替换原参数ReplaceMe为ReplaceMe=123。

由于流程参数ReplaceMe=ReplaceMe123，所以运行整个业务流程时，该节点赋值为ReplaceMe123。

4. 在手动业务流程的编辑页面，单击工具栏中的图标，保存配置。

Shell节点获取流程参数

1. 在**数据开发**页面的左侧导航栏，单击**手动业务流程**。
2. 展开相应手动业务流程下的**数据开发**，双击打开Shell节点的编辑页面。
3. 单击右侧的**属性**，输入**参数**为${ReplaceMe}。

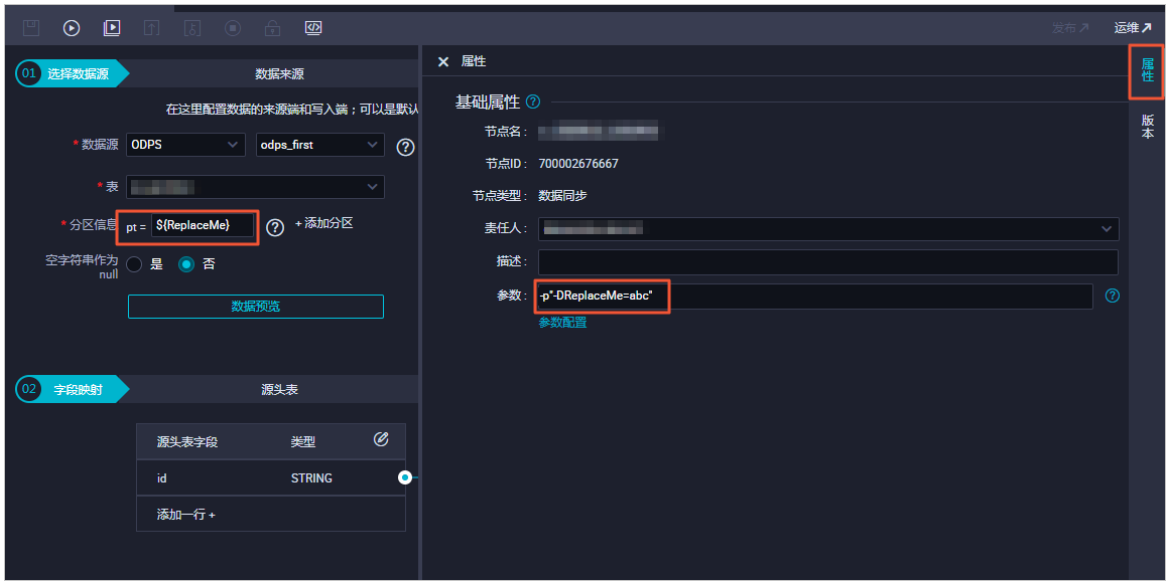


> **说明** 请注意Shell节点的参数定义和赋值。

4. 在手动业务流程的编辑页面，单击工具栏中的图标。

数据同步节点获取流程参数

1. 在**数据开发**页面的左侧导航栏，单击**手动业务流程**。
2. 展开相应手动业务流程下的**数据开发**，双击打开数据同步节点的编辑页面。
3. 单击右侧的**属性**，输入**参数**为-p"-DReplaceMe=abc"。



此处数据集成参数配置为ReplaceMe=abc，流程参数为ReplaceMe=ReplaceMe123，运行整个业务流程时，流程参数的赋值ReplaceMe=ReplaceMe123替换了代码中的ReplaceMe，所以pt="ReplaceMe123"，流程参数会覆盖节点中的ReplaceMe的赋值。

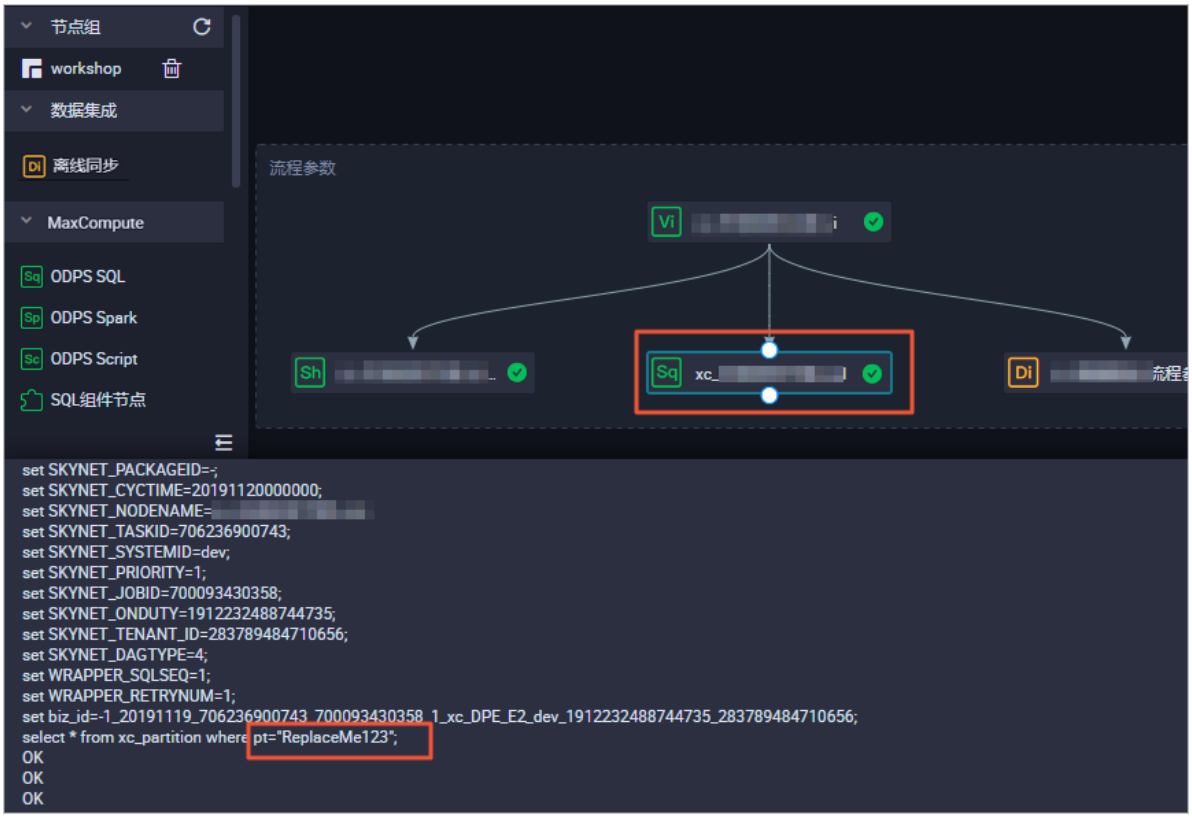
> **说明** 数据同步节点的参数格式为-p"-D参数名=参数值"。

4. 在手动业务流程的编辑页面，单击工具栏中的图标。

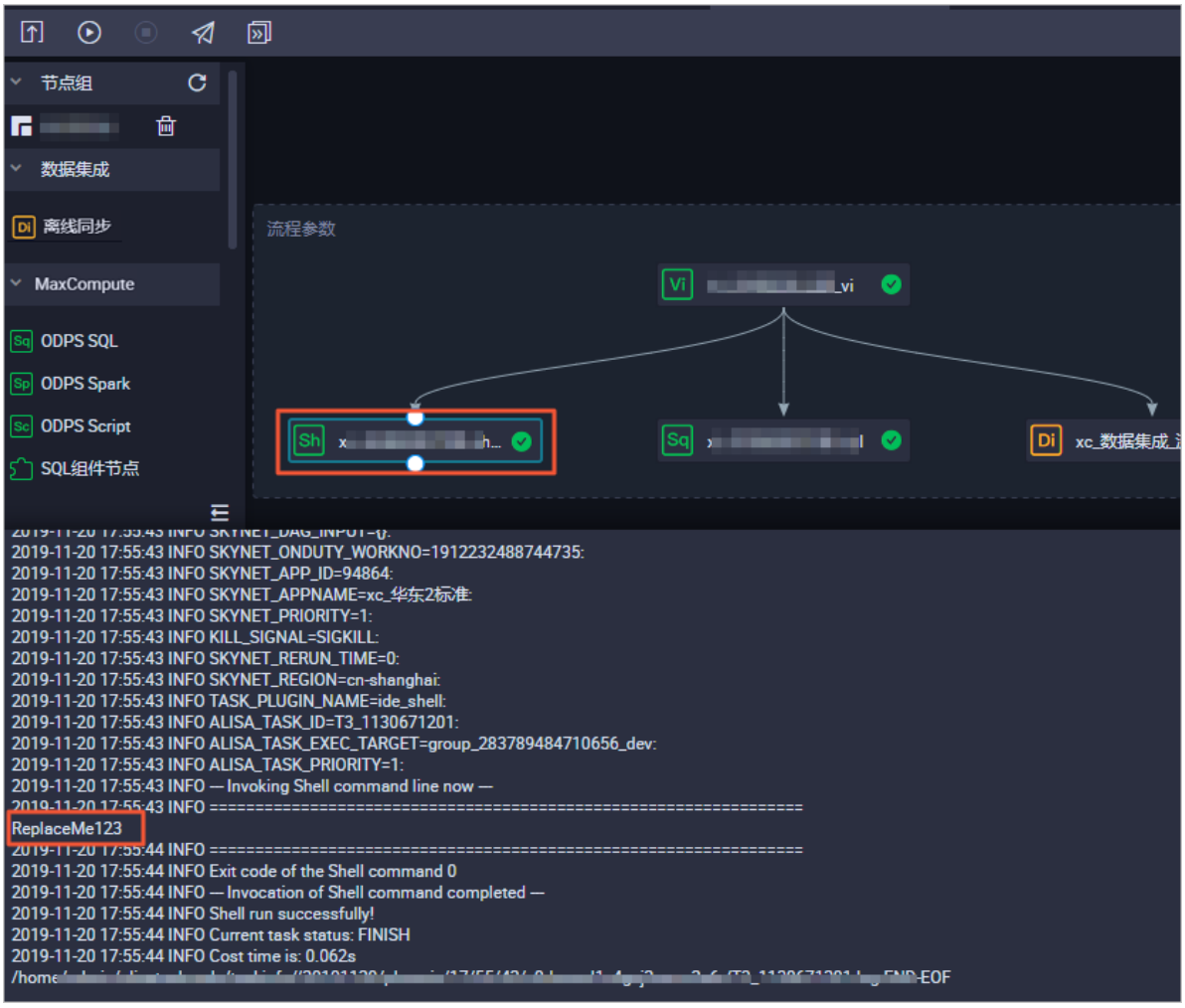
运行并查看结果

在手动业务流程的编辑页面，单击工具栏中的图标。调度任务时，各节点的赋值才会替换为流程参数。所以在界面运行手动业务流程时，您需要在**填写参数**对话框中，为变量ReplaceMe赋值：

- 右键单击ODPS SQL节点，单击**查看日志**，即可在页面下方查看ODPS SQL节点的赋值情况。

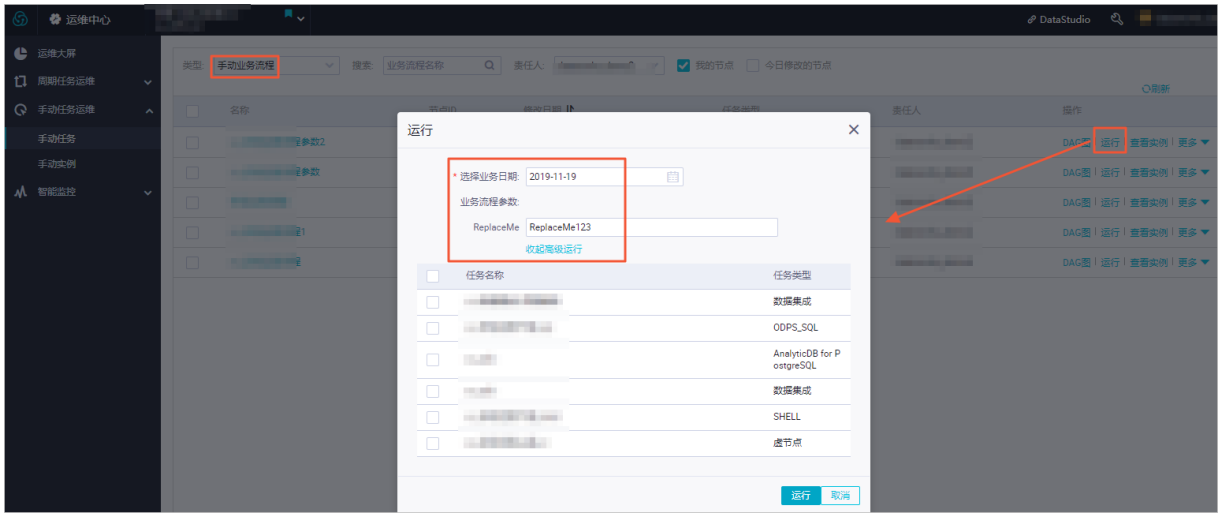


- 右键单击Shell节点，单击**查看日志**，即可在页面下方查看Shell节点的赋值情况。



- 右键单击数据同步节点，单击**查看日志**，即可在页面下方查看数据同步节点的赋值情况。

如果您未在手动业务流程编辑页面右侧的**流程参数**中赋值，则每次在生产环境运行该业务流程时，都需要手动给业务流程参数赋值。



10.1.5. 执行冒烟测试

完成代码开发后，您需要调试运行。本文为您介绍如何在开发环境进行冒烟测试。

背景信息

为保障调度节点任务执行符合预期，建议您在发布前对任务进行冒烟测试。在使用调度参数的场景下，您也可以通过冒烟测试校验调度场景下的参数替换情况。

进入数据开发

1. 进入数据开发。
   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**数据开发**，进入DataStudio页面。

进入冒烟测试

在代码开发完成后，您可在任务发布生产调度前通过以下几种方式进行冒烟测试。

- 在提交节点时，在**数据开发**节点编辑界面的上方工具栏单击图标，在单选按钮**冒烟测试**后选择**是**。
- 在提交节点后，在**数据开发**节点编辑界面的上方工具栏单击图标，执行冒烟测试。
- 在提交节点后，在**任务发布**界面单击相应节点后的**冒烟测试**按钮。
- 在**运维中心**页面左上角切换到开发环境运维中心后，在左边导航栏单击**周期任务运维 > 周期任务**。然后右键单击相应节点，在提示框中单击**测试**。

配置冒烟测试参数

通过上述入口进入冒烟测试界面后，您需要在界面配置中选择冒烟测试的业务日期。

> **说明** 冒烟测试选择业务时间为今天或者昨天时，冒烟测试任务将等待定时时间到达后才会执行。示例：当前日期为2022/06/02 12：00，任务定时时间为15：00，若选择业务时间为2022/06/01日，此时冒烟测试任务将由于定时时间未到15:00而出现等待时间的情况。

查看冒烟测试记录

1. 在**数据开发**节点编辑界面的上方工具栏单击图标，进入**查看冒烟测试记录**页面。

   > **说明**
   > - 提交节点时，在单选按钮**冒烟测试**后选择**是**，可在提交后通过上述方式查看冒烟测试过程。
   > - 在节点编辑界面点击**冒烟测试**按钮后，可以在配置冒烟测试参数后的弹窗中查看冒烟测试记录。
   > - 若您在数据开发界面左侧无法看到该入口，您可在设置页面添加该模块，详情请参见：改变布局：定制化展示模块。

2. 您可在**冒烟测试记录**页面查看**测试时间**、**版本**、**测试人**、**业务日期**等信息。单击**在DataStudio发起**按钮，可以快速定位在DataStudio界面触发的冒烟测试记录。

   > **说明** DataStudio界面触发的冒烟测试记录勾选后，查询结果中将不包括在开发环境运维中心执行的测试记录。

3. 您可在该页面通过点击**查看日志**，查看具体冒烟测试详细执行日志。**状态**为**运行中**的记录可通过单击**停止**按钮终止运行。

10.2. 提交与发布任务

10.2.1. 代码评审

DataWorks提供代码评审功能，开启强制代码评审开关后，开发人员提交的节点必须通过评审人对代码的审核才可以发布。

前提条件

代码评审限DataWorks专业版及以上版本使用。标准模式的工作空间支持开发者自主选择代码评审或由管理员开启强制代码评审，简单模式的工作空间仅支持开发者自主选择代码评审。

使用限制

DataWorks的简单模式工作空间不支持代码评审功能，该功能只适用于标准模式的工作空间。

开启强制代码评审

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. 单击左下方的图标，在右侧展开**设置**页面。
3. 在**设置**页面，单击**工作空间配置**。
4. 在**代码评审配置**区域，打开**启用强制代码评审**开关，并**指定强制代码评审基线范围**。

代码评审的基线范围包括**1级基线任务**、**3级基线任务**、**5级基线任务**、**7级基线任务**、**8级基线任务**及**非基线任务**，其相关说明如下：

- 任务的数值越大优先级越高，基线任务的优先级高于非基线任务。配置基线任务的优先级，详情请参见基线管理。
- 当**指定强制代码评审基线范围**指定了对应级别的基线任务，该级别的任务在发起评审后，必须由评审人员对任务代码审核通过才可进行发布。
- 多个任务并发启动运行时，高优先级的任务会优先抢占资源，此时，如果您对高级别的任务指定了**指定强制代码评审基线范围**，则评审人员可以对任务的代码质量进行把控，防止由于任务代码有误，未经审核直接发布后报错，长时间占用资源。

代码评审流程

1. 在节点或业务流程的编辑页面，单击工具栏中的图标。
2. 在**提交新版本**对话框中，输入**变更描述**并**指定代码评审人**。

> **说明**
>
> - 循环、遍历等组合型节点不支持发起代码评审。
> - 指定代码评审人后，会生成代码评审单。
> - 如果工作空间未启用强制代码评审，代码评审为可选。如果工作空间启用强制代码评审，则代码评审为必选，且评审不通过会阻断任务的发布。

3. 单击**确认**。
4. 发起评审后，您可以单击左上方的图标，选择**全部 > 代码评审**，进入**代码评审**页面。

5. 在**代码评审**页面，审核人可以评审代码，提交人可以查看发起的评审。

   审核人可以进行**评论**、**通过**、**不通过**、**废弃**和**重开**等操作。开启强制评审后，审核人的评审结果会影响节点的发布。其中**通过**操作会触发代码审核检查器检查通过，**不通过**和**废弃**操作会拦截提交人发布节点。您可以在代码审核详情页面对比提交版本和生产版本的代码。

操作	描述
评论	对当前版本进行评论。

操作	描述
通过	通过当前版本的评审。
不通过	不通过当前版本的评审。 **说明** 如果工作空间未开启**启用强制代码评审**，不通过评审仍可以发布。如果工作空间已开启**启用强制代码评审**，不通过评审会阻塞发布。
废弃	废弃当前版本的评审。
重开	废弃后，您可以在**我审核的 > 废弃**页面，重开本次评审。

代码评审页面包括**我审核的**和**我发起的**：

- **我审核的**：您可以查看自己审核的所有评审记录。

单击相应记录后的**查看**，您可以对评审请求进行**评论**、**通过**、**不通过**、**废弃**和**重开**等操作。

- **我发起的**：您可以查看自己提交的所有评审记录。

10.2.2. 任务发布

10.2.2.1. 发布任务

本文为您介绍如何发布标准模式工作空间的节点，以及如何通过跨项目克隆发布简单模式工作空间的节点。

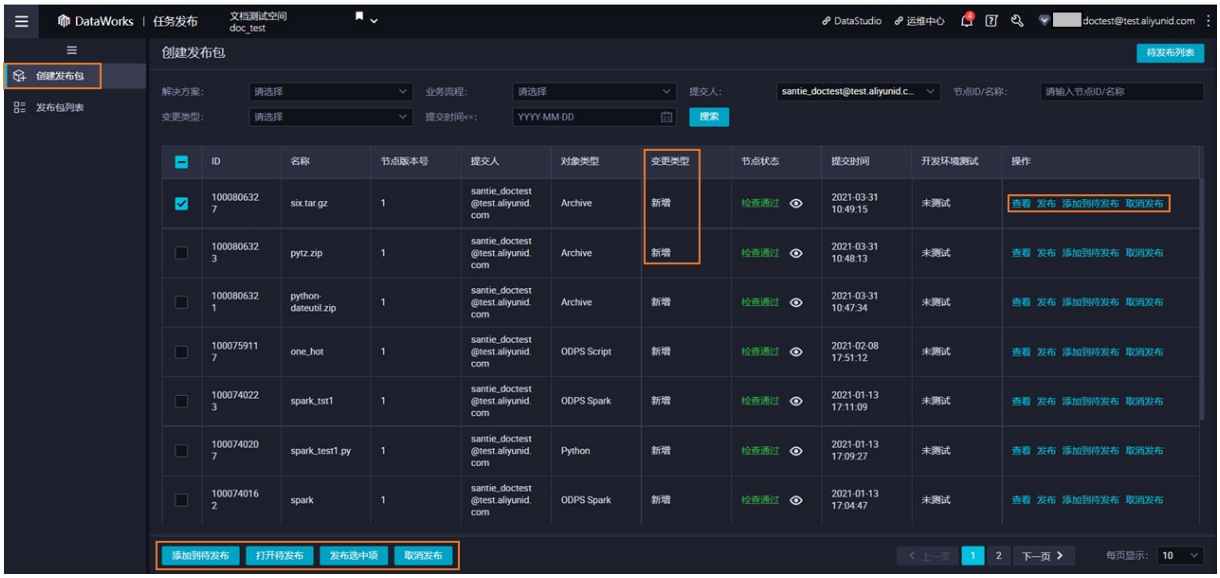
背景信息

在严谨的数据开发流程下，开发者通常会在用于开发的项目内，完成代码开发、流程调试、依赖属性和周期调度属性配置后，再提交任务至生产环境调度运行。

DataWorks的标准模式为您提供在一个项目内，完成从开发到生产的全链路能力，建议您通过该模式完成数据开发与生产发布。详情请参见简单模式和标准模式的区别。

标准模式工作空间下，提交的节点会默认添加至**创建发布包**页面，该页面为您展示已提交的新增、更新、下线节点、资源和函数等操作。

在**创建发布包**页面发布的任务会生成发布包，您可以在**发布包列表**页面查看相关节点的发布记录和状态。



如果您在**数据开发**页面新增、更新、删除的节点、资源和函数，同样在生产环境生效。您需要在**创建发布包**页面发布相关操作至生产环境。您可以在**创建发布包**页面添加单个或多个节点至**待发布列表**，进行批量发布。

创建发布包页面支持修改**每页显示**的条数。

单击相应节点后的**查看**，即可查看当前版本的代码变更及调度配置变更。其中，调度配置相关参数描述如下表所示。

参数	描述
appId	节点所属的DataWorks工作空间ID，您可以进入工作空间配置页面查看ID。详情请参见配置工作空间。
createUser	创建节点的用户ID。
createTime	创建节点的时间。

参数	描述
lastModifyUser	最近一次编辑节点的用户ID。
lastModifyTime	最近一次编辑节点的时间。
owner	节点责任人的ID。您可以进入**调度配置 > 基础属性**页面查看，详情请参见配置基础属性。
startRightNow	节点生成周期实例的方式，取值如下： • 0：表示T+1次日生成。 • 1：表示发布后即时生成。 详情请参见实例生成方式：发布后即时生成实例。
taskRerunTime	节点自动重跑的重跑次数。
taskRerunInterval	节点自动重跑的时间间隔，单位为毫秒。
reRunAble	节点是否可以重跑，取值如下： • 0：表示运行成功后不可重跑，运行失败后可以重跑。 • 1：表示运行成功或失败后均可重跑。 • 2：表示运行成功或失败后皆不可重跑。
startEffectDate	调度生效日期的开始时间。
endEffectDate	调度生效日期的结束时间。
cycleType	调度周期类型，取值如下： • 0表示日、周、月、年调度。 • 1、2、3：表示、分钟小时调度。
cronExpress	调度时间表达式。

extConfig	更多配置信息。JSON格式，包含的关键信息如下：
	• ignoreBranchConditionSkip：是否沿用上一周期空跑属性。
	  ◦ true：沿用上一周期空跑属性。
	  ◦ false：不沿用上一周期空跑属性。
	  详情请参见是否沿用上游的空跑属性。
	• alisaTaskKillTimeout：超时定义，单位为小时。

resgroupId	节点所选的调度资源组ID。详情请参见配置资源属性。
isAutoParse	是否开启自动解析，取值如下： • 1：开启自动解析。 • 0：未开启自动解析。 详情请参见配置同周期调度依赖。

参数	描述

input	节点的输入输出配置，取值如下：
	• str：输入或输出的值。
	• refTableName：输出表。
	• parseType：输入输出的添加方式。取值如下：
	  ○ 0：自动解析。
	  ○ 1：手动添加。
	  ○ 2：系统生成。
	详情请参见同周期调度依赖逻辑说明。

inputList	
output	
outputList	
dependentTypeList	节点设置的上一周期依赖类型，取值如下： • 0：未勾选任何选项。 • 1：其他节点。 • 2：一级子节点。 • 3：本节点。 详情请参见配置上一周期调度依赖。
dependentDataNode	自定义的上一周期依赖的节点ID列表。当dependentTypeList参数配置为1时生效。
inputContextList	节点上下文配置，详情请参见配置节点上下文。
outputContextList	
tags	保留字段，无业务含义。
tagList	
fileId	
isStop	
dependentType	

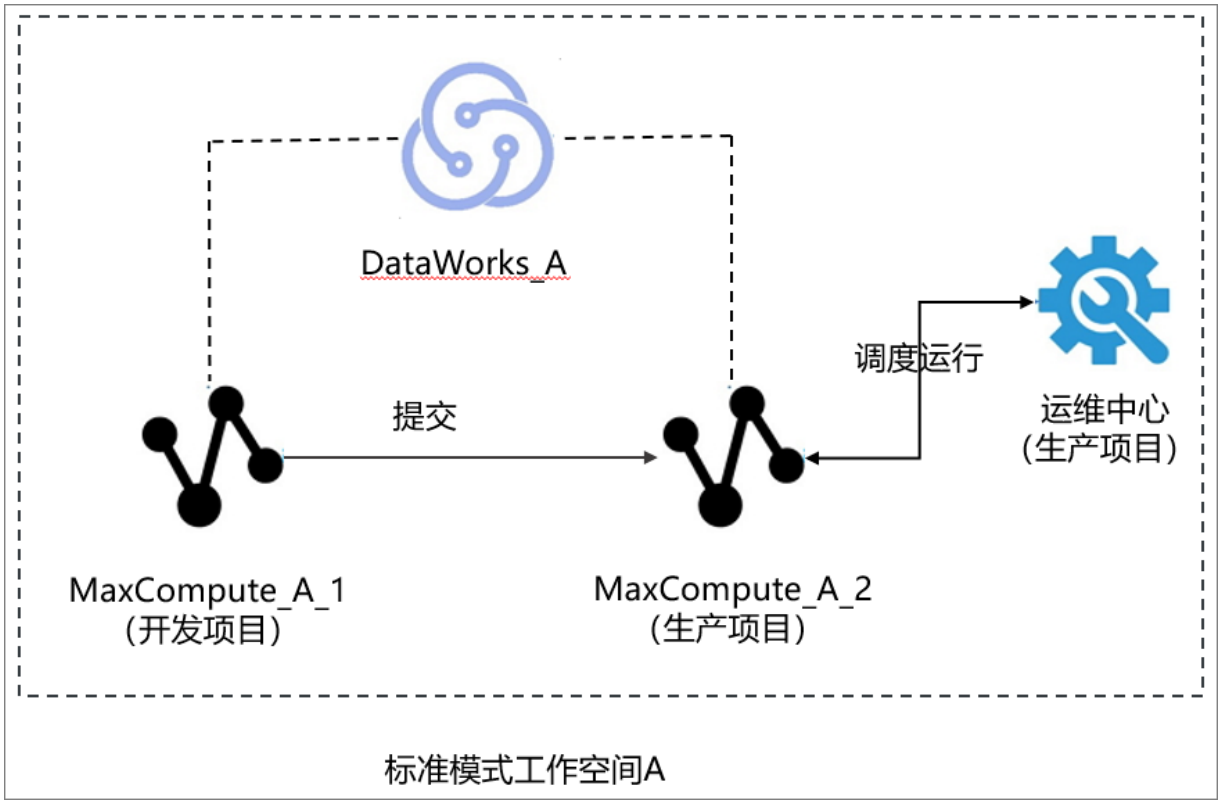
更多时间属性配置，详情请参见时间属性配置说明。

不同实例生成方式对实例生效时间的影响如下：

- T+1次生成实例的节点：在23:30前发布变更操作，周期节点运维在第二天生效。
- 发布后即时生成实例的节点：如果是新增的节点，定时时间在发布时间点十分钟后的实例会正常转出。如果是修改的节点，定时时间在发布时间点十分钟后的实例，会根据最新的调度配置替换修改操作之前的实例。详情请参见实例生成方式：发布后即时生成实例。
- 当天23:30后发布新增或修改的节点，会在第三天生效。
- 当天23:30后发布的即时生成的实例，不会生效。

发布标准模式工作空间的节点

如果您使用的是标准模式的工作空间，系统默认一个DataWorks工作空间对应两个相互绑定的MaxCompute项目（开发环境与生产环境），您可以直接从开发环境提交并发布任务至生产环境。



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**进入数据开发**。
2. 提交节点。

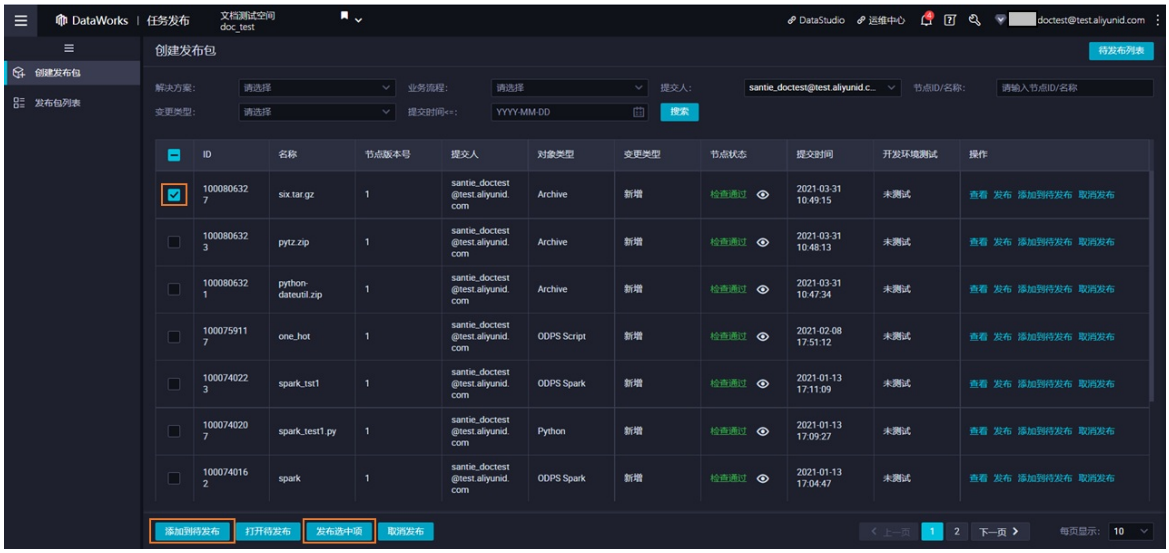
   在标准模式的工作空间，仅开发角色可以提交节点。
   i. 双击打开已配置完成的业务流程，单击工具栏中的图标。
   ii. 在**提交**对话框，选择需要提交的节点名称，输入**备注**，并选中**忽略输入输出不一致的告警**。

   > **说明** 如果您的节点已经被提交，且没有修改节点内容，只是修改了业务流程或节点属性。您可以不选择节点，输入备注后直接提交业务流程。相关改动会正常被提交。
   >
   > 如果节点已经被提交过，在不改变节点内容的情况下，无法再次选择该节点。

   iii. 单击**提交**。
3. 提交成功后，单击右上角的**任务发布**。

   在标准模式的工作空间，仅运维、部署及管理员角色可以发布节点任务。
4. 在**创建发布包**页面，批量选中需要发布的节点，单击**添加到待发布**。

   您可以根据**提交人**、**节点ID**、**节点类型**和**变更类型**等条件过滤和搜索任务。如果单击**发布选中项**，可以立即发布至生产环境调度运行。



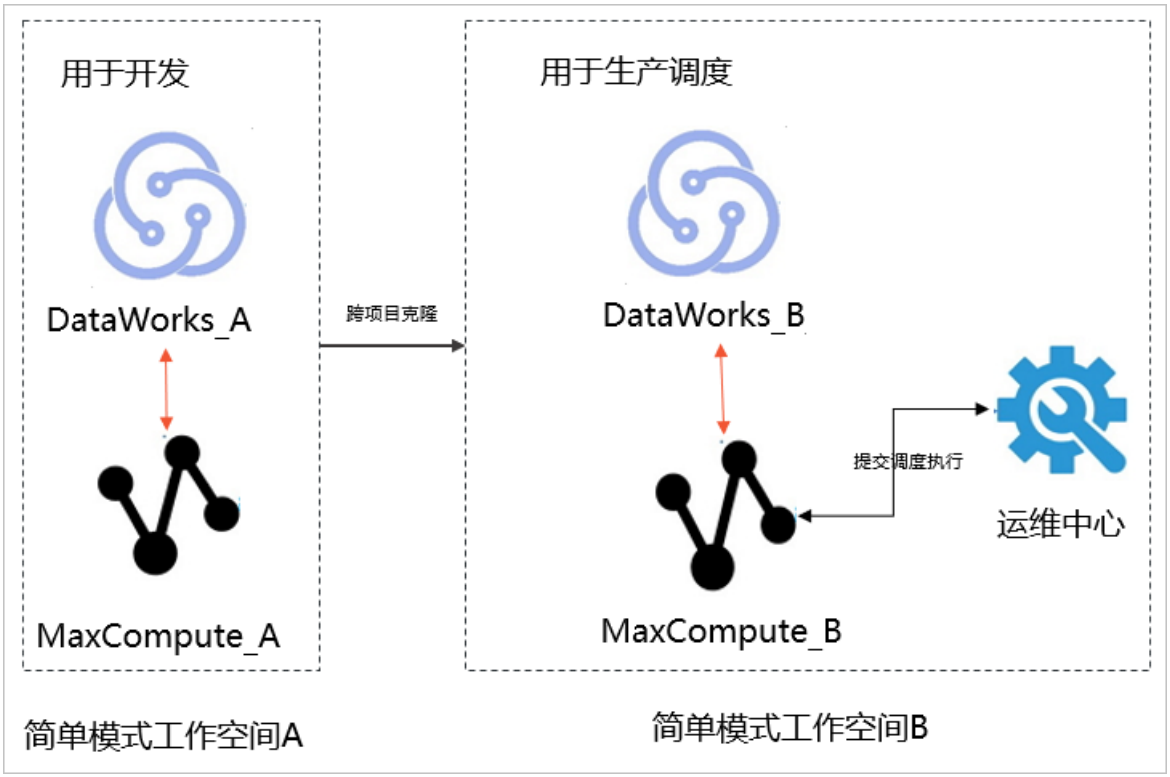
5. 单击**打开待发布**，确认待发布列表中的信息无误后，单击**全部打包发布**，即可发布列表中的节点至生产环境。

> **说明** 简单模式的工作空间严禁直接操作生产环境内的表数据。您可以通过标准模式的工作空间，获得始终稳定、安全、可靠的生产环境。因此，建议您使用标准模式工作空间进行任务的发布与调度。

发布简单模式工作空间的节点

您可以克隆并提交任务至用于生产的工作空间，即通过简单模式工作空间（用于开发）结合简单模式工作空间（用于生产），实现简单模式工作空间内开发环境和生产环境隔离。

例如，您创建两个简单模式的工作空间，分别用于开发和生产。您可以先使用**跨项目克隆**功能，克隆A工作空间中的任务至B工作空间，再在B工作空间内提交克隆的任务至调度引擎进行调度。



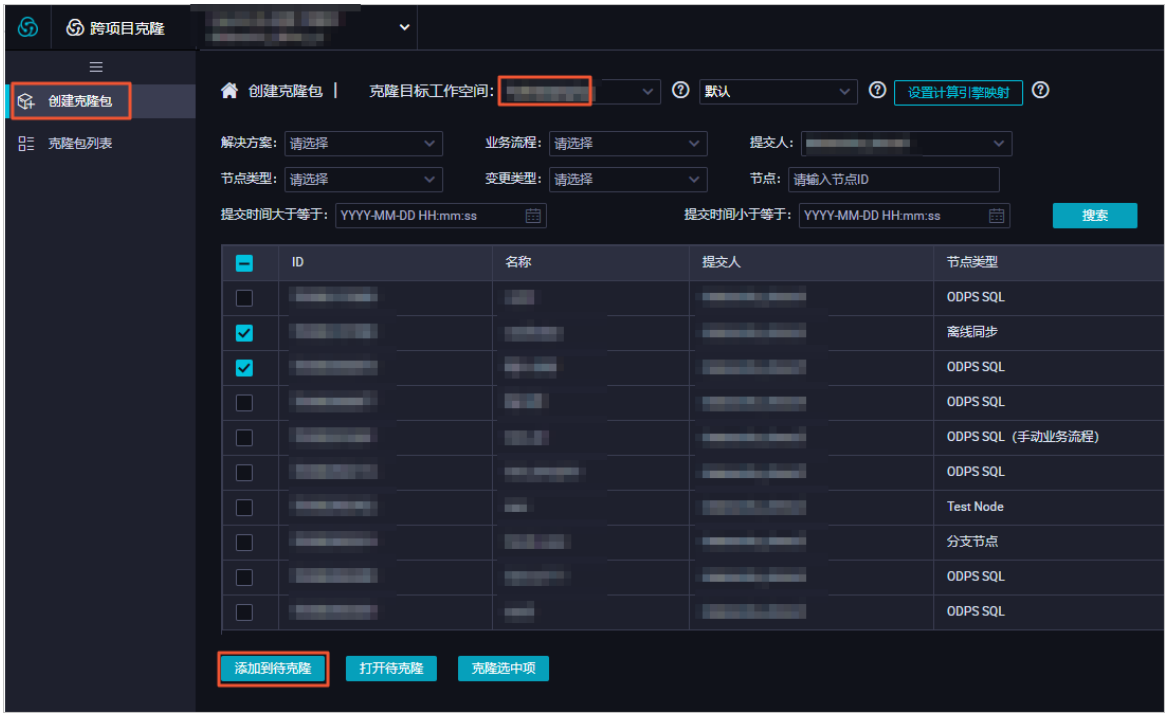
> **说明**
> - 权限要求：除项目管理员外，执行操作的子账号需要具有**运维**角色的权限（创建克隆包、发布克隆任务），才能独立完成该流程。
> - 支持的工作空间类型：仅简单模式工作空间支持克隆任务至其它工作空间。
> - 准备工作：创建简单模式的源工作空间A和标准模式的目标工作空间B。

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**进入数据开发**。
2. 提交节点。
   i. 双击打开已配置完成的业务流程，单击工具栏中的图标。
   ii. 在**提交**对话框，选择需要提交的节点名称，输入**备注**，并选中**忽略输入输出不一致的告警**。
   iii. 单击**提交**。
3. 单击页面右上角的**跨项目克隆**。
4. 在**创建克隆包**页面，选择需要克隆的节点和**克隆目标工作空间**。
5. 单击**设置计算引擎映射**，设置当前工作空间与目标工作空间计算引擎的映射关系。

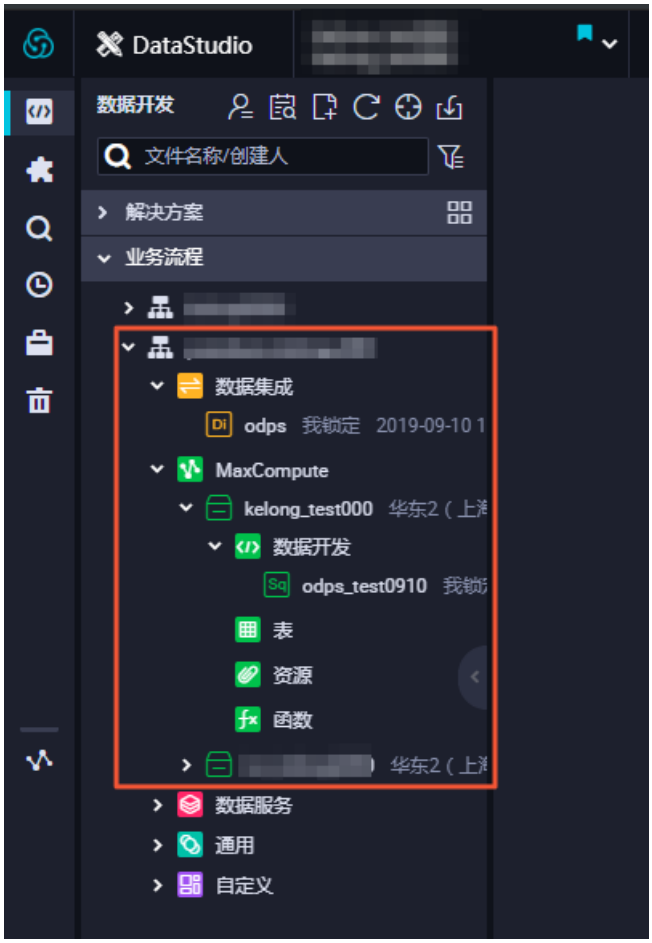
   目标工作空间存在多个计算引擎，因此需要设置当前工作空间与目标工作空间计算引擎的映射关系，才可以进行克隆。如果不设置，则克隆至目标工作空间的默认计算引擎中。

> 说明
>
> - 如果目标工作空间不存在克隆的节点所属的引擎类型，**计算引擎映射信息**对话框会进行提示。您可以通过选择**跳过目标引擎实例为空的节点**，来过滤无法克隆的节点，否则在克隆过程中会报错。
> - 在源工作空间和目标工作空间某种引擎类型存在两个以上引擎实例的情况下，会显示**设置计算引擎映射**按钮。

6. 单击**添加到待克隆**，添加需要克隆的节点至**待克隆列表**。



7. 打开右上角的**待克隆列表**，单击**全部克隆**。
8. 确认引擎映射预检后的**计算引擎映射信息**，单击**确定**。
9. 待页面提示克隆成功后，即可进入目标工作空间查看克隆结果，通常会克隆业务流程的整体目录结构。

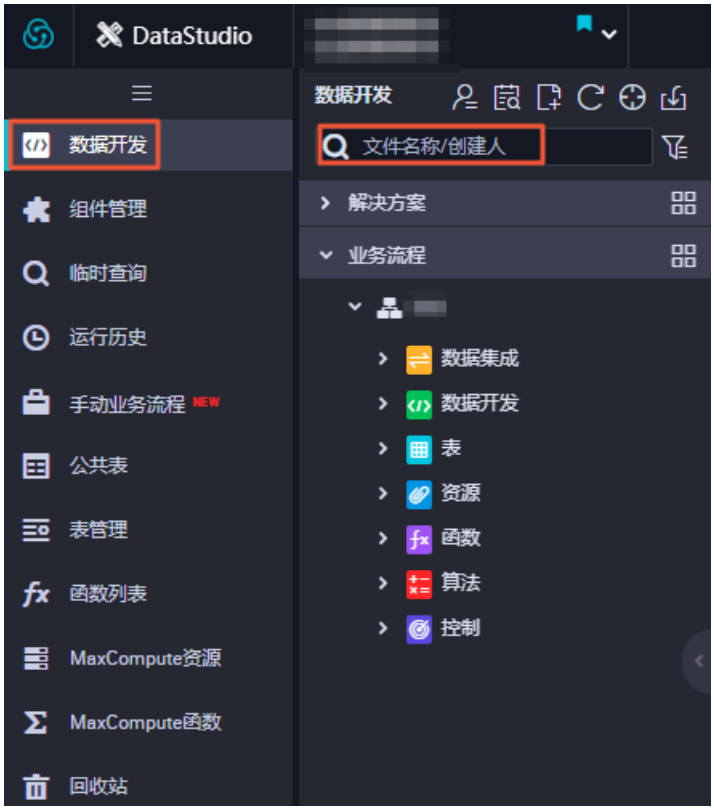


10.2.2.2. 下线任务

任务下线是指在某些情况下，需要将任务永久删除，包括开发环境的任务下线和生产环境的任务下线两种场景。

开发环境的任务下线

1. 登录DataWorks控制台，进入**数据开发**页面。
2. 通过任务节点类型、关键字来搜索需要删除的任务。



3. 右键单击要删除的任务，选择**删除**，则开发环境任务下线完成。

生产环境的任务下线

当已发布到生产环境的任务需要删除时，需按照删除任务>发布下线任务>执行发布的流程进行下线。

说明 子节点依赖关系的处理：由于生产环境的任务下线涉及到子节点的依赖，因此在您删除任务前，请先一层层往下处理好子节点的依赖关系后，再进行删除。否则会提示存在子节点，下线失败。

处理步骤：

1. 查找此节点的下游节点，可在工作流管理查看生产调度依赖关系。
2. 在数据开发页面重新编辑此子节点的父节点，或者直接删除此子节点。

如果提示子节点下还有子节点，请参见上述步骤逐层向下处理。

1. 删除任务。

   可以参见前文开发环境的任务下线操作，删除需要下线的任务。

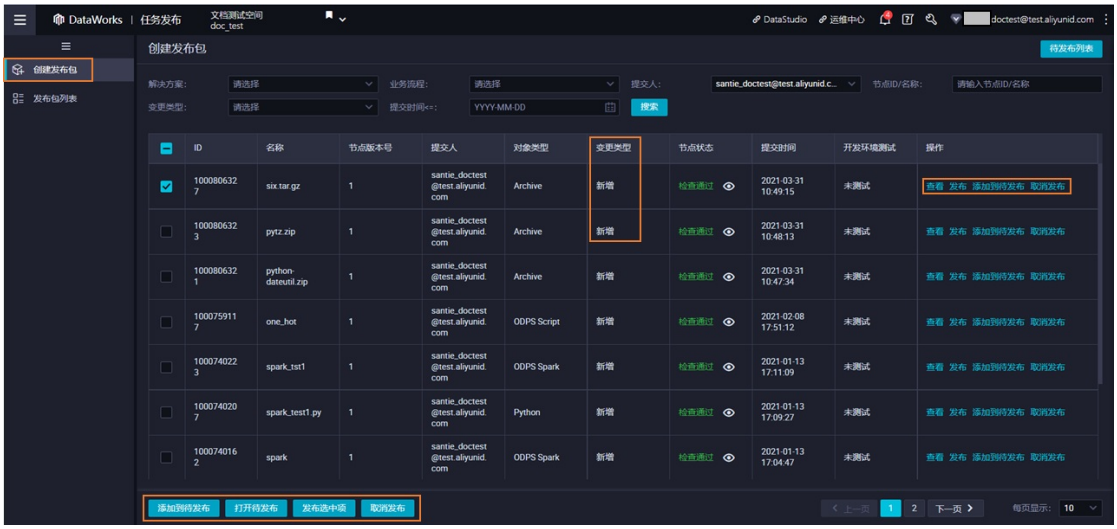
   说明 在运维中心下线任务，无需执行任何审批流程，目标任务便会在生产环境直接删除，不推荐使用该方式下线任务。建议您先在开发环境删除任务后，将任务提交发布至生产环境下线，执行下线操作审核流程。

2. 发布下线任务。

   说明 仅管理员及运维角色具有发布权限。如果是其他角色，建议通知运维人员进行发布。

   i. 删除需要下线的任务后，单击右上角的**任务发布**。

ii. 在**创建发布包**页面，勾选需要下线的任务。



iii. 单击**发布选中项**。

您也可以单击**添加到待发布**，进入**待发布列表**进行发布。

3. 执行发布。

单击**确认执行**对话框中的**发布**，完成下线任务的发布。

10.2.3. 跨项目克隆

10.2.3.1. 跨项目克隆说明

跨项目克隆主要用于隔离同租户（阿里云账号）简单模式工作空间下的开发环境和生产环境，您也可以利用跨项目克隆功能实现计算、同步等类型的任务在工作空间之间的克隆迁移。本文为您介绍如何处理跨项目克隆时任务间的依赖关系。

通过**跨项目克隆**功能进行克隆任务后，系统为区分同租户（阿里云账号）下不同工作空间之间任务的输出名称，会自动对每个任务输出名称作出一系列命名更改，目的是为了平滑复制依赖关系或保持原有依赖关系不变。

> 说明
> - 跨项目克隆不支持跨地域发布。
> - 目前不支持克隆旧版工作流至新的工作空间，请迁移源端旧版工作流中的任务至业务流程下的某个目录后，再克隆该业务流程至目标端工作空间。

克隆责任人分为**默认**和**克隆包创建者**：

- 当克隆责任人为默认的项目管理员时，克隆至目标工作空间后，您可以选择克隆后任务责任人为**默认**或**克隆包创建者**。

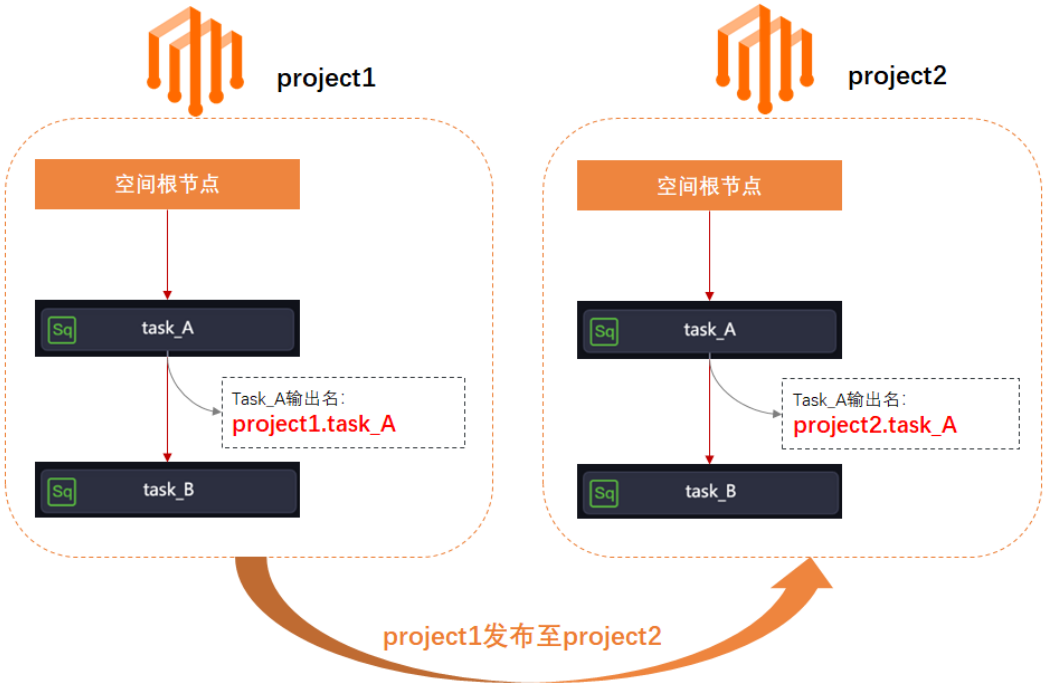
  克隆成功后，责任人将第一优先级被置为原责任人。如果原责任人不在目标工作空间，则置为克隆包创建者。

- 当克隆责任人为克隆包创建者时，克隆至目标工作空间后，您可以选择克隆后任务责任人为**默认**或**克隆包创建者**。

  克隆成功后，责任人将第一优先级被置为原责任人。如果原责任人不在目标工作空间，会询问是否变更责任人。如果确认变更，则任务克隆成功且责任人变更为克隆包创建者。如果不变更责任人，则克隆任务取消。

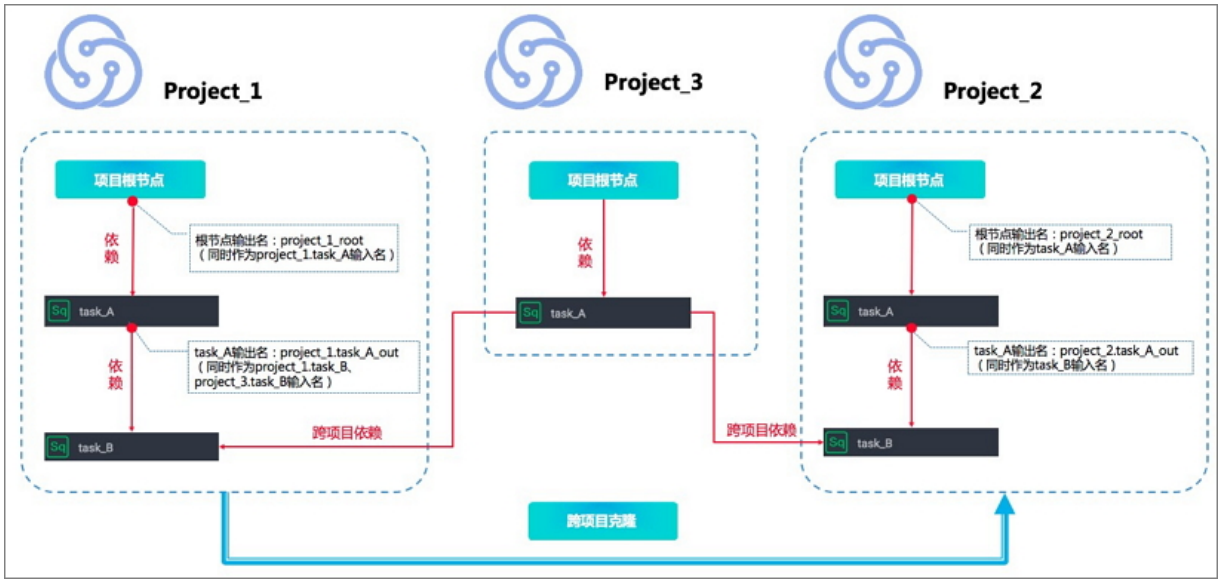
完整的业务流程克隆

用户使用task_A任务的输出点在project_1中为project_1.task_A_out，克隆至project_2之后输出点名为project_2.task_A_out。



跨项目依赖任务克隆

project_1中的任务task_B依赖了project_3中的任务task_A，在将project_1. task_B克隆为project_2.task_B之后，依赖关系将一同克隆，即project_2.task_B仍然依赖project_3.task_A。



10.2.3.2. 跨项目克隆实践

本文将为您介绍跨项目克隆的操作实践。

支持的场景

跨项目克隆支持以下两种场景：

- 从一个简单模式的工作空间克隆至另一个简单模式的工作空间。
- 从一个简单模式的工作空间克隆至另一个标准模式的工作空间。

> **说明**
>
> - 跨项目克隆以节点为单位，文件夹和业务流程会跟随节点一起克隆至目标工作空间。
> - 跨项目克隆不能在上游节点没有克隆至目标工作空间的情况下，进行克隆下游节点的操作。

操作步骤

1. 进入DataStudio（数据开发）页面，打开相应的业务流程。
2. 单击右上角的**跨项目克隆**，跳转至相应的克隆页面，过滤出相应的节点，并将任务克隆到目标工作空间。

3. 筛选需要克隆的节点，并选择**克隆目标工作空间**。

4. 单击**设置计算引擎映射**，设置当前工作空间与目标工作空间计算引擎的映射关系。

目标工作空间存在多个计算引擎，因此需要设置当前工作空间与目标工作空间计算引擎的映射关系，方可进行克隆操作。如果不设置，则克隆至目标工作空间的默认计算引擎中。

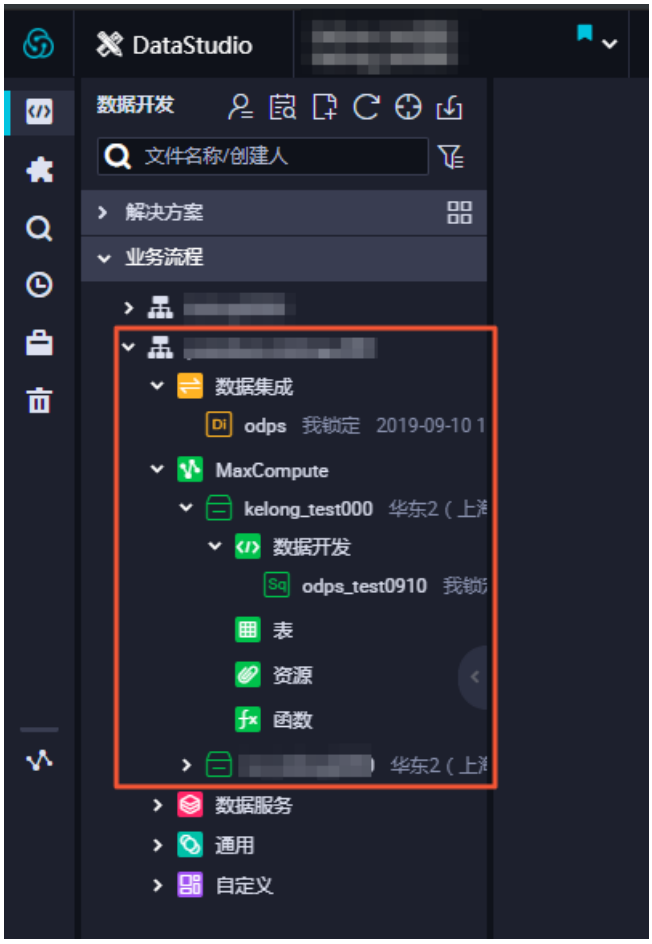
> **说明**
>
> - 当克隆的节点中存在部分节点所属的引擎类型在目标工作空间中不存在的情况，**计算引擎映射信息**对话框会进行提示，您可以通过勾选计算引擎确认对话框会给出提醒，可以通过勾选**跳过目标引擎实例为空的节点**，来过滤无法克隆的节点，否则在克隆过程中会报错。
> - 在源工作空间和目标工作空间某种引擎类型存在两个以上引擎实例的情况下，会显示**设置计算引擎映射**按钮。

5. 单击**添加到克隆**，添加需要克隆的节点至**待克隆列表**。

6. 打开右上角的**待克隆列表**，单击**全部克隆**。

7. 确认引擎映射预检后的**计算引擎映射信息**，单击**确定**。

8. 待页面提示克隆成功后，即可进入目标工作空间查看克隆结果，通常会克隆业务流程的整体目录结构。



11.高级功能与开发提效

11.1. 代码搜索

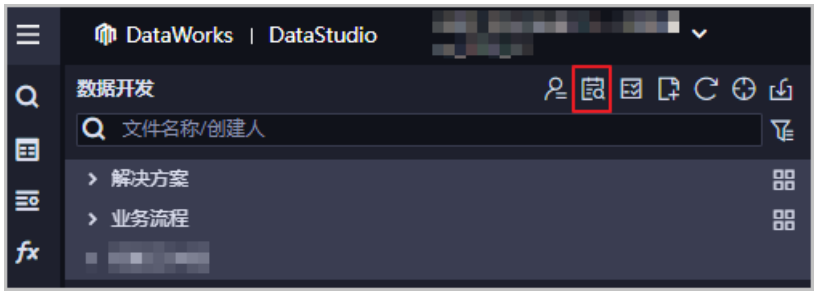
DataWorks的代码搜索功能，用于通过关键字搜索节点中的代码片段，并展示包含该代码片段的所有节点及片段的详细内容。当目标表数据产生变更，您需要查找操作源（即导致目标表数据变更的任务）时，可以使用该功能。本文以数据开发功能为例，为您介绍代码搜索的操作详情。

使用限制

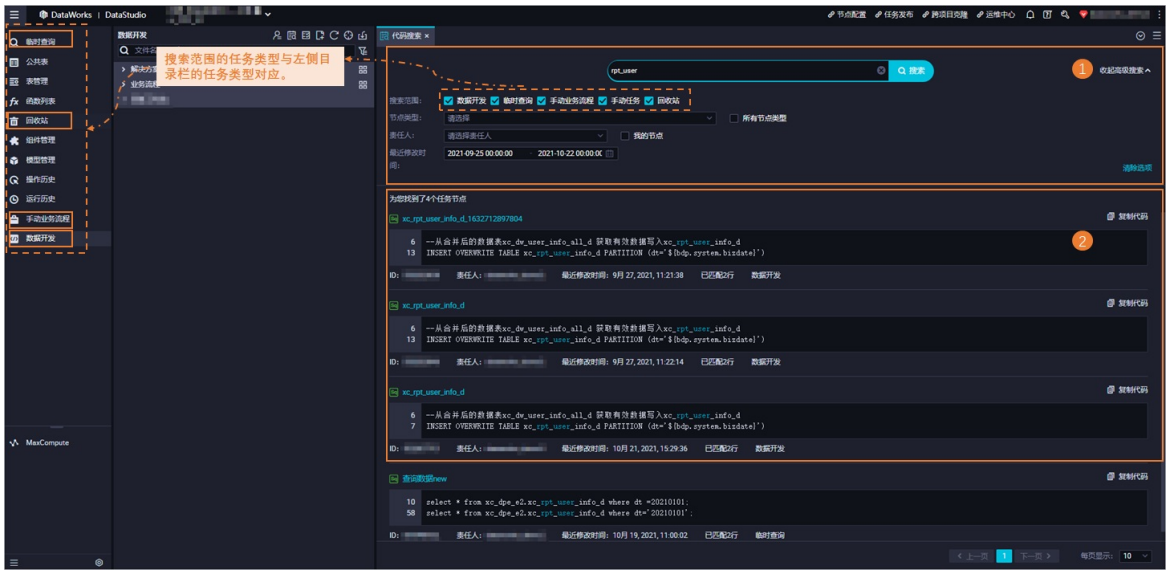
- 仅DataWorks基础版以上（不包含基础版）的版本才能使用代码搜索功能。
- 代码搜索功能仅支持搜索**数据开发**、**临时查询**、**手动业务流程**、**手动任务**、**回收站**目录下的节点。各目录功能，详情请参见数据开发功能索引
- 仅DataWorks V1.0版本会显示**手动任务**目录。

操作步骤

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. 在数据开发目录树区域，单击顶部菜单栏的图标，进入代码搜索页面。



3. 配置搜索条件并查看搜索结果。



i. 输入搜索关键词并配置搜索条件。

a. 在区域1的搜索框，输入需要搜索的关键词，例如，需要查询的代码片段、表名称等。输入的关键词越详细，查询的结果越精确。

示例：配置关键词为 `alter table test` 操作， `rpt_user` 表。

b. 单击搜索框右侧的**展开高级搜索**，通过**搜索范围**、**节点类型**、**责任人**、**最近修改时间**等条件进行筛选，确定关键词的搜索范围。

> **说明**
>
> - 高级搜索条件能够帮助您更精确、快速的定位目标节点，如果您不配置高级搜索条件，则默认在当前工作空间的全局范围内搜索关键词。
> - 代码搜索功能仅支持搜索**数据开发**、**临时查询**、**手动业务流程**、**手动任务**、**回收站**目录下的节点。各目录功能，详情请参见数据开发功能索引
> - 仅DataWorks V1.0版本会显示**手动任务**目录。

ii. 单击**搜索**按钮。

iii. 查看搜索结果。

在区域2，您可以查看通过指定条件搜索出的、包含关键词的所有节点及代码详情，快速定位目标节点，获取变更操作。同时，您还可以执行如下操作：

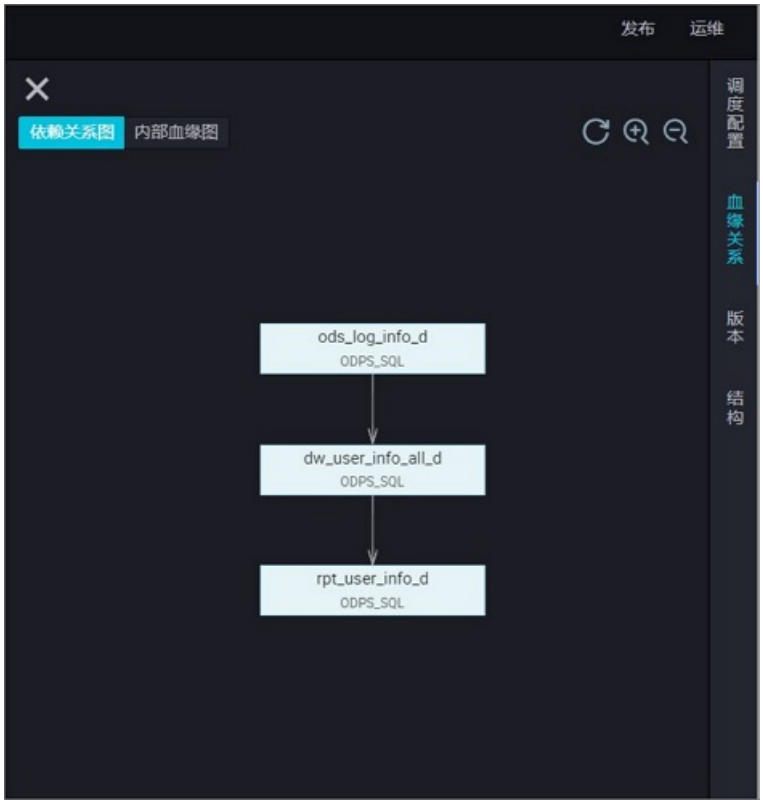
- 单击节点名称，即可跳转至该节点页面查看节点详情。
- 单击**复制代码**，即可复制当前代码片段进行后续开发使用。

11.2. 血缘关系

血缘关系为您展示当前节点和其它节点的关系，展示依赖关系图和内部血缘图两部分。

依赖关系图

依赖关系图根据节点的依赖关系，展示当前节点的依赖是否为自己预期的情况。如果不是，可以返回调度配置界面重新设置。

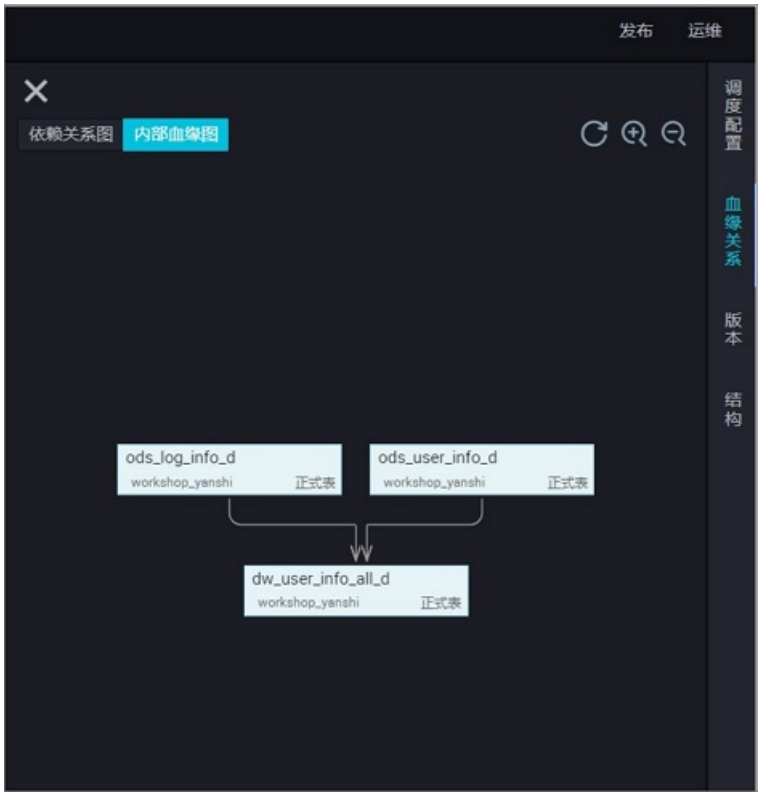


内部血缘图

内部血缘图根据节点的代码进行解析，如下所示。

```
INSERT OVERWRITE TABLE dw_user_info_all_d PARTITION (dt='${bdp.system.bizdate}')
SELECT COALESCE(a.uid, b.uid) AS uid
  , b.gender
  , b.age_range
  , b.zodiac
  , a.region
  , a.device
  , a.identity
  , a.method
  , a.url
  , a.referer
  , a.time
FROM (
  SELECT *
  FROM ods_log_info_d
  WHERE dt = ${bdp.system.bizdate}
) a
LEFT OUTER JOIN (
  SELECT *
  FROM ods_user_info_d
  WHERE dt = ${bdp.system.bizdate}
) b
ON a.uid = b.uid;
```

根据上述SQL语句，解析出如下内部血缘图。将dw_user_info_all_d作为JOIN拼接ods_log_info_d的输出表解析，展示表之间的血缘关系。



11.3. 版本

版本是指当前节点的提交、发布记录。您可以在版本面板查看节点历史版本、提交人、提交时间、变更类型、状态、备注等信息。

> **说明** 只有提交过的节点才会存在版本信息，每次提交都会生成一个新版本，并在**版本**面板产生一条新纪录。

信息	说明
文件ID	当前节点的节点ID。

信息	说明
版本	每次发布都会生成一个新的版本，第一次新增为V1，第二次修改为V2，以此类推。
提交人	提交发布节点的操作人。
提交时间	版本的提交时间，默认记录最后一次提交的时间。
变更类型	当前节点的操作历史。首次的提交记录为新增，之后对节点进行修改的提交记录为修改。
状态	各版本中，该节点的操作状态记录。包括以下三种状态： • **已提交**：已提交至开发环境，但未提交至发布包、未发布至生产环境，节点处于已提交但未发布的状态。 • **中间版本**：已提交至开发环境且已提交至任务发布页面，节点处于已提交待发布的状态。 • **已发布**：已提交至开发环境且已发布至生产环境，节点处于已提交并发布状态。
备注	在提交时，对该节点进行变更描述。以便其他人操作该节点时，找到相应的版本。
操作	包括**代码**和**回滚**操作： • **代码**：查看此版本的代码，精确查找需要回滚的记录版本。 • **回滚**：将当前节点回滚到之前某个需要的版本，回滚后需要重新提交发布。
比较	将两个版本的代码和参数进行比较。  单击**查看详情**，可以进入详情页，对比代码、调度属性的变更情况。 **说明** 比较只可以在两个版本中进行，无法选中一个或三个以上（包括三个）节点进行比较。

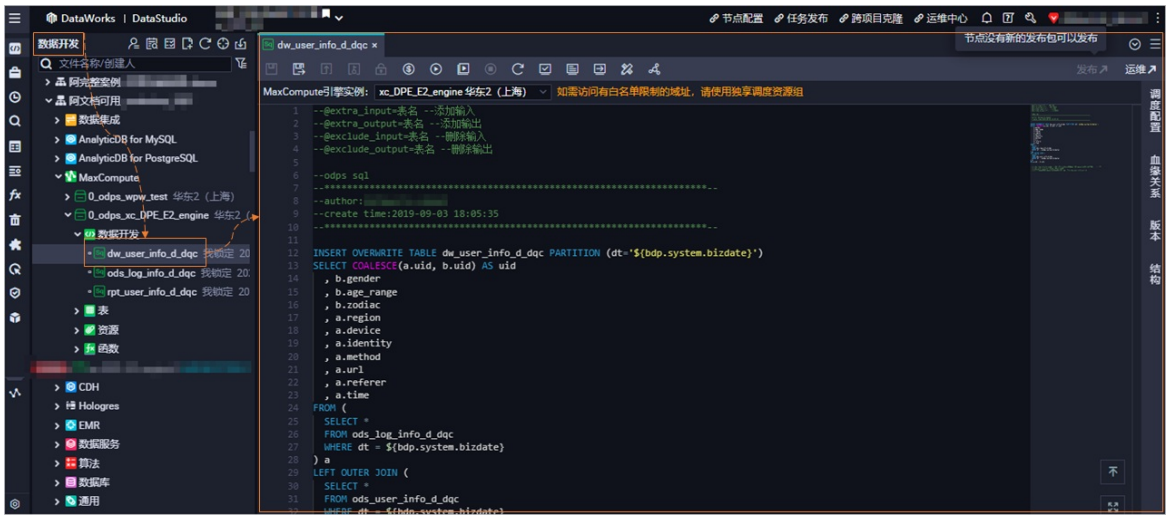
11.4. 查看代码结构

DataWorks的代码结构功能，帮助您根据当前节点编写的SQL代码，解析出SQL代码运行的流程结构图。您可以通过代码结构快速定位、查看及修改SQL代码。

查看代码结构

1. 进入DataStudio。
   i. 登录DataWorks控制台，选择目标区域后，在左侧导航栏单击**工作空间列表**。
   ii. 单击目标工作空间后的**进入数据开发**，默认进入该工作空间的DataStudio功能模块。
2. 进入目标节点的编辑页面。

   您可以在**数据开发**或**手动业务流程**的目录树下，找到目标节点，双击该节点即可进入节点的编辑页面。



3. 查看代码结构。

   您可以在节点编辑页面的右侧导航栏，单击**结构**，查看该节点中SQL代码的经典结构及大纲结构。

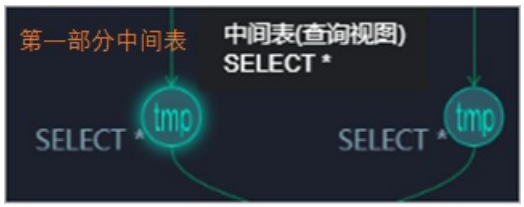
   - 经典结构：SQL代码的执行过程通常是通过SQL算子层层递进的，最终得到想要的结果。经典结构主要用于查看SQL代码中涉及的SQL算子，及各算子之间的关联关系。

     鼠标悬停至对应算子的图标上，即可查看该算子的含义。常用的SQL算子如下。

- **源表**：查询的目标表。

- **筛选**：筛选目标表中需要查询的具体分区。

- **中间表（查询视图）**：存放查询结果的临时表。



本文示例图中包含两部分中间表，第一部分的中间表用于将查询数据的结果放入一张临时表；第二部分的中间表，用于将JOIN的结果汇总到一张临时表，该临时表可以保存3天，3天后自动清除。

- **关联（join）**：用于将所有查询结果通过JOIN语句拼接。

- 大纲结构：主要用于查看SQL代码中核心语句的层级结构。

单击经典结构的SQL算子或大纲结构的核心代码，即可快速定位至该代码的位置。您可以根据实际需求编辑修改目标代码。

11.5. 资源组编排

DataWorks的资源组编排功能，帮助您在数据开发阶段，批量修改指定业务流程下目标节点使用的调度资源组。当您的工作空间中有多个调度资源组时，可以根据实际业务需求，使用该功能快速为目标节点重新分配资源组，促进资源的合理使用。本文为您介绍如何进行资源组编排。

前提条件

- 资源组编排是基于业务流程使用的功能，因此您需要先创建业务流程。创建业务流程，详情请参见创建业务流程。
- 资源组编排是用于对当前工作空间使用的调度资源组进行快速合理分配，因此您需要先开通调度资源组。

开通DataWorks后，DataWorks自动为用户提供公共调度资源组，供用户共享使用。如果公共调度资源组无法满足业务需求，则可开通独享调度资源组。开通独享调度资源组，详情请参见独享调度资源组概述。

背景信息

资源组编排功能，用于在数据开发阶段批量修改指定业务流程下目标节点使用的调度资源组。不同规格的资源组支持的最大并发任务数不同，您可以根据实际业务需求，将目标节点分配至不同的资源组上运行。资源组规格及其支持的最大并发任务数，详情请参见独享调度资源组计费说明：包年包月。

> **说明** 生产阶段批量修改调度资源组，详情请参见周期任务的批量操作功能。

使用限制

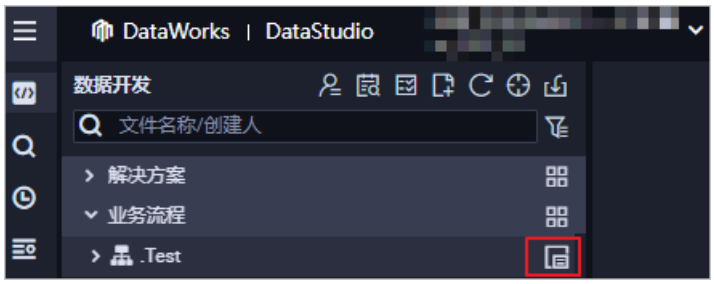
- DataWorks当前仅支持对周期任务使用的调度资源组进行编排。
- 虚拟节点运行时不占用调度资源，因此虚拟节点不需要并且不支持修改调度资源组。

操作步骤

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. 进入资源组编排页面。

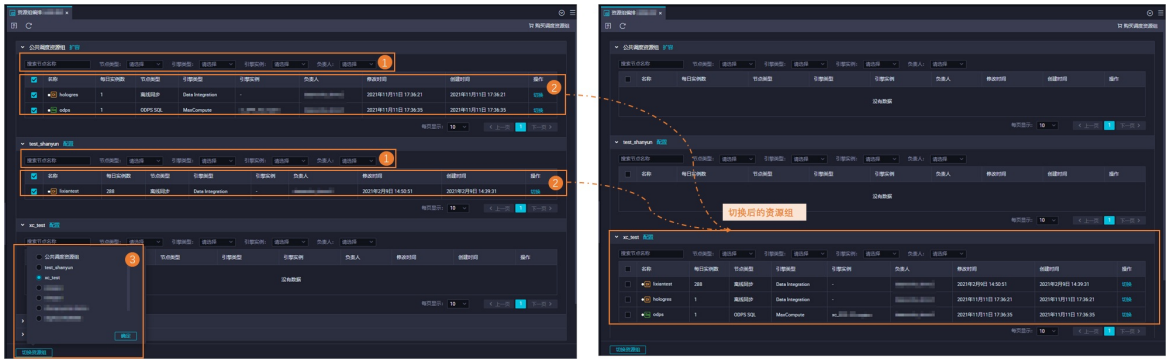
   在数据开发页面，鼠标悬停至目标业务流程，单击业务流程后的图标，进入该业务流程的资源组编排界面。

   > **说明** 资源组编排是基于业务流程使用的功能，因此您需要先定位至目标业务流程。

3. 编排资源组。

   > **说明**
   > - 当不同资源组下的目标节点需要切换至相同的资源组时，DataWorks支持对其进行同批次的统一编排操作。
   > - 当目标节点需要切换至不同的资源组时，您需要分批次对其执行编排操作。



i. （可选）您可以通过**节点名称**、**节点类型**、**引擎类型**等条件进行筛选，查找指定条件的节点。

ii. 确认节点信息并勾选需要批量切换资源组的目标节点。

iii. 单击**切换资源组**，选择需要使用的目标资源组。

iv. 单击**确定**，完成资源组切换。

资源组的其他操作如下：

- **扩容**：仅公共调度资源组支持扩容功能。单击**公共调度资源组**后的**扩容**，即可跳转至公共资源组页面，购买公共调度资源包。
- **配置**：仅独享调度资源组支持配置功能。单击**独享调度资源组**后的**配置**，即可跳转至独享资源组页面，查看该资源组的详情，进行相关配置操作。
- **购买调度资源组**：如果当前工作空间没有可用的资源组，您可以单击**购买调度资源组**购买。
- **切换单个节点的资源组**：单击目标节点**操作**列的**切换**，即可修改该节点使用的调度资源组。

4. 单击图标，提交资源组的编排操作。

5. 发布资源组的编排操作。

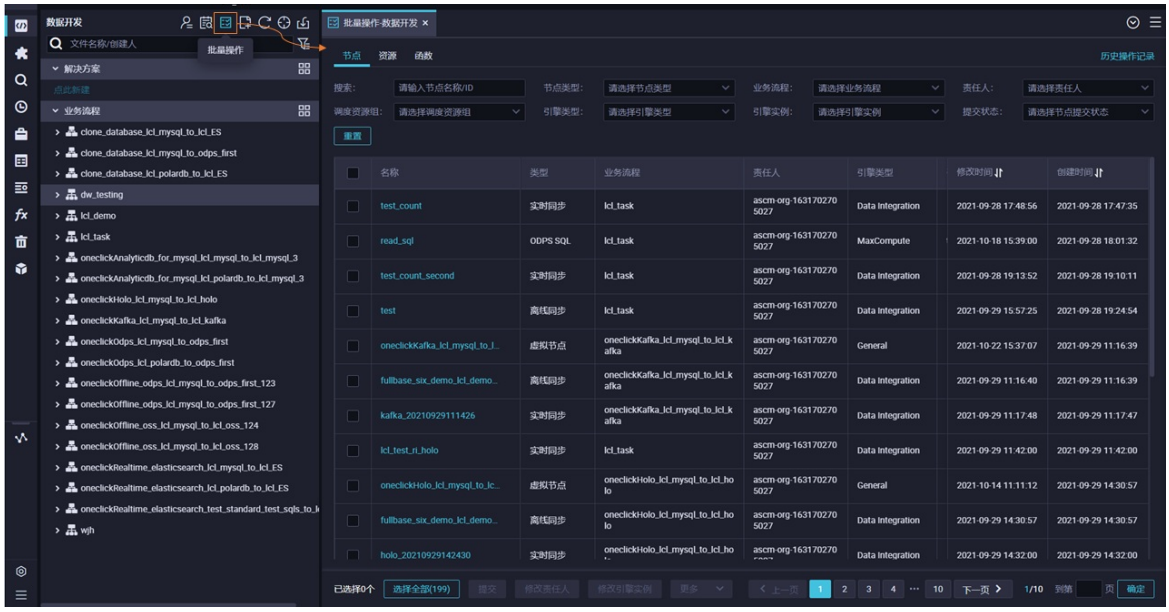
   节点切换资源组后，您需要进入任务发布界面将节点的更新操作进行发布。发布后，节点在生产环境调度时才可以使用修改后的调度资源组。发布节点，详情请参见发布任务。

11.6. 批量操作

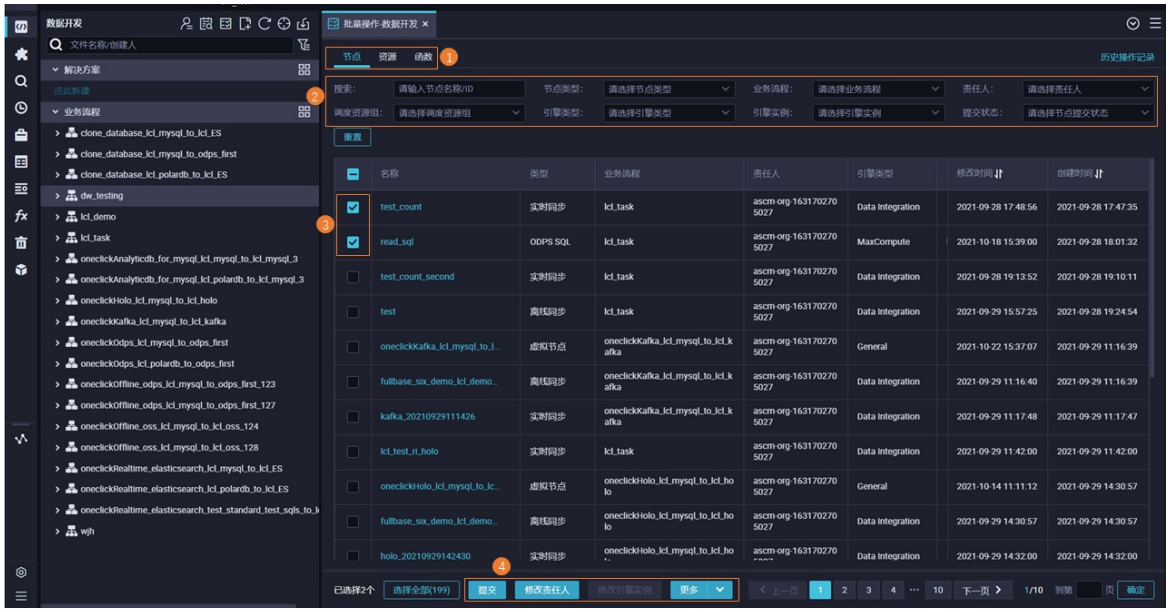
DataWorks支持对节点、资源、函数进行批量修改责任人等批量编辑操作，并支持批量提交并发布，将变更操作发布至生产环境生效。本文为您介绍批量操作的界面入口及详细指导。

操作步骤

1. 登录DataWorks控制台，进入**数据开发**页面后，在顶部的快捷操作按钮中单击**批量操作**按钮，即可打开批量操作页面。



2. 批量修改。



i. 在**批量操作**页面中，您可以在顶部页签中选择对**节点**、**资源**或者**函数**进行批量处理。

ii. 您可以在页签中的上部通过**业务流程**、**责任人**等条件进行过滤。

> **说明**
>
> - 离线同步任务支持数据集成资源组、数据来源与去向类型、数据来源与去向数据源进行过滤。
> - 对**节点**、**资源**或者**函数**的过滤条件不完全一致，已实际界面为准。

iii. 在过滤后的列表中，您可以勾选待批量处理的**节点**、**资源**或者**函数**。

iv. 选择完成后，您可以在底部选择对已选的多个对象进行批量修改的操作。

> 说明
> - 对节点进行批量操作时，界面出现强制修改配置选项时，选择是将一并修改被其他人锁定的节点，选择否将只修改自己锁定的节点。
> - 此界面的批量修改、提交仅对开发环境生效，如果您需要修改生产环境节点、资源、函数，批量修改并提交后，您需要进入任务发布界面，将刚刚的变更操作发布至生产环境。

3. 您可单击右上角的**历史操作记录**，跳转至操作历史界面查看历史操作详情。

11.7. 上传数据

DataWorks支持将本地的CSV文件或部分文本文件数据直接上传至MaxCompute表中，本文为您介绍操作步骤详情。

前提条件

已准备好用于接收本地数据的MaxCompute表。

您可以选择已创建的MaxCompute表，或者直接新创建一个MaxCompute表，建表操作可参见创建MaxCompute表。

使用限制

当前仅支持上传本地数据至MaxCompute表。

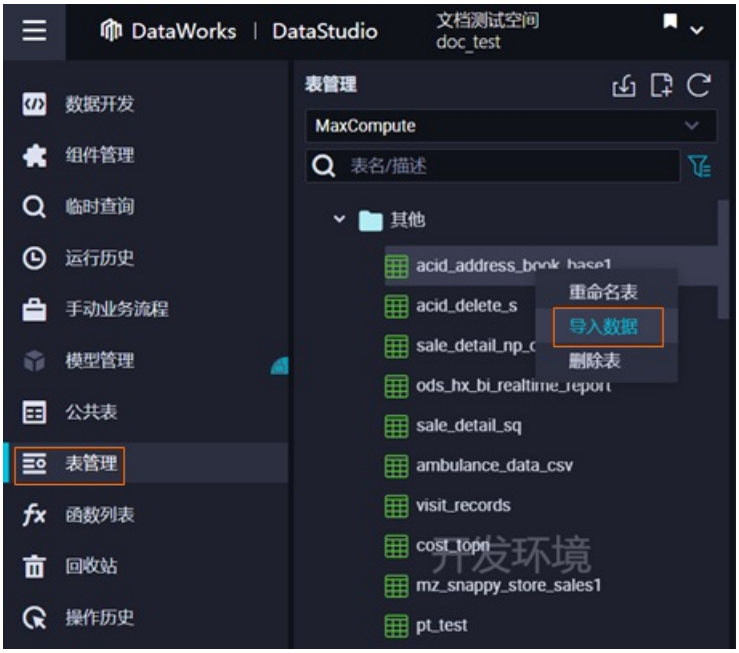
上传数据操作入口

您可以在数据开发的头部菜单栏、业务流程下的表分组或表管理页面中进行上传数据的操作，入口如下所示。

- 在**数据开发**页面的头部菜单栏中。

- 在**数据开发**页面业务流程下的表分组中。

- 标准模式下，在**表管理**页面中，对开发表右键上传数据。



上传数据

当前仅支持上传本地数据至MaxCompute表。

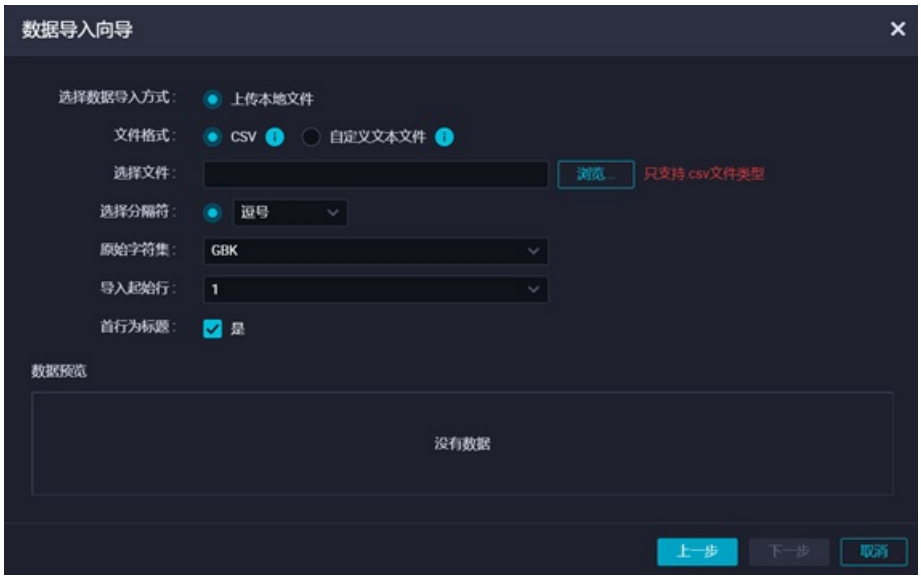
1. 参考上述步骤，进入上传数据入口后，单击**导入数据**。
2. 在弹出的**数据导入向导**页面中，确认数据导入的表，选择分区并检测分区是否存在，完成后单击**下一步**。

> **说明** 非分区表无需设置分区，直接单击**下一步**即可。

您可以设置数据导入到该表的哪个分区，完成后可单击**检测**按钮，测试分区是否存在。

- 当填写的分区不存在时，继续操作将会新建该分区，并插入数据到此新建的分区中。
- 当填写的分区存在时，将上传的数据追加在该分区中。

3. 配置导入数据的文件格式，并上传导入文件，设置导入数据的分隔符等导入设置，完成后单击**下一步**。



主要配置参数如下。

参数	配置说明
文件格式	您可以根据待上传的数据文件类型选择文件格式，当前支持**CSV**和**自定义文本文件**两种文件格式，其中自定义文本文件支持 `.txt` 、 `.csv` 和 `.log` 类型的文件。
选择文件	单击**浏览**，根据界面提示选择待上传的数据文件。

参数	配置说明
选择分隔符	选择用于切分字段的分隔符。支持逗号、Tab、分号、空格、|、#、&。 **说明** 当导入自定义本义文件时，也支持自定义分隔符，即设置自定义字符或字符串作为字段分隔符。
原始字符集	根据实际情况选择上传数据文件的原始编码格式。目前支持GBK、UTF-8、CP936、ISO-8859。
导入起始行	设置从待导入数据文件的多少行开始导入。
首行为标题	选择是否设置待导入的数据文件的第一行为标题行。 ○ 勾选是，首行数据将不上传。 ○ 未勾选，则首行数据上传。

4. 选择目标表字段与源字段的匹配方式，确认后单击**导入数据**。

   您可以选择按位置匹配或按名称匹配两种方式来匹配待上传的数据与MaxCompute表字段的对应关系。

   完成后，界面提示数据导入成功，您即完成了从本地上传数据至MaxCompute表中，您可以在临时查询页面中查看已上传的数据，操作请参见创建临时查询。

11.8. 编辑器快捷键列表

本文为您介绍代码编辑器的常用快捷键。

Windows的Chrome版本下

`Ctrl + S`：保存。

`Ctrl + Z`：撤销。

`Ctrl + Y`：重做。

`Ctrl + D`：同词选择。

`Ctrl + X`：剪切一行。

`Ctrl + Shift + K`：删除一行。

`Ctrl + C`：复制当前行。

`Ctrl + I`：选择行。

`Shift + Alt + 鼠标拖动`：列模式编辑，修改一整块内容。

`Alt + 鼠标点选`：多列模式编辑，多行缩进。

`Ctrl + Shift + L`：为所有相同的字符串实例添加光标，批量修改。

`Ctrl + F`：查找。

`Ctrl + H`：替换。

`Ctrl + G`：定位到指定行。

`Alt + Enter`：选中所有查找匹配上的关键字。

`Alt↓` 或 `Alt↑`：向下或向上移动当前行。

`Shift + Alt + ↓` 或 `Shift + Alt + ↑`：向下或向上复制当前行。

`Shift + Ctrl + K`：删除当前行。

`Shift + Ctrl + \`：光标跳至匹配的括号。

`Ctrl + ]` 或 `Ctrl + [`：增加或减小缩进。

`Home` 或 `End`：移至当前行最前或最后。

`Ctrl + Home` 或 `Ctrl + End`：移至当前文件的最前或最后。

`Ctrl + →` 或 `Ctrl + ←`：向右或向左按单词移动光标。

`Shift + Ctrl + [` 或 `Shift + Ctrl + ]`：折叠或展开光标所在区域。

`Ctrl + K + Ctrl + [` 或 `Ctrl + K + Ctrl + ]`：折叠或展开光标所在区域的子区域。

`Ctrl + K + Ctrl + 0` 或 `Ctrl + K + Ctrl + j`：折叠或展开所有区域。

`Ctrl + /`：注释或解除注释光标所在行或代码块。

Mac的Chrome版本下

`Cmd + S`：保存。

`Cmd + Z`：撤销。

`Cmd + Y`：重做。

`Cmd + D`：同词选择。

`Cmd + X`：剪切一行。

`Cmd + Shift + K`：删除一行。

`Cmd + C`：复制当前行。

`Cmd + I`：选择当前行。

`Cmd + F`：查找。

`Cmd + Alt + F`：替换。

`Alt↓` 或 `Alt↑`：向下或向上移动当前行。

`Shift + Alt + ↓` 或 `Shift + Alt + ↑`：向下或向上复制当前行。

`Shift + Cmd + K`：删除当前行。

`Shift + Cmd + \`：光标跳至匹配的括号。

`Cmd + ]` 或 `Cmd + [`：增加或减小缩进。

`Cmd + ←` 或 `Cmd + →`：移至当前行的最前或最后。

`Cmd + ↑` 或 `Cmd + ↓`：移至当前文件的最前或最后。

`Alt + →` 或 `Alt + ←`：向右或向左按单词移动光标。

`Alt + Cmd + [` 或 `Alt + Cmd + ]`：折叠或展开光标所在区域。

`Cmd + K + Cmd + [` 或 `Cmd + K + Cmd + ]`：折叠或展开光标所在区域的子区域。

`Cmd + K + Cmd + 0` 或 `Cmd + K + Cmd + j`：折叠或展开所有区域。

`Cmd + /`：注释或解除注释光标所在行或代码块。

多光标 / 选择

`Alt + 点击鼠标`：插入光标。

`Alt + Cmd + ↑` 或 `Alt + Cmd + ↓`：向上或向下插入光标。

`Cmd + U`：撤销最后一个光标操作。

`Shift + Alt + I`：向选中的代码块的每一行最后插入光标。

`Cmd + G` 或 `Shift + Cmd + G`：查找下一个或上一个。

`Cmd + F2`：选中所有鼠标已选择的字符。

`Shift + Cmd + L`：选中所有鼠标已选择部分。

`Alt + Enter`：选中所有查找匹配上的关键字。

`Shift + Alt + 拖拽鼠标`：选择多列编辑。

`Shift + Alt + Cmd + ↑` 或 `Shift + Alt + Cmd + ↓`：上下选择多列编辑。

`Shift + Alt + Cmd + ←` 或 `Shift + Alt + Cmd + →`：左右选择多列编辑。

12.界面风格设置

12.1. 个人设置

您可以通过个人设置功能，自定义您的DataStudio模块和编辑器的界面显示、主题风格。本文为您介绍个人设置的相关功能。

进入个人设置

1. 登录DataWorks控制台，选择目标区域后，在左侧导航栏单击**工作空间列表**。
2. 单击目标工作空间后的**进入数据开发**，默认进入该工作空间的DataStudio功能模块。
3. 在DataStudio界面左侧导航栏底部单击图标，默认进入**设置 > 个人设置**页面。

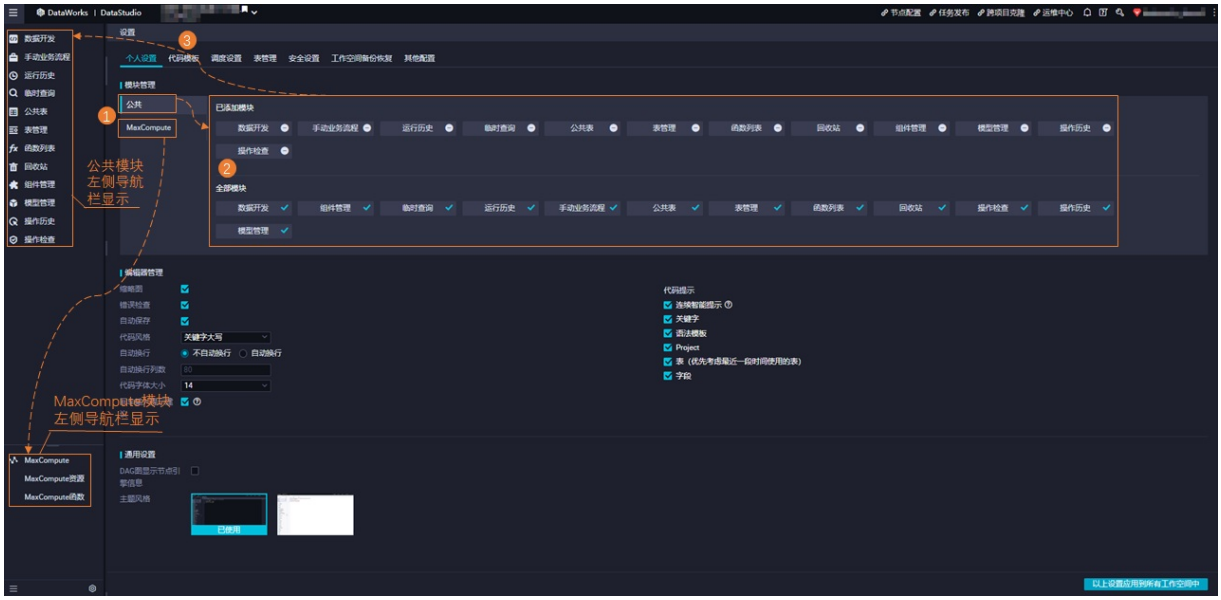
   在该页面，您可以进行如下设置：

   - 设置DataStudio显示的功能模块，详情请参见模块管理。
   - 设置代码编辑器的缩略图、错误检查、字体风格等功能的默认配置，详情请参见编辑器管理。
   - 设置DataStudio的通用主题风格，详情请参见通用设置。

   设置完成后，您可以单击**个人设置**页面右下角的**以上设置应用到所有工作空间中**，即可批量修改当前登录账号下所有工作空间的配置。

模块管理

模块管理区域，用于设置DataStudio左侧导航栏模块的显示情况。

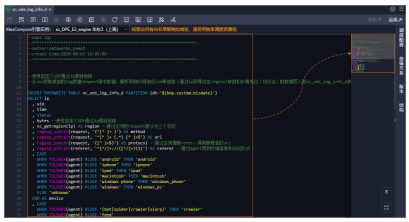
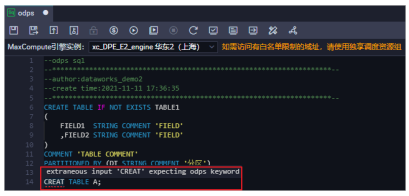
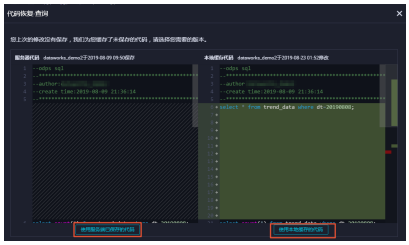


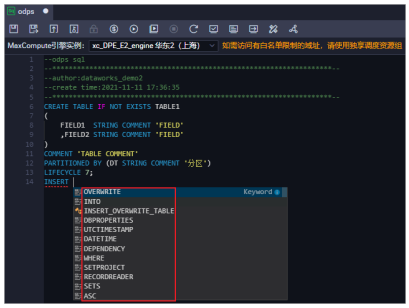
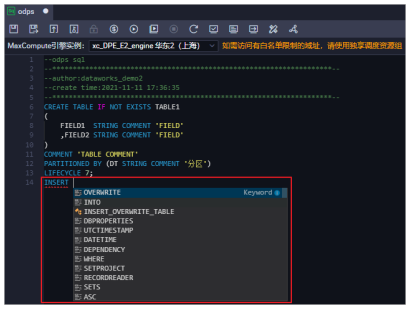
您可以选择切换**公共**或**MaxCompute**，在对应的**全部模块**中选择需要显示的模块，将其添加至**已添加模块**。添加后，该模块将会显示在左侧导航栏中。您可以根据不同功能模块进行相关数据开发操作。

> **说明**
>
> - 如果您在DataStudio左侧导航栏无法找到目标模块，则可能是因为该模块未设置为显示状态，您可以在**模块管理**中将其添加为显示模块。
> - 如果您需要取消DataStudio左侧导航栏中显示的目标模块，则可以在**模块管理**中相应的**已添加模块**区域，单击该模块后的图标，即可在DataStudio左侧导航栏将其移除。模块被移除后，仅仅只是在DataStudio左侧导航栏中不显示，该模块实际在DataWorks中仍然存在。

编辑器管理

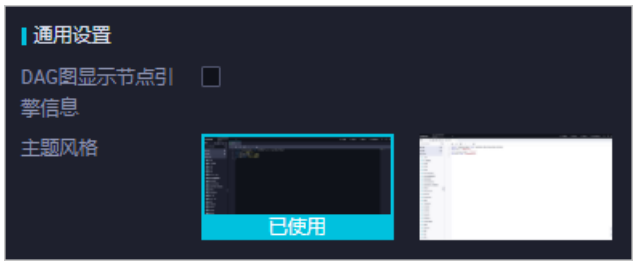
编辑器管理区域，用于设置代码及关键字的相关功能，该设置实时生效。

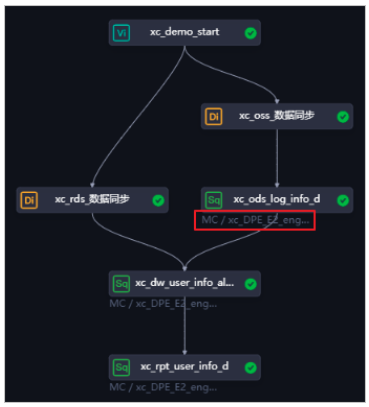
功能	描述	效果展示
缩略图	用于对编辑器界面的代码进行缩略显示。当代码较长时，您可以在缩略图中移动鼠标，来切换需要显示的代码区域。	
错误检查	用于检查当前代码中的错误语句。当鼠标放置标有红色底纹的代码字段时，则会显示该字段的具体报错。	
自动保存	用于对当前编辑的代码自动缓存，避免在编辑过程中，页面出错导致代码无法保存。您可以选择**使用服务端已保存的代码**或**使用本地缓存的代码**。	

功能	描述	效果展示
代码风格	用于设置代码关键字为大写或小写。	
自动换行	用于设置编写代码时是否自动换行。 • **不自动换行** • **自动换行**	无
自动换行列数	• 当设置代码为**不自动换行**，则自动换行列数默认为*80*，且不可更改。 • 当设置代码为**自动换行**，则可以自行设置换行的列数。	无
代码字体大小	代码字体大小支持设置为*12*~*18*号字体。您可以根据代码书写习惯和数量更改设置。	无
回车输入提示建议	开启该功能后，当您在编辑器输入回车键时，系统会自动弹出代码提示建议。 **说明** • 关闭该功能，则可减少*换行操作*和*键入提示建议*之间的歧义。 • 除了使用回车键外，您还可以使用Tab键输入提示建议。	
代码提示	用于在输入代码时，提供如下智能提示： • **连续智能提示**：智能感知输入目的，连续给出下一步输入提示建议。 • **关键字**：提示代码支持输入的关键字。 • **语法模板**：代码支持的语法模板。 • **Project**：输入联想的项目名称。 • **表**：联想需要输入的表，优先考虑最近一段时间使用的表。 • **字段**：智能提示选择目标表中的字段。	无

通用设置

通用设置区域，用于设置DAG图中节点的引擎显示及DataStudio的整体界面风格。



功能	描述	效果展示
DAG图显示节点引擎信息	用于设置是否在业务流程的DAG图中显示节点的引擎信息。	
主题风格	用于设置DataStudio的主题背景颜色为黑色或白色。	无

12.2. 代码模板

代码模板是在创建节点后，默认展示在该节点代码编辑器界面最前端的内容。您可以根据实际需求设置ODPS SQL、ODPS MR、SHELL类型节点的代码模板。

使用限制

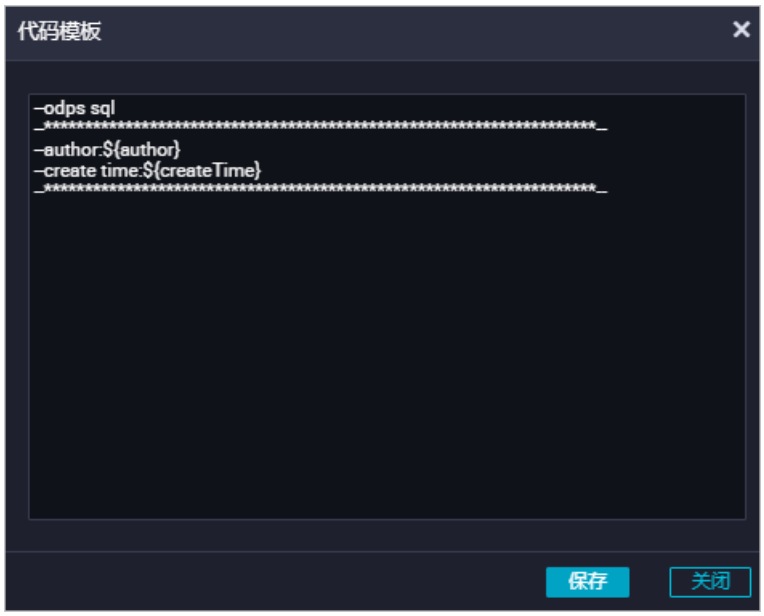
- 仅工作空间管理员可以修改代码模板。如果您需要修改代码模板，则可以授权目标账号为空间管理员角色权限，详情请参见规划与配置角色权限。
- 目前仅支持设置ODPS SQL、ODPS MR、SHELL类型节点的代码模板。

设置代码模板

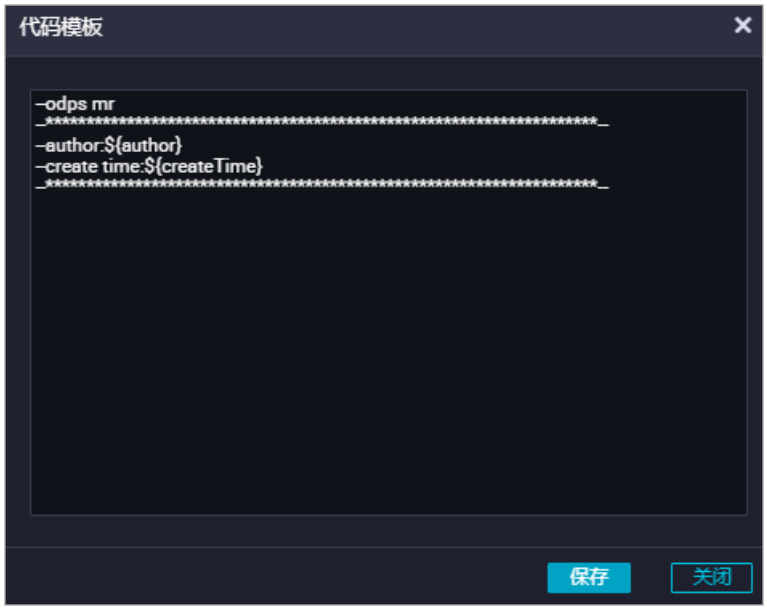
1. 进入代码模板。
   i. 登录DataWorks控制台，选择目标区域后，在左侧导航栏单击**工作空间列表**。
   ii. 单击目标工作空间后的**进入数据开发**，默认进入该工作空间的DataStudio功能模块。
   iii. 在DataStudio界面左侧导航栏底部单击图标，进入**设置**页面。
   iv. 在**设置**页面，单击**代码模板**，进入**代码模板**页面。
2. 设置代码模板。
   i. 编辑模板。

在**代码模板**页面，单击相应模板后的**编辑**，即可根据需求修改该类型节点的代码模板。DataWorks为您提供的三种类型节点的默认模板如下：

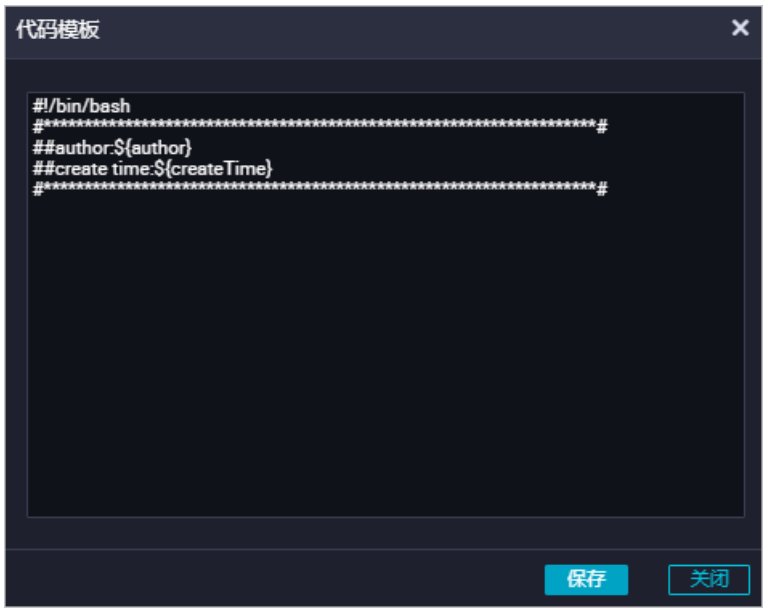
- **ODPS SQL模板**



- **ODPS MR模板**



- SHELL模板



ii. 单击**保存**，完成设置。

后续创建的相应节点便会使用该模板。

12.3. 调度设置

您需要进入DataStudio的调度设置页面启用调度周期后，周期任务才能自动调度运行。本文为您介绍如何开启调度功能并设置相应调度参数的默认配置。

设置周期任务的默认调度配置

1. 登录DataWorks控制台，选择目标区域后，在左侧导航栏单击**工作空间列表**。
2. 单击目标工作空间后的**进入数据开发**，默认进入该工作空间的DataStudio功能模块。
3. 在DataStudio界面左侧导航栏底部单击图标，进入**设置**页面。
4. 单击**调度设置**，设置调度任务相关功能的默认配置。

功能项	描述
启用调度周期	开启该功能后，当前工作空间下的所有周期任务才会自动调度运行。
调度资源组	任务调度运行时默认使用的资源组。
数据集成资源组	数据集成任务运行时默认使用的集成资源组。
重跑属性	周期任务运行时默认的重跑策略。 ○ **运行成功或失败后均可重跑** ○ **运行成功后不可重跑，运行失败后可以重跑** ○ **运行成功或失败后皆不可重跑** **说明** 当**重跑属性**配置为可重跑时，应尽量保证任务的幂等性，避免多次重跑出现数据质量问题。
自动重跑次数	周期任务调度执行失败情况下，默认自动重跑的次数。 重跑次数最少配置为1（即任务出错后自动重跑1次），最多配置为10（即任务出错后会自动重跑10次）。您可以根据业务需求进行修改。

功能项	描述
重跑间隔	周期任务重跑时默认的重跑时间间隔。 时间间隔最小支持设置为1分钟，最大支持设置为30分钟。
自动解析	周期任务是否启用自动解析功能。启用后，提交节点时，系统会根据最新代码解析出本节点及其依赖的上游节点的输出名称。

5. 单击**保存配置**，成功设置周期任务的默认调度配置。

   调度设置配置完成后，新建的周期任务将会使用相关功能的默认配置。

12.4. 表管理

您可以在表管理设置页面，制定分区格式、分区字段命名、表前缀定义、以及表的主题、层级等进行设置管理。

使用限制

仅工作空间管理员可以添加多个主题，并根据用途、名称等，对表进行分类。如果您需要添加主题，则可以授权目标账号为空间管理员角色权限，详情请参见规划与配置角色权限。

进入表管理

1. 登录DataWorks控制台，选择目标区域后，在左侧导航栏单击**工作空间列表**。
2. 单击目标工作空间后的**进入数据开发**，默认进入该工作空间的DataStudio功能模块。
3. 在DataStudio界面左侧导航栏底部单击图标，进入**设置**页面。
4. 在**设置**页面，单击**表管理**，进入**表管理**页面。

   在该页面，您可以进行如下设置：

   - 设置表的基本格式，详情请参见设置表格式。
   - 设置表的主题，详情请参见主题管理。
   - 设置表的层级，详情请参见层级管理。

   配置完成后，单击**保存配置**。

设置表格式

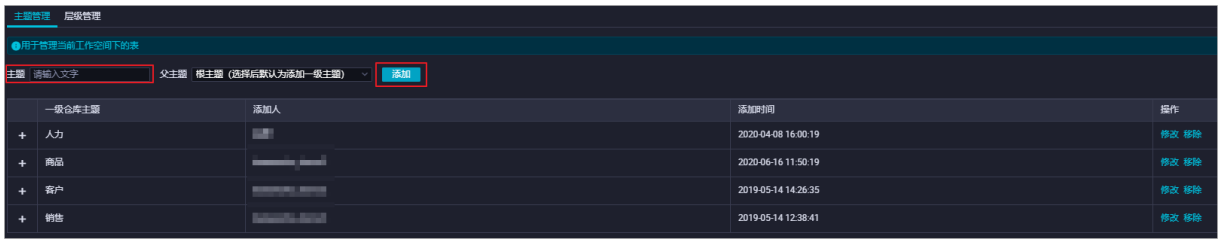
您可以在**表管理**页面，设置分区表的格式及前缀要求。

功能	描述
分区日期格式	分区表的日期格式。系统默认设置的分区日期格式为**YYYYMMDD**。

功能	描述
分区字段命名	分区字段的标识，建议使用**dt**。
临时表前缀	临时表的标识。默认为**t_**。
上传表（导入表）前缀	标识上传至DataStudio的表。示例表的前缀为**upload_**。

主题管理

表管理页面存在非常多的表，您可以根据选取的主题，将表存放在一级主题所在的文件夹下。用于对表进行归纳的文件夹，即为主题。

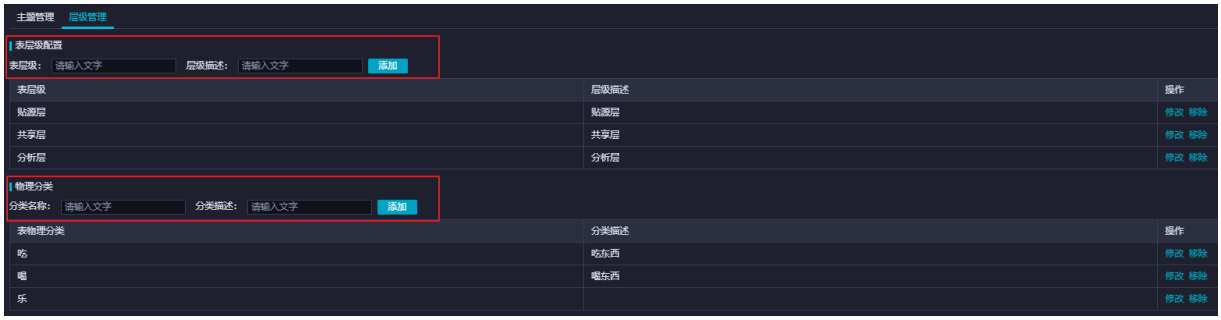


在**主题**输入框输入需要创建的主题，单击**添加**，即可创建所需要的主题类别。您也可以根据实际需求单击相应主题**操作**列的**修改**或**删除**，修改或删除目标主题。

层级管理

层级管理用于对表的物理层级进行设计。表层级是用来定义和管理数据仓库分层的，通常，我们推荐将数据仓库分层设置为数据引入层（ODS）、明细粒度事实层（DWD）、公共汇总粒度事实层（DWS）、数据应用层（ADS）等层级，详情请参见创建数仓分层。而表的物理分类，则是允许您从业务的使用视角对表进行更详细的分类。

> **说明** 工作空间不存在默认的层级，需要项目所有者或者项目管理员根据需求手动添加。



- **表层级配置**：用于定义表所属的层级。您可以根据业务需求，创建、修改、删除表层级。
  - 创建层级：在**表层级**输入层级名称，在**层级描述**输入该层级的介绍信息，单击**添加**，即可创建新的层级。
  - 修改或删除层级：单击目标层级**操作**列的**修改**或**移除**，即可修改或删除该层级。
- **物理分类**：用于定义表所属的类别。您可以根据业务需求，创建、修改、删除表分类。
  - 创建分类：在**分类名称**输入新建的分类名称，在**分类描述**输入该分类的介绍信息，单击**添加**，即可创建新的分类。
  - 修改或删除分类：单击目标分类**操作**列的**修改**或**移除**，即可修改或删除该分类。

12.5. 安全设置

安全设置功能用于控制在当前DataWorks工作空间中使用数据开发（DataStudio）执行查询操作时，是否对返回结果涉及的敏感信息进行脱敏展示。

背景信息

DataStudio内置了数据脱敏规则。如果您开启了**启用页面查询内容脱敏**，则当您在DataStudio中运行代码后返回的数据命中了该脱敏规则，系统会对显示的数据做脱敏处理。如果在系统内置的脱敏规则之外，您需要自定义脱敏规则，请使用数据保护伞功能，详情请参见数据保护伞。

> **说明**
> - **启用页面查询内容脱敏**开关的生效范围是当前工作空间。
> - 该脱敏操作为动态脱敏，不会影响底层存储的数据。

使用限制

安全设置是对DataWorks工作空间范围生效，如果您需要所有工作空间在查询数据时对敏感信息均脱敏展示，则需要对所有工作空间逐一开启该功能。

> **说明** 例如，工作空间A设置了脱敏展示，工作空间B未设置。如果工作空间B具有查看工作空间A中表的权限，则通过工作空间B查看工作空间A中的表时，将会看到明文结果。

启用查询脱敏

1. 进入安全设置页面。
   i. 登录DataWorks控制台，选择目标区域后，在左侧导航栏单击**工作空间列表**。
   i. 单击目标工作空间后的**进入数据开发**，默认进入该工作空间的DataStudio功能模块。
   ii. 在DataStudio界面左侧导航栏底部单击图标，进入**设置**。
   iii. 在**设置**页面，单击**安全设置**，进入**安全设置**页面。
2. 在**安全设置**页面，打开**启用页面查询内容脱敏**开关。

   启用查询内容脱敏后，该配置立即生效。后续在DataStudio模块执行查询操作时，返回结果将根据DataWorks的内置脱敏规则进行脱敏展示。

   > **说明**
   > - 该功能可与数据保护伞模块配合使用，如果在系统内置的脱敏规则之外，您需要自定义脱敏规则，请使用数据保护伞功能，详情请参见数据保护伞。
   > - 如果您未启用查询内容脱敏功能，则可能会造成敏感信息泄露。

   DataWorks的内置脱敏规则如下表所示。

类型	脱敏规范	原始数据	脱敏后数据
身份证	仅显示首位及末位，适用于15位和18位的身份证。	111222190002309999	1**********9

类型	脱敏规范	原始数据	脱敏后数据
手机号	仅显示前7位，并使用 * 符号代替后四位，适用于中国大陆手机号。	13900001234	1390000****
邮箱	○ @ 符号前大于等于3位，则仅显示前3位。 ○ @ 符号前不够3位，则显示全部，后面添加3个 * 符号。	○ username@example.com ○ a@example.net	○ use***@example.com ○ a***@example.net
银行卡	仅显示最后4位，适用于信用卡和储蓄卡。	○ 6888 8888 8888 8888 ○ 4666 6666 6666 6666	○ **** **** **** 8888 ○ **** **** **** 6666
IP或MAC地址	仅保留第1段内容，其余几段内容替换为 * 符号。	○ 192.168.0.1 ○ 01-80-C2-00-00-00	○ 192.***.*.* ○ 01-**-**-**-**-**
车牌号	地区信息+车牌号后3位的明文显示，其余均使用 * 符号显示。	○ 杭A 666666 ○ 杭A 888888	○ 杭A***666 ○ 杭A***888

12.6. 工作空间备份恢复

工作空间备份恢复可以用于不同工作空间之间进行代码迁移，本文将为您介绍如何进行工作空间的备份和恢复。

单击**数据开发**页面左下角的，在右侧展开**设置**页面。

单击菜单栏中的**工作空间备份恢复**，即可进入**工作空间备份恢复**页面。

- **工作空间备份**可以将工作空间下的节点代码、节点依赖关系、资源和函数等整体打包为一个压缩文件。
- **工作空间恢复**可以为恢复的工作空间保留原有的调度配置，目标工作空间恢复成功后，所有节点处于已保存但未提交的状态。

进入工作空间备份恢复

1. 登录DataWorks控制台，单击相应工作空间后的**进入数据开发**。
2. 单击左下角的，在右侧展开**设置**页面。
3. 单击顶部菜单栏中的**工作空间备份恢复**。

工作空间备份

项目备份会以压缩包的形式，将当前工作空间下的节点代码、节点依赖关系、资源和函数等压缩为资源包。

- 仅项目管理员可以导出配置和恢复配置。

- 备份时无法对旧版工作流和组合节点进行备份，建议您使用业务流程进行开发。
- 在工作空间的同一路径下相同的任务，备份时会覆盖原有的任务，建议您创建新的工作空间进行工作空间恢复。
- 工作空间备份不会备份表数据，您可以通过以下方式同步数据：
  - 进入**工作空间管理 > 数据源管理**页面配置MaxCompute数据源，通过创建同步任务进行备份。
  - 在工作空间A中，通过执行DDL语句 `create table select * from 工作空间B.表名` 进行数据的迁移。

1. 进入**工作空间备份恢复 > 备份**页面，单击右上角的**新建备份**。
2. 在**新建备份**对话框中，选择**备份方式**和**备份版本格式**。
   - **备份方式**

     新建备份时，您可以选择**全量备份**或**增量备份**。
     - **全量备份**：备份整个工作空间下所有的节点代码、节点依赖关系、资源和函数。
     - **增量备份**：您可以选择**增量开始日期**，即备份从选择的增量开始时间到当前时间这一时间段内新增或修改的节点。

     > **说明** 增量备份时请注意增量同步任务间的依赖关系，依赖关系不正常会导致工作空间恢复失败，建议您使用**全量备份**。
   - **备份版本格式**

     备份包括**公共云**、**专有云3.6.1-3.8.1**和**专有云<3.6.1**三种版本格式。
3. 配置完成后，单击**开始备份**。

工作空间恢复

1. 进入**工作空间备份恢复 > 恢复**页面，单击右上角的**新建恢复**。
2. 在**新建恢复**对话框中，**选择恢复文件**。

   > **说明** 您可以上传备份的压缩文件至当前工作空间。
3. 配置完成后，单击**开始恢复**。
4. 在**设置计算引擎映射**对话框中，选择**目标工作空间计算引擎实例**。

如果备份有多个计算引擎的工作空间，在恢复过程中会扫描所有的计算引擎实例信息，且仅恢复当前工作空间下已有的计算引擎任务。因此需要配置计算引擎的映射关系，才可以继续恢复工作空间的备份。

> 说明
> - 如果当前恢复的工作空间下没有某个引擎类型，例如EMR，或者该计算引擎类型下没有该计算引擎实例时，不会恢复该引擎类型的节点。
> - 不同地域的自定义节点类型不一致，因此同样需要建立映射关系。例如现有的Hologres开发、Data Lake Analytics、AnalyticDB for MySQL和AnalyticDB for PostgreSQL等自定义节点。

12.7. 其他配置

DataWorks支持丰富的数据开发配置，您可以在DataStudio的**其他配置**页面开启代码强制评审，配置代码审核人员，把控开发任务的代码质量；也可以批量删除所有不再使用的DataBlau DDM数据模型。本文为您介绍**其他配置**页面的配置要点。

使用限制

删除DataBlau DDM数据模型的限制条件如下：

- 仅在DataWorks数据建模 (DataBlau DDM) 商品到期后不再开通该功能时，可执行删除操作。
- 仅**空间管理员**角色支持删除DataBlau DDM数据模型。

进入其他配置

1. 登录DataWorks控制台，选择目标区域后，在左侧导航栏单击**工作空间列表**。
2. 单击目标工作空间后的**进入数据开发**，默认进入该工作空间的DataStudio功能模块。
3. 在DataStudio界面左侧导航栏底部单击图标，进入**设置**页面。
4. 在**设置**页面，单击**其他配置**，进入**其他配置**页面。

   在该页面，您可以执行如下操作：
   - 配置开启代码强制评审，详情请参见开启代码强制评审。
   - 批量删除DataBlau DDM数据模型，详情请参见批量删除DataBlau DDM数据模型。

开启代码强制评审

代码强制评审功能，用于严格把控开发任务的代码质量。开启代码强制评审后，您可以配置**任意开发角色用户**或**指定开发角色用户**为项目代码的评审人，后续在完成任务代码开发，提交代码时，需要该评审人审批通过后，任务代码才能发布。

代码评审的基线范围包括**1级基线任务**、**3级基线任务**、**5级基线任务**、**7级基线任务**、**8级基线任务**及**非基线任务**，其相关说明如下：

- 基线是管理任务优先级的工具，如果您需要对重要任务开启代码评审，则可以勾选对应优先级的基线，详情请参见基线管理。
- 当指定了对应级别的基线任务，该级别的任务在发起评审后，必须由评审人员对任务代码审核通过才可进行发布。
- 多个任务并发启动运行时，高优先级的任务会优先抢占资源。此时，如果您对高级别的任务指定了指定强制代码评审基线范围，则评审人员可以对任务的代码质量进行把控，防止由于任务代码有误，未经审核直接发布后报错，长时间占用资源。更多代码评审的相关内容，详情请参见代码评审。

批量删除DataBlau DDM数据模型

当您的DataWorks数据建模 (DataBlau DDM) 商品到期后，后续不需要再使用该功能，则可以在**DataBlau DDM**区域，单击**删除**，查看当前DataWorks工作空间中已创建的模型数量，并批量删除所有DataBlau DDM数据模型。

> **说明**
> - 仅在DataWorks数据建模 (DataBlau DDM) 商品到期后不再开通该功能时，可执行删除操作。
> - 仅**空间管理员**角色支持删除DataBlau DDM数据模型。
> - 删除数据模型不会影响计算引擎内已存在的表结构与数据内容，但被删除的DataBlau DDM数据模型将永久不可恢复，请您谨慎操作。

12.8. 工作空间配置

工作空间配置页面包括分区日期格式、分区字段命名、临时表前缀、上传表（导入表）前缀和启用页面查询内容脱敏共计5个配置项。

单击**数据开发**页面左下角的，即可在右侧展开**设置**页面。

单击顶部菜单栏中的**工作空间配置**，即可配置工作空间。

参数	描述
分区日期格式	默认参数、代码中参数的显示格式，您也可以根据自己的需求修改参数的格式。
分区字段命名	分区中默认的字段名称。
临时表前缀	以t_开头的字段，默认识别为临时表。
上传表（导入表）前缀	在DataStudio页面上传表时，表的名称前缀。
启用页面查询内容脱敏	启用后，当前工作空间下的临时查询任务返回的结果将会被脱敏。

开启DataWorks工作空间的查询脱敏

DataWorks的脱敏需要在每个工作空间进行逐一开启。开启脱敏后，会脱敏当前工作空间下的临时查询任务返回的结果。由于仅仅是动态脱敏，不会影响底层存储的数据。

> **说明** 例如，工作空间A设置了展示脱敏，但工作空间B没有设置。如果您可以从工作空间B访问工作空间A中的表，将会看到明文结果。

进入**项目配置**页面，启动**启用页面查询内容脱敏**选项。单击**保存配置**，即可开启DataWorks工作空间的查询脱敏。

> **说明** DataWorks默认不允许下载和数据脱敏。

DataWorks查询脱敏配置成功后，默认对以下数据进行脱敏。

类型	蚂蚁脱敏规范	原始数据	脱敏数据
身份证	仅显示前1后1位，适用于15位和18位的身份证。	111222190002309999	1*************9
手机号	仅显示前7位，使用*代替后四位，适用于大陆手机号。	13900001234	1390000****
邮箱	@前仅展示前3位，如果不够3位则显示全部，后面跟3个*。	• username@example.com • a@example.net	• use***@example.com • a***@example.net
银行卡	仅显示最后4位，适用于信用卡和储蓄卡。	• 6888 8888 8888 8888 • 4666 6666 6666 6666	• **** **** **** 8888 • **** **** **** 6666
ip/mac地址	仅保留第1段。	• 192.168.0.1 • 01-80-C2-00-00-00	• 192.***.*.* • 01-**-**-**-**-**
车牌号	地区信息+车牌后3位显示明文，其它均用*展示。	• 杭A 666666 • 杭A 888888	• 杭A***666 • 杭A***888

> **说明** 如果需要对更多类型的数据进行脱敏，或者对脱敏格式有自定义要求，请使用数据保护伞的脱敏配置。工作空间开启脱敏功能必须配合数据保护伞使用，详情请参见数据保护伞模块。