ALIBABA CLOUD

Alibaba Cloud

DataWorks

Data Development

Document Version: 20220713

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	**Danger:** Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	**Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Overview

DataStudio of DataWorks allows you to perform operations on a GUI to develop and test data in big data scenarios. This way, you can manage data in an intelligent and efficient manner. This topic describes the following aspects of DataStudio: nodes, supported node types, management and usage of resources during development, and management of permissions on resources and service modules during development.

> **Note**
>
> - This topic describes how to use DataStudio in a workspace in standard mode. In standard mode, the development and production environments are isolated.
> - If you change the code for a node in the production environment, you must modify node parameters on the DataStudio page. Then, commit and deploy the node.
> - If no engine is available in your workspace or the engine that you want to use is not displayed in the directory tree, check whether the engine is activated and associated with your workspace on the Workspace Management page. Only the engines that are associated with a workspace are displayed in a workflow. For more information about how to associate an engine with a workspace, see Configure a workspace.
> - If you cannot perform operations on specific service modules or cannot find an add entry, go to the User Management page to check whether you have development permissions. You have development permissions if you use an Alibaba Cloud account or a RAM user that is assigned the developer role or workspace administrator role. You can also check whether the DataWorks edition meets your requirements.

Organizational structure

You can organize your business based on **workspaces**, **solutions**, and **workflows**. You can plan and group workspaces based on enterprise departments, business projects, and data warehouse layers.

Concept	Description	Purpose
Workspace	You can specify administrators and members for each workspace based on your business requirements. The role settings of members and parameters for a compute engine instance are different among workspaces. For more information about workspace planning, see Plan workspaces.	Workspaces are basic units for managing permissions in DataWorks. You can create workspaces based on the organizational structure of your company. You can use a workspace to manage development permissions and O&M permissions. Workspace members can collaborate to develop and manage the code for all nodes in a workspace.

Concept	Description	Purpose
Solution	A solution is a group of workflows that are dedicated to a specific business goal. A workflow can be added to multiple solutions. After you develop a solution and add a workflow to the solution, other users can reference and modify the workflow in their solutions or workflows for collaborative development.	You can use a solution for business integration.
Workflow	A workflow is an abstract business entity that allows you to develop code based on your business requirements. Workflows and nodes in different workspaces are separately developed. Workflows can be displayed in a directory tree or in a panel. The display modes enable you to organize code from the business perspective and show the resource classification and business logic in a more efficient manner. • The directory tree allows you to organize your code by node type. • The panel shows the business logic in a workflow.	A workflow is a basic unit for code development and resource management.

DataStudio works based on nodes in a workflow. You can create one or more nodes in a specific workflow in the panel. In each workflow, nodes are grouped by engine type. In the section of a specific engine, nodes are classified into data synchronization nodes, tables, resources, and functions. These components can be used to meet a specific business goal. Only the components that are used in a workflow are displayed in the workflow.

- To use DataStudio, you must create a workflow.
- If you change the code for a node in the production environment, you must modify node parameters on the DataStudio page. Then, commit and deploy the node.

Development logic

Workflows and nodes are required for data development. You can select a manually triggered node or an auto triggered node for data development. You can select engine nodes, control nodes, or custom nodes to cleanse data. If you select an auto triggered node, you must configure scheduling parameters and commit the node to the Create Package page. Then, deploy the node in the Nodes to Deploy panel. After the node is deployed, the node is in the production environment and is scheduled based on the scheduling parameters that you configure.

Main features of DataStudio

- Node type

  Engine capabilities are encapsulated into DataWorks. You do not need to use complex engine CLIs. DataWorks provides custom wrappers for custom nodes. This allows you to add computing task types and use custom nodes to access custom computing services. You can use custom nodes with system nodes of DataWorks to process complex data.

  - Use data synchronization nodes in Data Integration to synchronize data.
  - Use engine nodes to develop data.
  - Use engine nodes and general nodes to manage complex processes.
  - Use custom nodes to develop data.

  For more information about how to select a node type, see the Select a data development node section in this topic.

- Node development

  Allows you to select a manually triggered node or an auto triggered node and perform operations on a GUI in an intelligent and efficient manner to develop data.

  - Hybrid orchestration mode: allows you to drag different types of engine nodes in a workflow to the canvas and view the result. For more information, see **Create an auto triggered node** of the Select a node type section in this topic.
  - AI-powered SQL editor: supports code hinting and displays the structure of SQL operators. For more information, see **View node code and manage node versions** of the Select a node type section in this topic.

  For more information about how to create an auto triggered node and a manually triggered node, see Select a node type.

- Visualized management and use of tables, resources, and functions

  For more information, see the Manage and use tables, resources, and functions section in this topic.

- Permission and development behavior management of members

  The permissions on the following items are managed in DataStudio:

  - Resources
  - GUI-based operations
  - Operation procedure

  For more information, see the Manage the permissions and development behavior of members section in this topic.

- Code version management and operational audit

  You can use ActionTrail in the following scenarios:

- Obtain the audit logs of operations that are performed by developers on a GUI.
- Configure parameters in advance to collect important data for cause analysis.
- Audit permissions on a MaxCompute table.
- Restore table data and nodes.
- Compare and roll back node versions.

For more information, see the ActionTrail section in this topic.

Select a data development node

Engine capabilities are encapsulated as engine nodes in DataWorks. This way, you do not need to use complex engine CLIs. DataWorks provides general nodes that you can use together with engine nodes to manage complex processes. DataWorks also provides custom wrappers for custom nodes. This allows you to add computing task types and use custom nodes to access custom computing services. You can use custom nodes to customize code processing methods.

> **Note** More features will be available in the future.

- Data synchronization node in Data Integration: You can use a data synchronization node in Data Integration to synchronize data.

Data synchronization node in Data Integration	Scenario
Batch synchronization node	A batch synchronization node is used for offline data synchronization. ○ Batch synchronization nodes support different types of heterogeneous data sources in complex scenarios. These nodes are used to synchronize data based on a data transmission framework by using a reader and a writer, which are abstract data extraction and writing plug-ins. ○ A batch synchronization node supports more than 40 data sources of the following categories: relational databases, unstructured storage, big data storage, and message queues. For more information about the data sources that are supported by a batch synchronization node, see Supported data source types, readers, and writers.
Real-time synchronization node	A real-time synchronization node for real-time data synchronization. A real-time synchronization node uses three basic plug-ins to read, convert, and write data. These plug-ins interact with each other based on an intermediate data format that is defined by the plug-ins. For more information about the data sources that are supported by a real-time synchronization node, see Plug-ins for data sources that support real-time synchronization.

Data synchronization node in Data Integration	Scenario
Data synchronization solution	DataWorks provides solutions for various data synchronization scenarios, such as real-time synchronization, offline full synchronization, and offline incremental synchronization. These solutions help enterprises migrate data to the cloud in a more efficient and convenient manner. Synchronization solutions provide the following benefits: ○ Initializes full data. ○ Writes incremental data in real time. ○ Automatically merges full and incremental data at a scheduled time and writes the data to the new partition of a full table. For more information about synchronization solutions, see Overview.

- Engine node: You can use an engine node to develop data.

  In a workflow, you can find an engine and create a node of the engine to issue the engine code to the data cleansing engine to run.

Engine integrated with DataWorks	Encapsulated engine capability
MaxCompute	○ Create an ODPS SQL node ○ Create a MaxCompute Spark node ○ Create a PyODPS 2 node ○ Create a PyODPS 3 node ○ Create an ODPS Script node ○ Create an ODPS MR node
E-MapReduce	○ Create an EMR Presto node ○ Create an EMR Hive node ○ Create and use an EMR MR node ○ Create an EMR Spark SQL node ○ Create and use an EMR Spark node ○ Create and use an EMR Shell node ○ Create and use an EMR Spark Streaming node
AnalyticDB For PostgreSQL	Create an ADB for PostgreSQL node
AnalyticDB For MySQL	Create and use an AnalyticDB for MySQL node
Hologres	Create a Hologres SQL node
Database	Create a MySQL node

Engine integrated with DataWorks	Encapsulated engine capability
ClickHouse	Create and use a ClickHouse SQL node
Algorithm	Create a Machine Learning (PAI) node

- General node: You can use general nodes together with engine nodes to manage complex processes.

  You can create a general node and use the node together with engine nodes to manage complex processes in a workflow.

Scenario	Node type	Description
Business management	Create a zero-load node	A zero load node is a control node that supports dry-run scheduling and does not generate data. In most cases, a zero load node serves as the root node of a workflow and allows you to manage nodes and workflows.
Event triggering	Create an HTTP Trigger node	Use this type of node if you want to schedule nodes in DataWorks after nodes in other systems finish running.
	OSS Object Inspection node	Use this type of node if you want to trigger a descendant node to run by monitoring whether Object Storage Service (OSS) objects are generated.
	Create an FTP Check node	Use this type of node if you want to trigger a descendant node to run by monitoring whether File Transfer Protocol (FTP) files are generated.
Parameter value assignment	Configure an assignment node	Use this type of node if you want to use the outputs parameter of an assignment node to pass the data from the output of the last row of the code for the assignment node to its descendant nodes.
Control	Configure a for-each node	Use this type of node to traverse the result set of an assignment node.
	Configure a do-while node	Use this type of node to execute logic of specific nodes in loops. You can also use this type of node together with an assignment node to generate the data that is passed to a descendant node of the assignment node in loops.
	Configure a branch node	Use this type of node to route results based on logical conditions. You can also use this type of node together with an assignment node.
	Configure a merge node	Use this type of node to merge the status of its ancestor nodes and prevent dry-run of its descendant nodes.
Parameter passing	Create a parameter node	Use this type of node to aggregate parameters of its ancestor nodes and distribute parameters to its descendant nodes.

passing Scenario	Node type	Description
	Create a Shell node	Shell nodes support standard shell syntax. The interactive syntax is not supported.
Code reuse	Create an SQL component node	An SQL component node is an SQL script template that contains multiple input and output parameters. You can create and run an SQL component node to filter source table data, join source tables, and aggregate source tables to generate a result table. **Note** Only the MaxCompute SQL syntax is supported.

- Custom node: You can use custom nodes to develop data.

  DataWorks provides custom wrappers for custom nodes. This allows you to add computing task types and use custom nodes to access custom computing services. You can write code for a custom wrapper on your on-premises machine and add the code to DataWorks on the node configuration page. When you use DataStudio, you can select the custom wrapper from the custom code group of a workflow to develop data.

  The following table describes how to use a custom node.

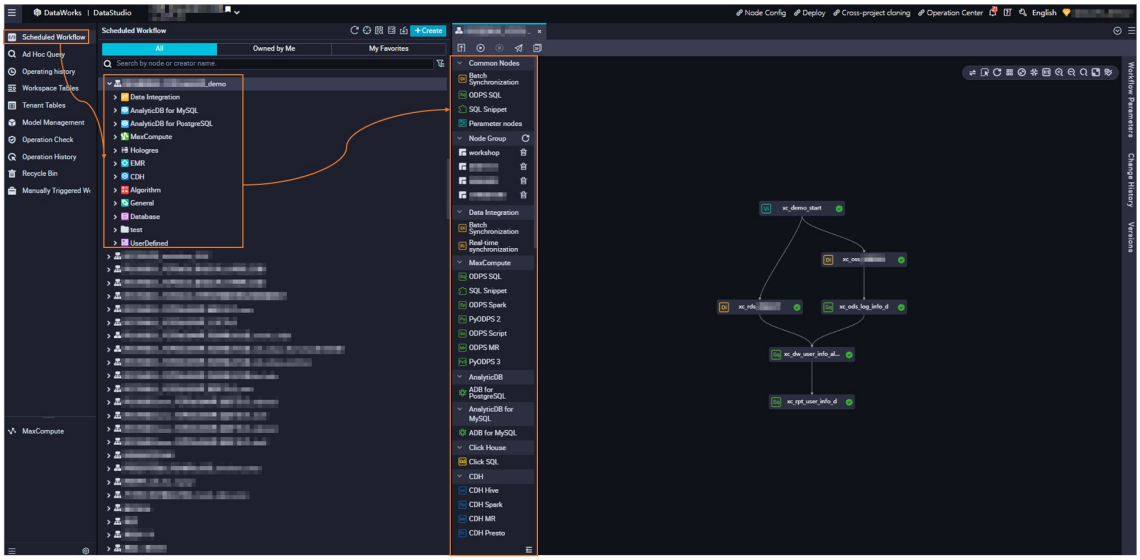
Procedure	Description
Step 1: Develop a custom wrapper package	To run a task by using a custom node, you must use a custom wrapper. Before you can use a custom node, you must create a custom wrapper package. Then, upload and deploy the package to DataWorks.
Step 2: Create a wrapper	Deploy the custom wrapper in DataWorks.
Step 3: Create a custom node type	Add a custom node and configure the relationship between the custom node and the custom wrapper. Then, configure the basic information, code editor, and interaction parameters of the custom node.

Select a node type

In DataWorks, you can create auto triggered nodes or manually triggered nodes to develop data. To create a node, you can right-click a workflow in the directory tree on the left, or double-click a workflow and drag a specific node type to the canvas.

- **Create an auto triggered node**
  i. Create an auto triggered node

Find a type of node and create a node. You can configure dependencies between nodes in a workflow by using a directed acyclic graph (DAG) and drag components to the canvas to orchestrate the workflow. You can configure node dependencies across workflows and workspaces by using the automatic parsing feature.



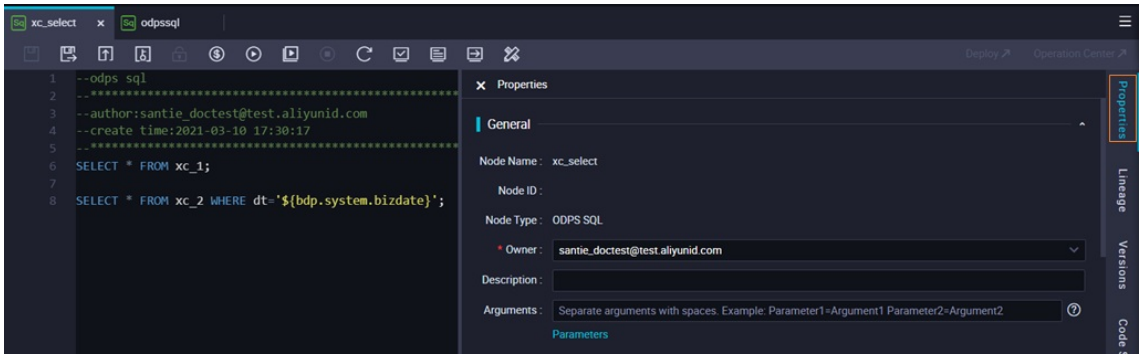
Double click the workflow to enter the editing panel of the workflow. In the panel, you can create the required node by dragging.

The following table describes the additional features that can be used to improve the efficiency of data development.

Additional feature	Description
Create and reference a node group	Groups several nodes that are frequently reused in a workflow as a node group and references the node group in other workflows.
Workflow parameters	Assigns a value to a variable or replaces the value of a parameter for all nodes in a workflow.
Change History	Displays the operation records on a workflow.
Versions	An updated version is generated each time you commit the same workflow. On the Versions tab, you can view and compare the committed versions.
Code search	Allows you to search for a node by entering code snippet keywords and displays all the nodes that contain the code snippet and details about the code snippet. You can use this feature to search for the node that causes data changes in the destination table.

ii. Configure scheduling parameters for the auto triggered node

In DataWorks, you can schedule an auto triggered node in different time granularities, such as minute, hour, day, month, or year. DataWorks allows you to run tens of millions of auto triggered nodes on a daily basis and supports parameter passing between nodes. For more information, see Basic properties.



The following tables describe major scheduling parameters.

> **Note** On the Batch Operation-Data development tab, you can perform an operation on multiple nodes, resources, or functions at the same time. For more information, see the "Batch operations" section.

- Configure basic properties

Parameter	Description
Parameters	Assign values to variables in code. You can add scheduling parameters in the Parameters section. This way, you can dynamically assign values to variables. The values of the added parameters are replaced with the time points at which nodes are scheduled. For more information, see Overview of scheduling parameters.

- Configure time properties

Parameter	Description
Instance Generation Mode	Specifies the time at which an auto triggered node is scheduled. You can view the auto triggered instances of the node on the Cycle Instance page. ■ Next Day: On the next day after an auto triggered node is deployed to the production environment, the auto triggered instances of the node are generated and are run as scheduled. ■ Immediately After Deployment: Auto triggered instances are generated immediately after an auto triggered node is deployed to the production environment. You must set the node scheduled time based on the specified requirement. For more information, see Configure immediate instance generation for a node.

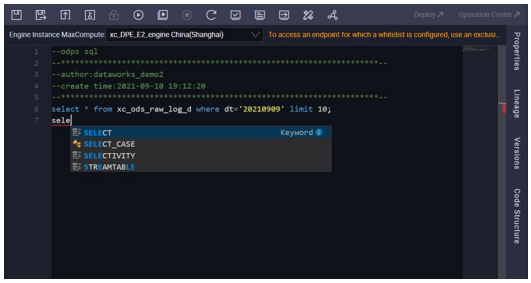
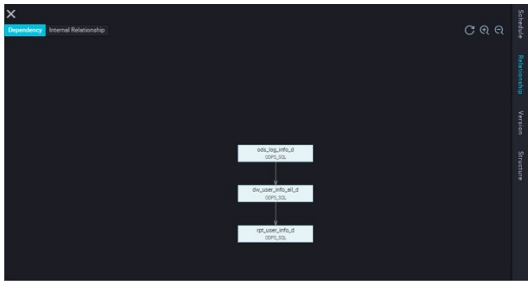
Parameter	Description
Recurrence	Specifies whether an auto triggered node is actually run in a specific scheduling scenario and the impact of each scheduling type on descendant nodes. ■ Normal ■ Description: An auto triggered node is scheduled normally. The descendant nodes of this node are also scheduled normally. ■ Scenario: By default, the Recurrence parameter is set to Normal. ■ Skip Execution ■ Description: An auto triggered node is frozen. However, the instances of this node are not frozen. This auto triggered node cannot be run and the descendant nodes are blocked from running. ■ Scenario: If you do not need to run a workflow within a specified period of time, you can set the Recurrence parameter to Skip Execution. This allows you to freeze the root node of the workflow. ■ Dry Run ■ Description: The scheduling system does not run a node or generate running logs. Instead, the system directly returns a success message for the node and the node does not consume resources. The descendant nodes of the node can be run as scheduled. ■ Scenario: If you do not need to run a node within a specified period of time but you need to run the descendant nodes as scheduled, you can set the Recurrence parameter to Dry Run for this node.
Rerun	Specifies whether to rerun a node based on data idempotence. ■ Allow Regardless of Running Status ■ Allow upon Failure Only ■ Disallow Regardless of Running Status
Auto Rerun upon Error	Specifies the number of automatic reruns upon an error and rerun interval for a node in a specific scheduling scenario.
Validity Period	Specifies the time period within which a node is automatically rerun. The node is not automatically scheduled outside the specified time period and no auto triggered instances are generated.
Scheduling Cycle and Run At	Valid values: **Minute**, **Hour**, **Day**, **Week**, **Month**, and **Year**. **Note** The system generates dry-run instances for an auto triggered node outside the specified time period.
Timeout definition	Specifies the timeout duration. If a node is run for a period of time that exceeds the specified duration, the node automatically terminates and exits.

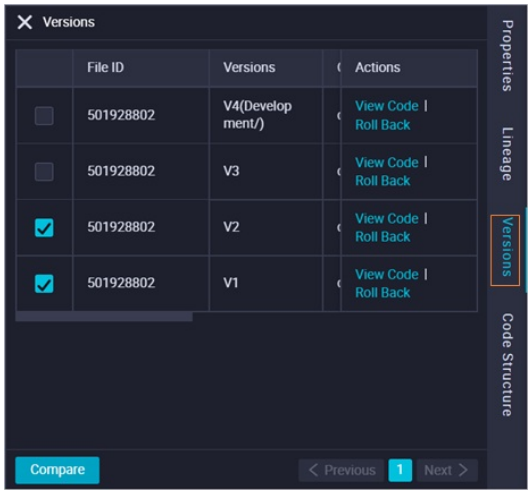
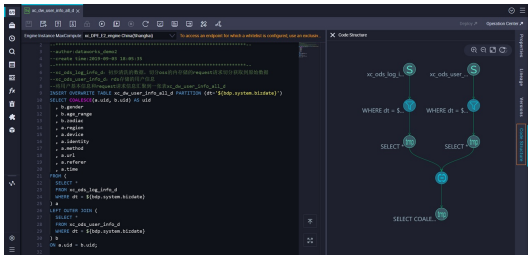
- Configure a resource group: specifies the resource group for node scheduling.
- Configure same-cycle scheduling dependencies

Parameter	Description
Same Cycle	Specifies the nodes that trigger the current node to run. The node depends on the same-cycle auto triggered instances that are scheduled to run on the current day for the ancestor nodes of this node. From the business perspective, the node is run based on the table data that is generated by its ancestor nodes on the current day.
Previous Cycle	Specifies the nodes that trigger the current node to run. The node depends on the previous-cycle auto triggered instances that are scheduled to run on the previous day for the ancestor nodes of the current node. From the business perspective, the node depends on the table data that is generated by its ancestor nodes on the previous day.

- Configure input and output parameters: This feature is used together with an assignment node. You can configure Input Parameters and Output Parameters in the Parameters section to pass the result set of the assignment node to its descendant nodes.

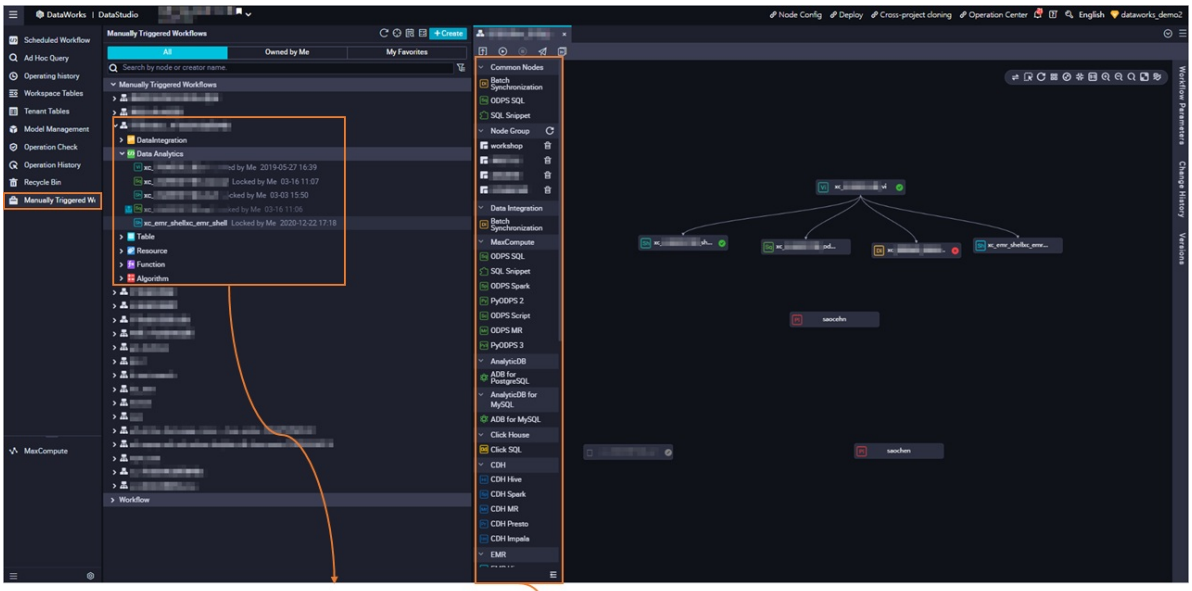
iii. View node code and manage node versions

Feature	Description	Screenshot
Code editing	The AI-powered SQL editor supports code hinting.	
Lineage	Displays node dependencies and node code lineage.	

Feature	Description	Screenshot
Versions	A new version is generated each time you commit the same node. On the Versions tab, you can compare and roll back the committed versions.	
Code Structure	Uses SQL operators to display the code structure.	

- **Create a manually triggered node**

In the left-side navigation pane, click Manually Triggered Workflows. On the page that appears, double-click Manually Triggered Workflows to create a workflow. Find the type of the node that you want to create and create a node in the workflow. You can configure dependencies between nodes in the workflow by using DAGs. You can drag components to the canvas and draw lines to connect the nodes in the workflow.



In the manually triggered workflows, right-click to create a new node and create a manually triggered node task.

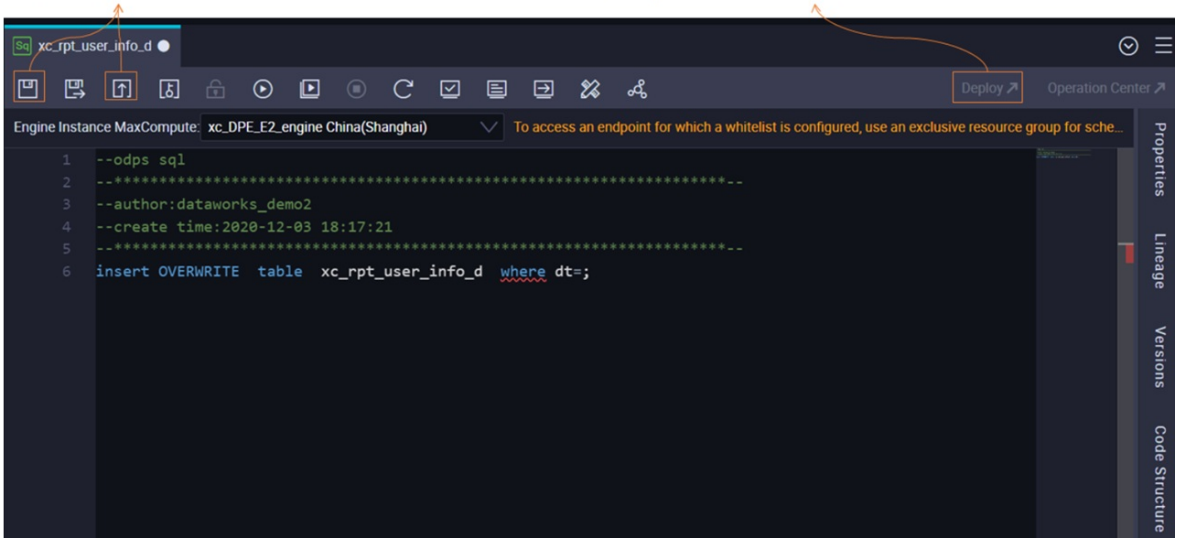
In the manually triggered workflows, double-click the manually triggered workflows to enter the workflows panel, and use drag and drop to create a manually triggered node task.

- **Commit a node**

  After you commit a node, the operation record is displayed on the Create Package page. You can determine whether to deploy the operation in the Nodes to Deploy panel. The node can be scheduled in the production environment only after you deploy the operation.

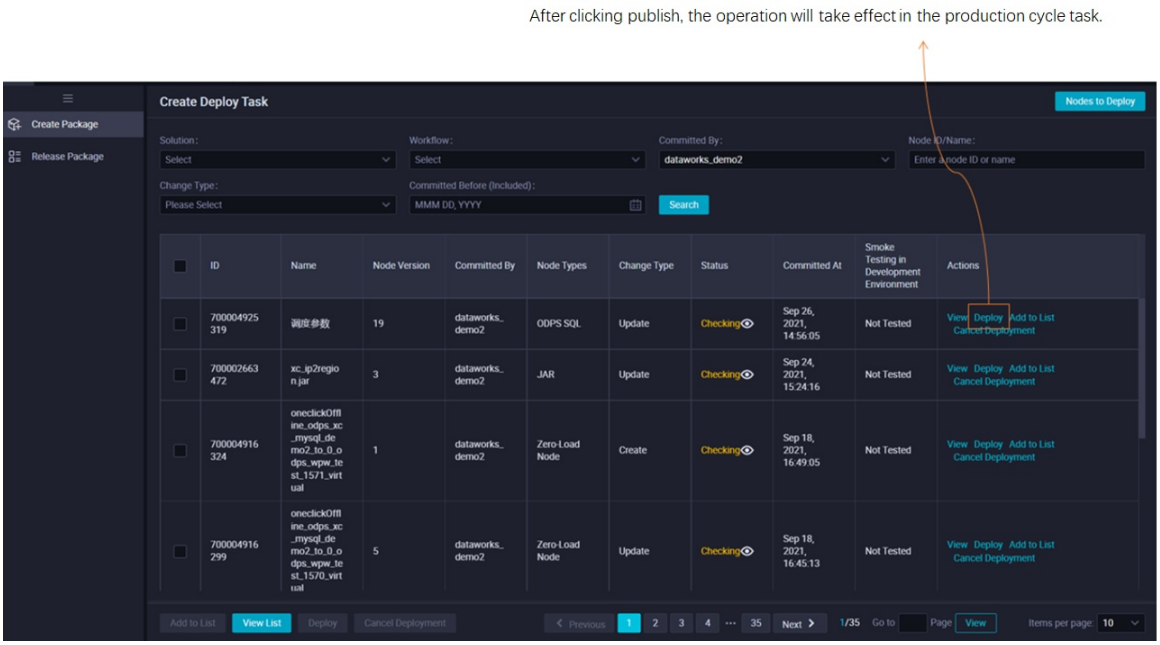
After saving and submitting the node, you will enter the task publishing interface.

You can click publish to enter the task publishing interface and quickly locate the records to be published on this node.



- **Deploy a node**

The Create Package page displays all the operations to be deployed in a workspace, including the records of add, update, and undeploy operations. A node can be scheduled as expected in the production environment only after you deploy the operation in the Nodes to Deploy panel.



> **Note** In a workspace in simple mode, you can use the cross-project cloning feature to deploy code in the workspace to another workspace.

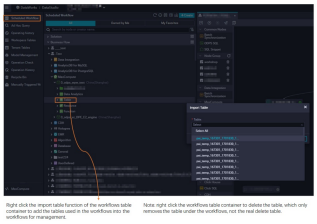
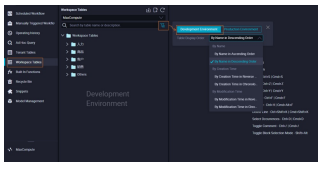
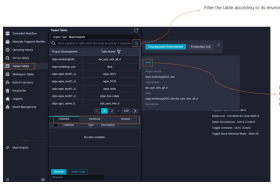
Manage and use tables, resources, and functions

DataWorks encapsulates tables, resources, and functions of an engine. You can create tables and resources, and register functions in a visualized manner.

- **Manage tables**

  You can manage tables in a visualized manner, upload data of a table from your on-premises machine, and export table data.

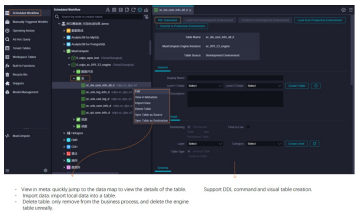
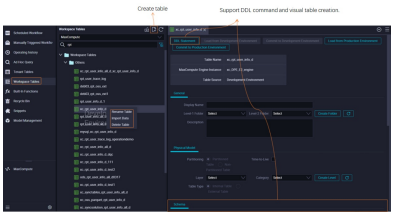
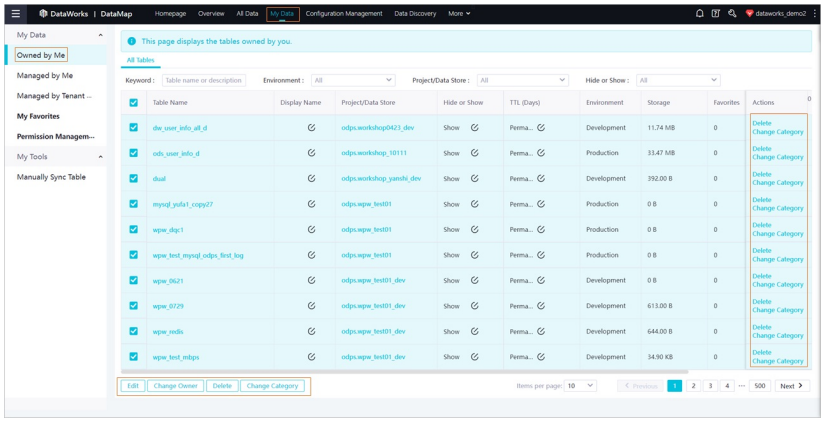
  DataWorks provides multiple entries to help you manage and use tables. You can select an entry based on your business requirements.

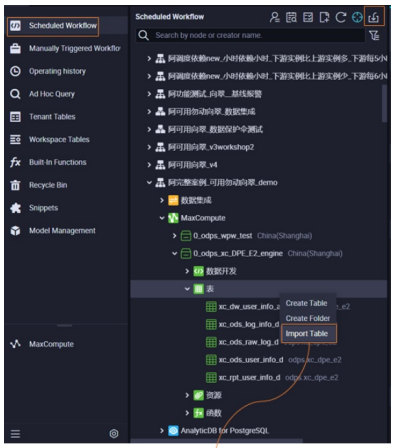
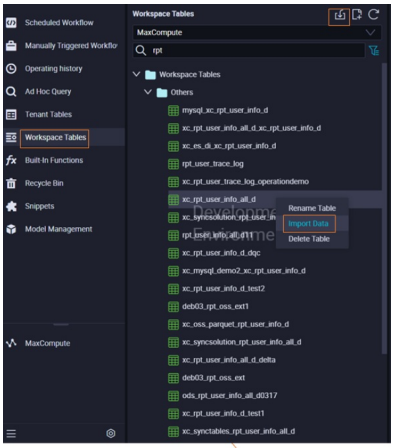
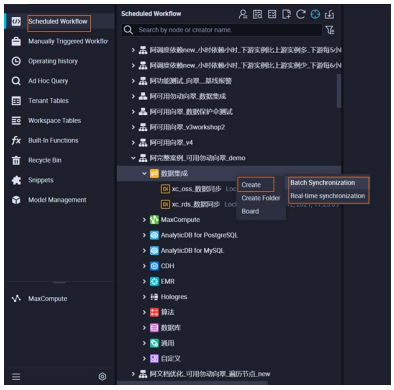
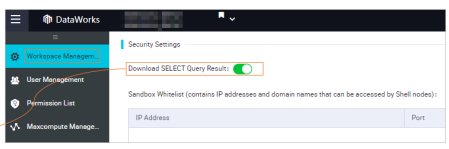
Entry	Scheduled Workflow	Workspace Tables	Tenant Tables
Entry Screen shot			

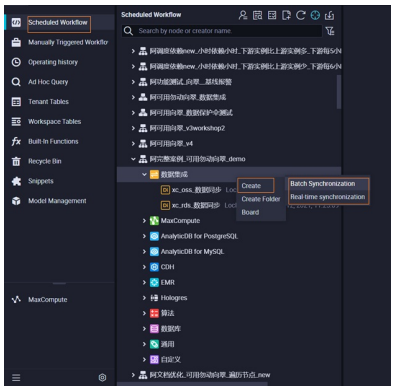
Entry	Scheduled Workflow	Workspace Tables	Tenant Tables
Operation description	Use workflows on the Scheduled Workflow page to manage tables based on business workflows. ◦ Import tables to a workflow: Use this feature to import tables that are required for a workflow to the workflow. This way, you can manage the tables based on your business requirements. ◦ Delete tables from a workflow: Use this feature to delete tables that are no longer required for a workflow. You can right-click the table that you want to delete and click Delete Table. Note The table is deleted only from the workflow. It is still retained in the engine.	◦ Manage tables: You can view all the tables in the development and production environments in a workspace. ◦ Manage settings for tables: You can create folders and manage tables by folder.	None.

Note You can view the basic metadata, lineage, and impact of a table in Data Map. For more information, see Overview.

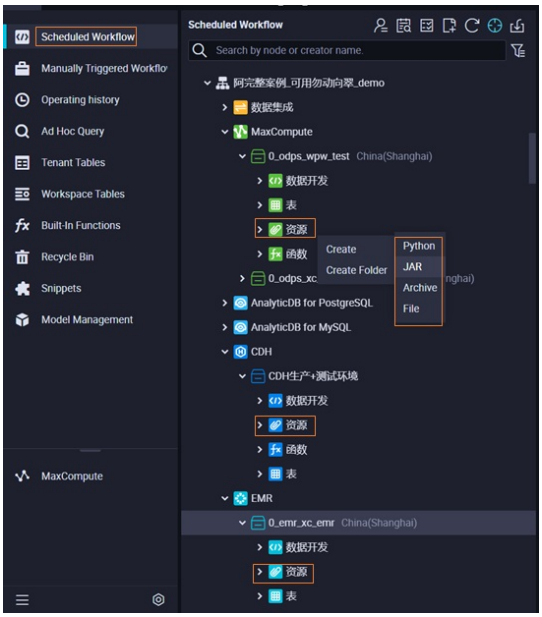
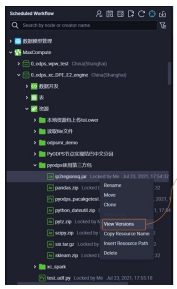
The following table describes the items that you must take note of when you manage tables by using different entries.

Operation category	Operation	Scheduled Workflow	Workspace Tables
Manage tables	Basic table operations: ○ Create a table ○ Delete a table in the development environment ○ Rename a table ○ Modify the comment of a table ○ Add a field ○ ...	The table operations are the same as the operations on an engine. 	The table operations are the same as the operations on an engine. 
	○ Delete a table in the production environment ○ Change multiple table owners at the same time ○ Change multiple table lifecycles at the same time ○ Change the display names of multiple tables at the same time	You cannot directly perform the four operations on a table in the production environment on the Scheduled Workflow or Workspace Tables page. You must perform the operations in Data Map. For more information, see Overview. 	

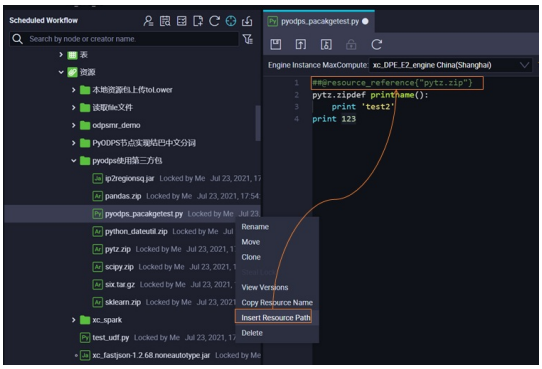
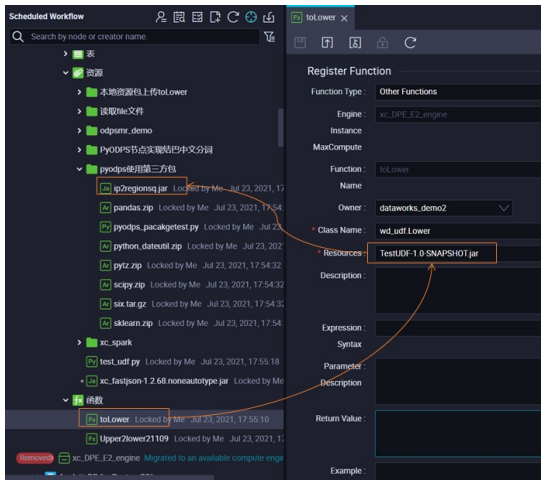
Operation category	Operation	Scheduled Workflow	Workspace Tables
Import table data	Upload data from your on-premises machine to a table	 - Develop specific tables under the workflow table container based on data and upload data. - Provide a quick entry to find the target table and upload the table data through the search table.	 - Upload table data based on table management. - Provide a quick entry to find the target table and upload the table data through the search table.
	Synchronize data from other data sources to a table	 If you want to synchronize data from other data sources to a table, you can use batch synchronization nodes and real-time synchronization nodes.	Not supported.
	Export data to your on-premises machine	  **Note** An administrator can specify the Download SELECT Query Result parameter.	

Operation category	Operation	Scheduled Workflow	Workspace Tables
Export table data	Export data to other data sources	 If you want to synchronize data to other data sources, you can use batch synchronization nodes and real-time synchronization nodes.	Not supported.

- **Manage and use resources**
  - Manage resources

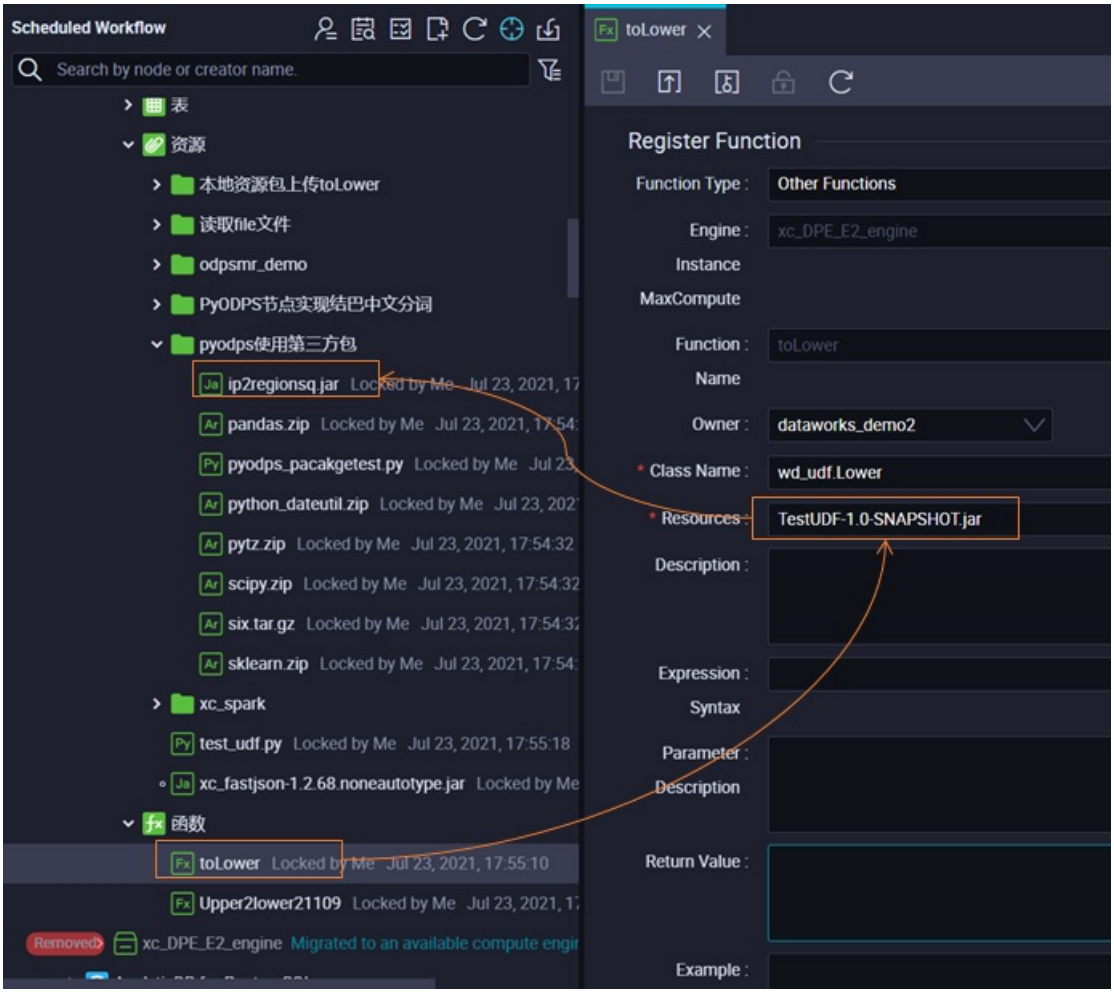
Visualized upload	Resource version management
	 

- Use resources

Use resources in a node	Use resources to register functions
	

- **Manage and use functions**

  You can register a function on the Scheduled Workflow page.



Manage the permissions and development behavior of members

The permissions in DataStudio are classified into the permissions on engines and the permissions on service modules.

- Manage the permissions on engines:

  After you associate an engine with a workspace, you can manage the permissions on the engine when you develop data in DataWorks.

  > **Notice** You are automatically granted the permissions on the engine with which the workspace is associated. You may encounter the following scenarios when you manage the permissions on engines:
  >
  > - If you associate an engine with a workspace, a built-in role is granted the permissions on tables, functions, and resources of some engines.
  > - If no built-in role is assigned or a built-in role is not granted the required permissions on a specific engine, you must go to the permission granting page of the specified engine to grant the required permissions. You are not allowed to directly grant the permissions in DataWorks.

- Manage permissions on service modules:

  You can manage the permissions on service modules of DataWorks when you perform data development operations that are not related to engines.

Development behavior management: DataWorks provides a permission management capability that can help you identify and block sensitive behavior at the earliest opportunity. If sensitive behavior is detected, you can manually block the behavior or DataWorks can automatically notify you of the sensitive behavior based on the custom event check logic. However, DataWorks does not block operation procedures.

- **Manage permissions on MaxCompute**

  If the MaxCompute engine is associated with a workspace in standard mode:

  - The DataWorks built-in roles and the roles in a MaxCompute project in the **development environment** have a permission mapping. By default, a DataWorks built-in role has all the permissions that the mapped MaxCompute project role has on the MaxCompute engine in the development environment.
  - The DataWorks built-in roles and the roles in a MaxCompute project in the **production environment** do not have a permission mapping. A DataWorks built-in role cannot directly manage resources of a MaxCompute project in the production environment.

  In summary, after a RAM user of DataWorks is assigned the administrator or developer role, the RAM user has all the permissions on a MaxCompute project in the development environment. However, the RAM user does not have the permissions on the same MaxCompute project in the production environment. If you want to use the RAM user to access a table in the production environment from the development environment, you must apply for the operation permissions on the table for the RAM user in Data Map. For more information, see Data Map.

  You can compile and debug code on the **DataStudio** page, commit and deploy the code to the production environment, and then run nodes in the production environment on the **Operation Center** page.

Operation page	Access a table in the development environment	Access a table in the production environment
DataStudio	◦ Sample code: `select col1 from tablename` ◦ Operation result description: You can use an Alibaba Cloud account or a RAM user to access a table in the development environment. Specify the table name in the **projectname_dev**.`tablename` format.	◦ Sample code: `select col1 from projectname.tablename` ◦ Operation description: You can use an Alibaba Cloud account or a RAM user to access a table in the production environment. Specify the table name in the **projectname**.`tablename` format.
Operation Center	Not supported.	◦ Sample code: `select col1 from tablename` ◦ Operation description: You can use an Alibaba Cloud account or a RAM user that you select when you associate an engine instance with a workspace to access a table in the production environment. Specify the table name in the **projectname**.`tablename` format.

- **Manage permissions on E-MapReduce (EMR)**

  DataWorks built-in roles and EMR roles do not have a permission mapping. When you associate an EMR cluster with a workspace, you can set Access Mode to **Shortcut Mode** or **Security Mode**. The association settings that you must configure and the operation permissions that you can have vary based on the access mode that you select. For more information, see Associate an EMR cluster with a DataWorks workspace.

  - **Shortcut Mode**

Operation page	Access a table in the development environment	Access a table in the production environment
DataStudio and Operation Center	Perform operations as the hadoop user.	

- **Security Mode**

Operation page	Access a table in the development environment	Access a table in the production environment	How it works
DataStudio	You can use the account that you selected in the **Development Environment** section when you associate the EMR cluster with the workspace to access all engine resources.	Not supported.	You can configure the Lightweight Directory Access Protocol (LDAP) permission mapping for members in a DataWorks workspace to manage the permissions of a RAM user on EMR clusters when the RAM user uses DataWorks. If you use an Alibaba Cloud account or a RAM user to commit code in DataWorks, the account that has the same username in EMR is used to run the node. You can use EMR Ranger to manage the permissions of users in an EMR cluster. This way, Alibaba Cloud accounts, node owners, and RAM users have different data permissions when they run EMR nodes in DataWorks.
Operation Center	Not supported.	You can use the account that you selected in the **Production Environment** section when you associate the EMR cluster with the workspace to access all engine resources.	

- **Manage permissions on other engines**

  If engines except MaxCompute and EMR are associated with a workspace, whether you have permissions to run nodes on the DataStudio page depends on the account that you selected when you configure the engine.

- **Manage permissions on service modules**

- Manage permissions on service modules

  DataWorks allows you to create a custom role, and then grant the read/write permissions on specific service modules to this role. For more information, see Manage workspace-level roles and members.

- Manage the permissions on features of a service module

  In DataWorks, if some features are dimmed or the entry of a feature is not found, check whether you have the required permissions. DataWorks built-in roles have different permissions. For more information, see Permissions of built-in workspace-level roles.

- **Manage permissions on operation procedures**

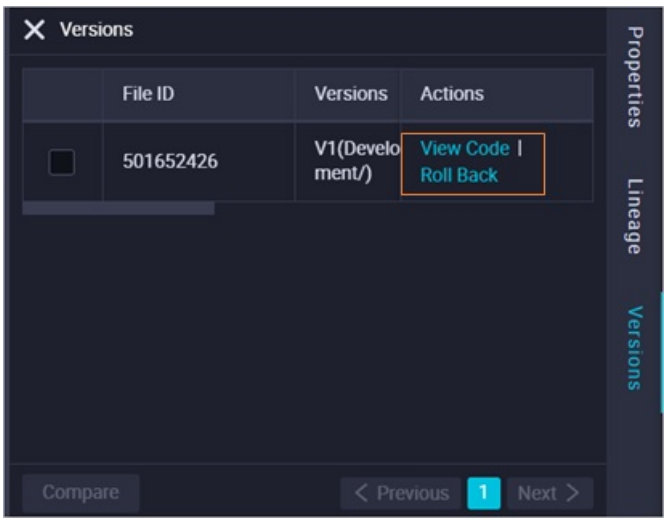
  DataWorks provides a permission management capability that can detect and block sensitive behavior at the earliest opportunity. If sensitive behavior is detected, you can manually block the behavior or DataWorks can automatically notify you of the sensitive behavior based on the custom event check logic. However, DataWorks does not block operation procedures.

  DataWorks provides the code review feature. This feature includes the forcible code review (block operation procedures) and message sending (notification without blocking operation procedures) sub-features. You can enable Force to review code for baseline tasks of the specified priority based on node importance (the baseline to which nodes belong). You can open data by opening relevant interfaces or features. This way, you can identify the core changes and take relevant measures at the earliest opportunity. You can enable the message opening feature to subscribe to the status changes of database tables and nodes in DataWorks and achieve personalized and automated responses.

ActionTrail

- Restore nodes and table data
  - Restore a node: You can restore nodes that are recently deleted. For more information, see Recycle bin. New node IDs are generated after you restore deleted nodes.
  - Restore data of MaxCompute tables: DataWorks provides the data backup and restoration feature. The system automatically backs up data of earlier versions, including deleted data or data that exists before an update, and retains the data for a period of time. For more information, see Backup and restoration.
- Compare and roll back node versions

On the DataStudio page, find the node whose versions you want to view and click Versions on the right side of the node configuration page. On the Versions tab, you can select the nodes and compare the node versions. You can also roll back node versions. For more information, see Versions.



- Obtain audit logs of operations that are performed in the DataWorks console, such as download data

  DataWorks is integrated with ActionTrail. You can view and retrieve DataWorks behavioral events of your Alibaba Cloud account over the previous 90 days in the ActionTrail console. You can use ActionTrail to deliver the event logs to a Logstore in Log Service or an Object Storage Service (OSS) bucket for monitoring and alerting. This way, you can audit the event logs and track and analyze issues at the earliest opportunity. For more information, see Use ActionTrail to query behavior events.

- Mask data and trace leaked data

  If your files are important, you can configure data masking rules for sensitive data to prevent file data leak and trace the leaked data by using the data watermark feature in Data Security Guard. For more information, see Create a data masking rule.

- Audit the permissions on MaxCompute tables

  Go to the Security Center page and click Data access control. On the Data access control page, click **Permission audit**. On the tab that appears, you can view the owner IDs that have permissions on tables, the details about the permissions, and the validity period of the permissions. You can also revoke the permissions on tables.

2.Features on the DataStudio page

This topic describes the features that are provided by DataWorks on the DataStudio page. This helps you understand the overall layout of and modules on the DataStudio page and easily access relevant topics.

Go to the DataStudio page

1. Log on to the DataWorks console, click **Workspaces** in the left-side navigation pane, and then select a region.
2. Find the workspace that you want to manage and click **DataStudio** in the Actions column to go to the DataStudio page of the workspace.

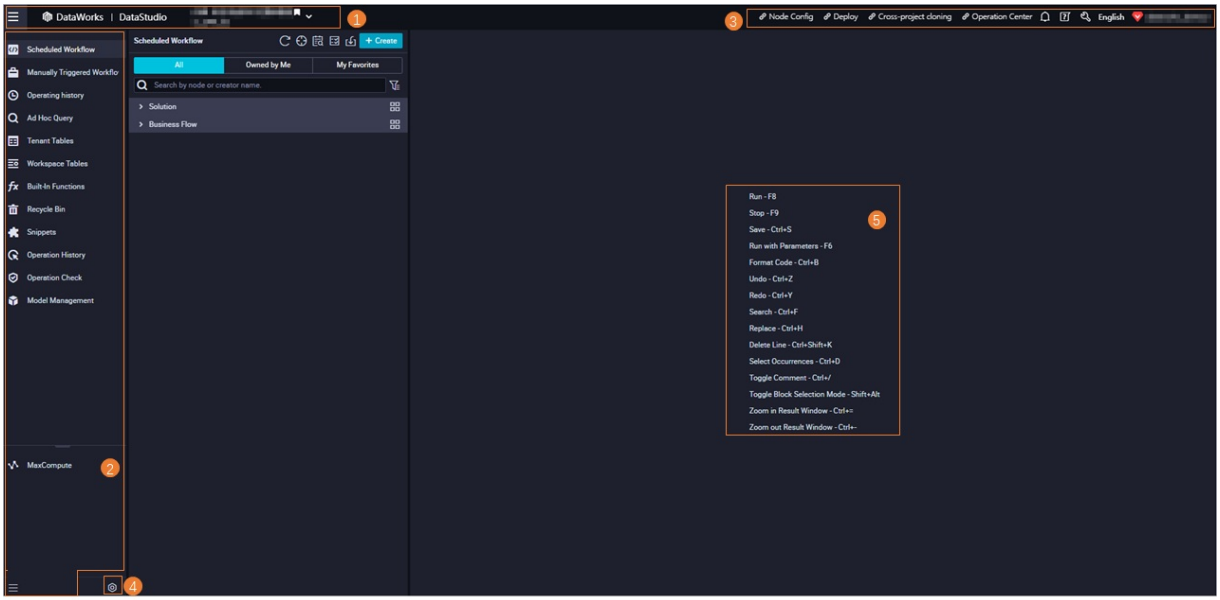
On the DataStudio page, you can create workflows and different types of nodes for data development based on your business requirements. For more information, see Manage workflows and Create a batch sync node.

The features for different data development operations vary. The following sections in this topic describe these features to facilitate your understanding:

- Overall layout of the DataStudio page: describes the overall layout of the DataStudio page.
- Features related to workflows: describes the features related to workflows on the DataStudio page.
- Shortcut menu related to workflows: describes the shortcut menu related to workflows on the DataStudio page.
- Features related to nodes: describes the features related to nodes on the DataStudio page.
- Shortcut menu related to nodes: describes the shortcut menu related to nodes on the DataStudio page.

Overall layout of the DataStudio page

The following figure shows the overall layout of the DataStudio page.



Section	Description
1	• Switch between workspaces. This section displays the name of the current workspace. You can click the icon to switch to another workspace in the current region. • Go to another DataWorks service. You can click the icon to go to another DataWorks service, such as Data Integration or Operation Center. ○ **Data Integration**: Data Integration. ○ **Data Modeling**: Data Warehouse Planning, Data Standard, Data Metric, and Dimensional Modeling. ○ **Data Development And Task Operation**: DataStudio, Operation Center, Deploy, and Code Review. ○ **Data governance**: DataMap, Data Quality, Security Center, Data Security Guard, and Data Governance Center. ○ **Data Analysis**: DataAnalysis. ○ **Data Service**: DataService Studio. ○ **Machine Learning**: PAI. ○ **Other**: Approval Center, Resource Optimization, **DataWorks(Home)**, Migration Assistant, and Global Member Management. • Go back to the DataWorks homepage. Click the icon. On the page that appears, click the Back to Console icon in the lower-left corner to go back to the DataWorks homepage.

Section	Description

2	In this section, you can click the icon to show or hide the names of the module tabs in the left-side navigation pane.
	• Scheduled Workflow: On this tab, you can create auto triggered nodes that use different compute engines for data development. The nodes created on this tab can be deployed to the production environment for O&M.
	Note Before you can use a specific compute engine for data development, you must associate your workspace with the compute engine.
	• Manually Triggered Workflows: On this tab, you can develop manually triggered nodes. The nodes created on this tab can be deployed to the production environment for O&M.
	• Operating history: On this tab, you can view the records of the nodes that are run within the previous three days in DataStudio.
	• Ad Hoc Query: On this tab, you can perform a simple ad hoc query to test your code. However, the code of an ad hoc query cannot be deployed to the production environment for O&M.
	• Tenant Tables: On this tab, you can view all production tables of the current Alibaba Cloud account.
	• Workspace Tables: On this tab, you can perform operations on a table in a visualized manner. The operations that you can perform on a table must be supported by the compute engine used to create the table.
	• Built-in Functions: On this tab, you can view the descriptions of all built-in MaxCompute functions.
	• Recycle Bin: On this tab, you can manage the nodes, resources, and functions that are removed from the **Scheduled Workflow** or **Manually Triggered Workflows** tab.
	• Snippets: A script template is a pre-defined block of code that involves multiple input and output parameters. Each SQL code block references one or more source tables. You can filter source table data, join source tables, and aggregate them to generate a table required by the new business.
	• **Operation History**: On this tab, you can filter and view historical operation records in the current workspace by operation type, operator, and operation time.
	• **Operation Check**: On this tab, you can filter and view operations by operation type and check status.
	• **MaxCompute**:
	  ◦ MaxCompute Resources: On this tab, you can manage the existing MaxCompute resources and view the operation records of a specific MaxCompute resource. In addition, you can add a MaxCompute resource that is not uploaded in DataWorks to the Scheduled Workflow tab for management.
	  ◦ MaxCompute Functions: On this tab, you can manage the existing MaxCompute functions and view the operation records of a specific MaxCompute function. In addition, you can add a MaxCompute function that is not registered with DataWorks to the Scheduled Workflow tab for management.
	Note If a specific module is not displayed in the left-side navigation pane, you can click the icon in Section 4 to add the module on the **Settings** page. For more information, see Personal settings.

Section	Description
3	DataStudio shortcuts to other services: • Node Config: You can click Node Config to manage custom nodes and wrappers to meet your requirements for data development and data quality. After you create a custom node, you can write an SQL statement for the node in DataStudio. DataWorks parses and executes the SQL statement based on the processing logic of the wrapper that you define for the node in the background. Before you create a custom node, you must create a wrapper to define the processing logic of the node. • Deploy: You can click Deploy to deploy a node that is developed in DataStudio to the production environment. You can manage the deployment process of the node. • Cross-project cloning: You can click Cross-project cloning to clone and migrate nodes such as compute nodes and synchronization nodes between workspaces. • Operation Center: You can click Operation Center to perform O&M operations on nodes in Operation Center. In Operation Center, you can switch between the development environment and the production environment. You can perform O&M operations on deployed nodes in Operation Center in the production environment. Common features of DataWorks services: Note DataWorks services share common features. The following content describes the common features that are provided by DataWorks on the DataStudio page. • **Notification Center ( )**: You can click this icon to obtain the latest updates of DataWorks at the earliest opportunity. • **Helps ( )**: You can click this icon to obtain information about how to use a specific feature based on your business requirements. • **Workspace Manage ( )**: You can click this icon to go to the Workspace Management page. On this page, you can view the basic information, scheduling properties, security settings, and associated compute engines of the workspace. For more information, see Configure a workspace. • **Language switch**: You can click the current language and switch to another language. For example, you can switch from Chinese to English. • **Account information**: You can click the account to view the personal information of the account and the status statistics about nodes in the workbench.

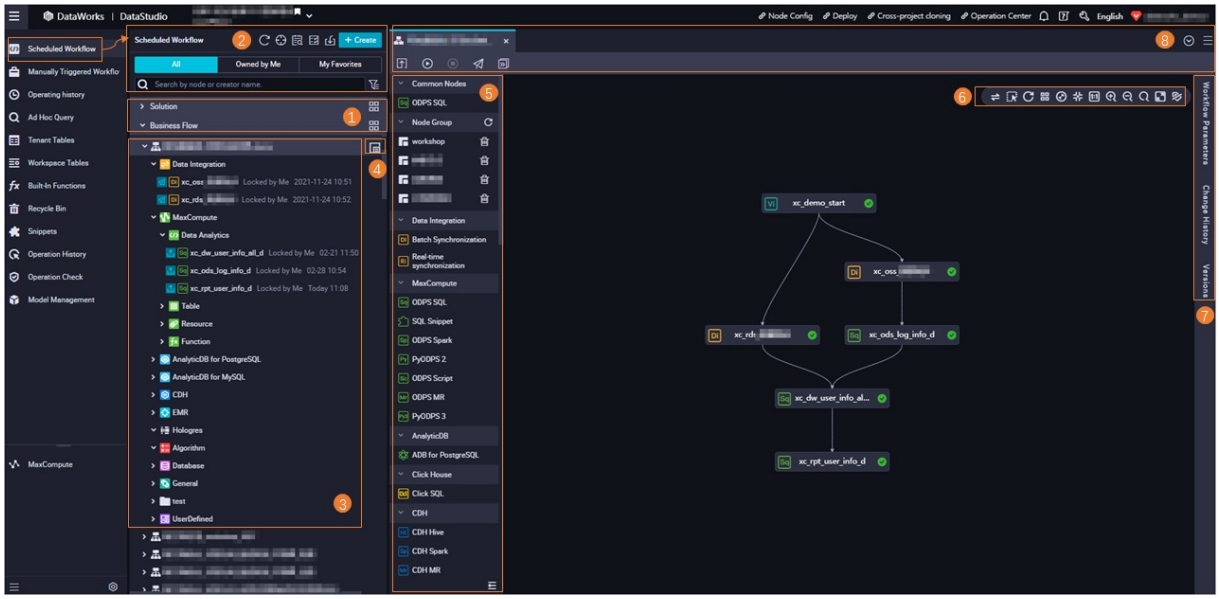
Section	Description

4	After you click the Settings icon in Section 4, you can set system configurations on the following tabs of the Settings page:
	• Personal Settings: On this tab, you can manage DataStudio modules, editor settings, and general settings, such as the DataWorks theme.
	• Code Templates: On this tab, you can modify a code template to a required style.
	• Scheduling Settings: On this tab, you can enable the periodic scheduling feature and configure the default scheduling settings for auto triggered nodes. Auto triggered nodes can be run as scheduled only after the periodic scheduling feature is enabled.
	• Table Management: On this tab, you can manage settings such as partition formats, identifiers of partition fields, prefixes of table names, table folders, and table levels.
	• Workspace Backup and Restoration: On this tab, you can compress and download code to back up the code of a workspace. You can also upload the code package and use the package to restore the code that is accidentally deleted.
	• **Security Settings and Others**:
	  ○ Security settings: You can specify whether to mask sensitive information in the returned results of queries that you perform in DataStudio in the current workspace.
	  ○ Other settings: You can enable forcible code review and specify one or more code reviewers to manage code quality of your nodes.

5	This section displays the keyboard shortcuts that are commonly used in the DataStudio editor. For more information about the keyboard shortcuts, see Editor shortcuts.

Features related to workflows

By default, the Scheduled Workflow tab appears if you go to the DataStudio page. On the Scheduled Workflow tab, you must create a workflow before you can organize your data development operations. For more information about how to create a workflow, see the "Create a workflow" section in the Manage workflows topic. The following figure shows the features related to workflows.



Section	Description
1	• Solution: You can create a solution to manage multiple workflows. A workflow can be added to different solutions. Solutions can be displayed by using lists and cards on a GUI. • Business Flow: A workflow is an abstract business entity. You can create a workflow to organize code development operations based on the business requirements. Click the icon to show all solutions or workflows in the current workspace.

2	• **Refresh ( )**: After you modify a workflow or solution, you can click this icon to refresh the corresponding directory tree.
	• **Locate ( )**: You can click this icon to find the current node on the Scheduled Workflow tab.
	• Search Code ( ): You can click this icon to search for a code snippet by using keywords. This way, you can find all nodes that contain the code snippet on the Scheduled Workflow, Manually Triggered Workflows, Ad Hoc Query, and Recycle Bin tabs and view the details of the code snippet in a centralized manner. You can also use this feature to identify the node that causes changes to a table.
	• Batch Operation ( ): You can click this icon to modify the configurations of multiple tables, resources, and functions at a time. The configurations include the owner, engine instance, resource group for scheduling, rerun properties, scheduling type, recurrence, and scheduling timeout period.
	• Import ( ): You can click this icon to upload the data of a local file to a table in DataWorks. Take note that you can import the data of a local file only to a MaxCompute table.
	• **Create** (+ Create): You can click Create to quickly create workflows, nodes, tables, resources, and functions.
	• **Solution and workflow directory trees**:
	  ◦ **All**: This directory tree displays all created objects, including nodes, resources, and functions, in the current workspace by solution and workflow.
	  ◦ **Owned by Me**: This directory tree displays the objects, including nodes, resources, and functions, that are owned by the current account by solution and workflow.
	  ◦ **My Favorites**: This directory tree displays the objects, including nodes, resources, and functions, that are added to favorites by the current account by solution and workflow.
	• **Node search**:
	  ◦ **Exact search**: You can enter the name of a node or the identifier of a node creator in the search box and click the icon to search for the specified node.

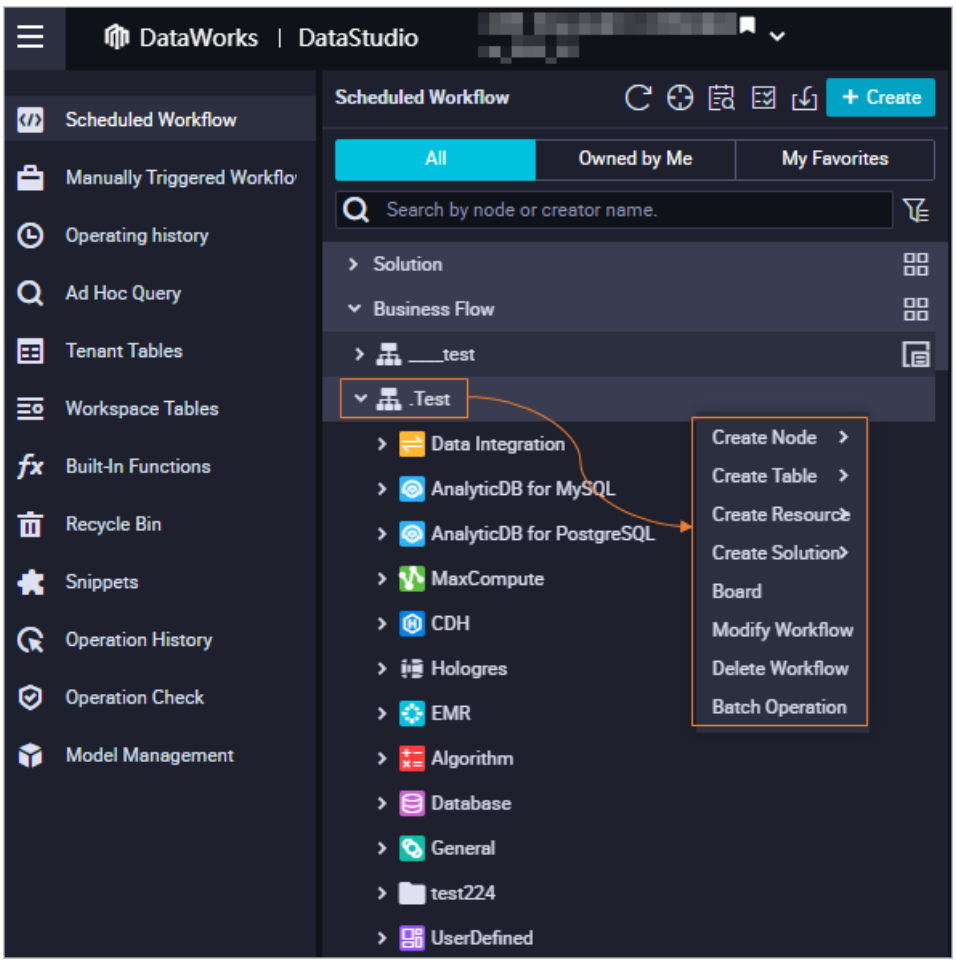
Section	Description
	◦ **Search by node type**: You can click the icon to specify the types of nodes that you want to search. After you specify a node type, the directory tree displays only nodes of the specified type in the current workspace. **Note** You can determine whether to hide engine instances or node folders based on your business requirements. After you select **Hide Engine Instances** or **Hide Node Folders**, engine instances or node folders are not displayed in the directory tree. ■ **Hide Engine Instances** and **Hide Node Folders** are applicable only to the latest version of workflows. ■ Generally, if an engine contains only one engine instance, we recommend that you hide the engine instance. ■ If you do not need to use node folders, such as Data Analytics, Table, Resource, and Function, you can hide them.
3	In this section, you can use a directory tree to manage the nodes, tables, resources, and functions in each workflow. • **Workflow**: the unit for business development. **Note** Before you perform data development operations in a new workspace, you must create a workflow and a node in the workflow. For more information about how to create a workflow, see the "Create a workflow" section in the Manage workflows topic. • **Node**: the smallest unit for code development. You can develop code by node type, such as engine nodes, algorithm nodes, Data Integration nodes, database nodes, general nodes, or custom nodes. • **Table**: You can manage tables in DataStudio in a visualized manner. • **Resource**: You can upload resources in DataStudio in a visualized manner. **Note** Only MaxCompute, E-MapReduce (EMR), and Cloudera Distribution Hadoop (CDH) engines support visualized uploading of resources. • **Function**: You can register functions in a visualized manner. **Note** Only MaxCompute, EMR, and CDH engines support visualized registration of functions. The icon before the name of a node indicates the status of the node: • icon: indicates that the node is not committed. You can click this icon to commit the node. • icon: indicates that the node is not deployed. You can click this icon to deploy the node. The last time when the node is edited is displayed after the node name. Double-click the name of a workflow to go to the configuration tab of the workflow, as shown in Sections 5 to 8. On this tab, you can perform data development operations.
4	Resource Group Orchestration ( ): You can click this icon to change the resource groups for scheduling used by multiple nodes in a workflow during data development. If multiple resource groups for scheduling are used in your workspace, you can use this feature to change the resource groups for scheduling for the nodes in the workspace based on your business requirements. This helps you improve resource usage. After you change the resource groups for scheduling used by multiple nodes, you must deploy the nodes to the production environment so that the change can take effect in the production environment.

Section	Description
5	• **Common Nodes**: This section displays the common types of nodes in the current workspace. This helps you quickly select a node type and create a node. • Node Group: You can use this feature to reference a set of nodes across workflows. You can add nodes that are frequently used in a workflow to a node group and reuse the node group in other workflows. • Quick node creation: You can drag nodes in sections, such as Data Integration, MaxCompute, and EMR, to the right-side canvas of a workflow to create the nodes in the workflow.
6	Tools on the canvas: • **Switch Layout** (): You can click this icon to switch the layout of the canvas to **Vertical**, **Horizontal**, or **Grid**. • **Box** (): You can click this icon to select nodes to form a node group and perform operations on the node group to manage selected nodes. • **Refresh** (): After you modify a workflow, you can click this icon to refresh the workflow. • **Format** (): You can click this icon to horizontally align the nodes on the canvas. • **Adapt** (): You can click this icon to adapt the current workflow layout to the size of the canvas. • **Center** (): You can click this icon to center nodes on the canvas. • **1:1** (): You can click this icon to change the scale of the directed acyclic graph (DAG) of nodes to 100%. • **Zoom In** (): You can click this icon to zoom in the nodes in the current workflow. • **Zoom Out** (): You can click this icon to zoom out the nodes in the current workflow. • **Search** (): You can click this icon and enter a keyword in the search box to search for a node whose name contains the keyword. **Note** Fuzzy match is supported. After you enter a keyword, DataWorks displays all nodes whose names contain the keyword in the current workflow. • **Toggle Full Screen View** (): You can click this icon to view the current workflow in full screen. • **Hide Engine Information** (): You can click this icon to show or hide the engine information of each node.

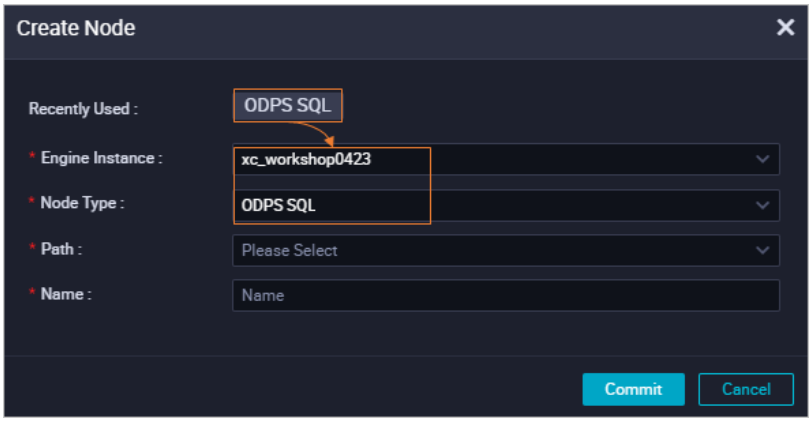
Section	Description
7	Tabs in the right-side navigation pane: • Workflow Parameters: You can click this tab and assign a value to a variable in the code for all **ODPS SQL** nodes in the current workflow at a time. • Change History: You can click this tab and view the operation records of nodes in the current workflow. • Versions: Each time nodes in the workflow are committed, a new version is generated for the workflow. You can click this tab and view all versions and version details of the workflow.
8	Tools in the toolbar and tools above the configuration tab: • **Submit** (): You can click this icon to commit one or more updated nodes in the current workflow to the Deploy page. • **Run** (): You can click this icon to run all nodes in the current workflow. • **Stop** (): If a workflow is running, you can click this icon to stop the nodes from running in the workflow. • Deploy (): You can click this icon to go to the Deploy page and view the nodes to be deployed in the current workflow. Then, you can deploy nodes based on your business requirements. • Go to Operation Center (): You can click this icon to go to Operation Center to view the O&M details of nodes. • **View opened configuration tabs**: If you have opened multiple configuration tabs on the DataStudio page, you can click the icon to view all configuration tabs that are open from the drop-down list. • **Close opened configuration tabs**: You can click the icon to close one or more configuration tabs.

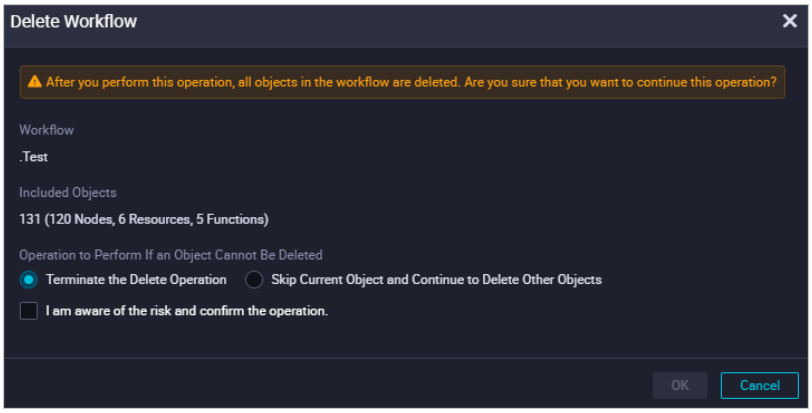
Shortcut menu related to workflows

Move the pointer over a workflow and right-click the workflow. The following figure shows the shortcut menu that appears, and the following table describes the commands supported by the shortcut menu.



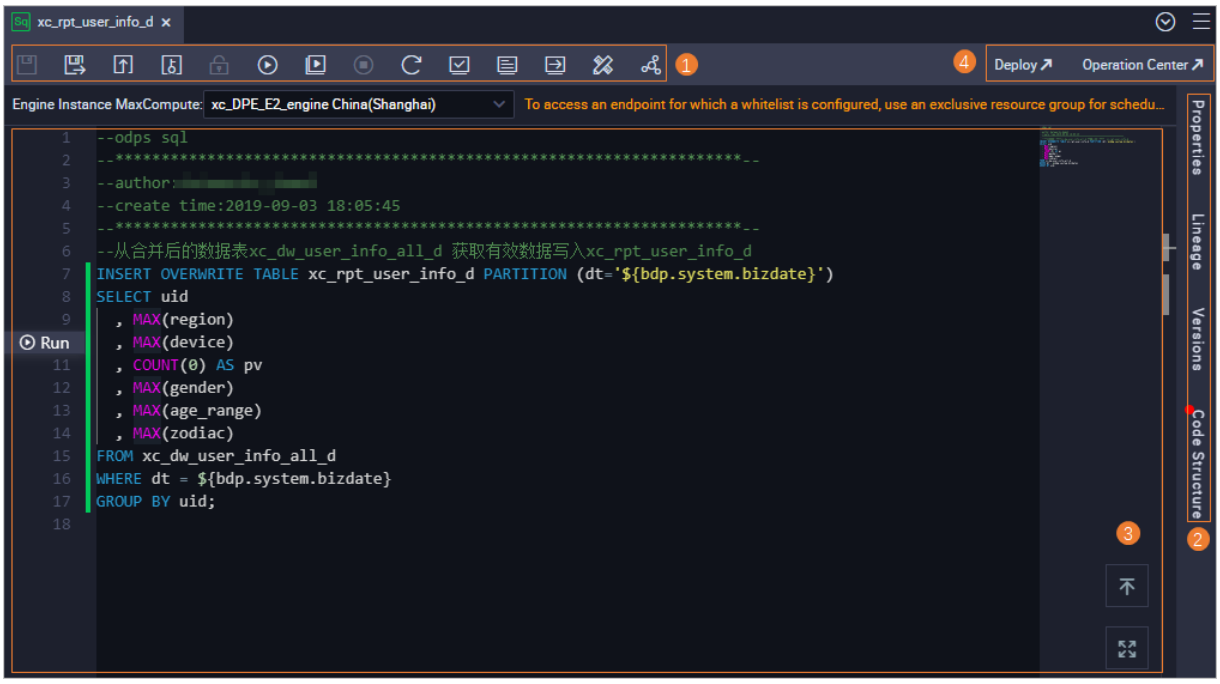
Command	Description

Command	Description
Create Node	This command allows you to quickly create nodes of different types. When you create a node, the system displays the node types that are recently used. If you click one of the node types, the system automatically sets the **Engine Instance** and **Node Type** parameters based on the information about the node that was last used of this type. You can create a node of a type that was recently used by using this method. 
Create Table	This command allows you to quickly create tables of different types.
Create Resource	This command allows you to quickly create resources of different engine types. **Note** This command supports only MaxCompute, CDH, and EMR resources.
Create Solution	This command allows you to quickly create functions of different engine types. **Note** This command supports only MaxCompute, CDH, and EMR functions.
Board	This command navigates you to the canvas of a workflow.
Modify Workflow	This command allows you to modify the name, owner, and description of a workflow.

Command	Description
Delete Workflow	This command allows you to delete the current workflow. **Note** If you perform this operation, all objects in the workflow will be deleted. Proceed with caution. The following options are available to cope with situations where an object cannot be deleted: • **Terminate the Delete Operation**: By default, this option is selected. If an object cannot be deleted, the delete operation will be terminated. This operation does not affect the deleted objects. • **Skip Current Object and Continue to Delete Other objects**: If an object cannot be deleted, the system skips the object and continues to delete other objects. 
Perform operations on multiple DataWorks objects at a time	This command allows you to modify the configurations of multiple nodes, resources, or functions at a time. For example, you can modify the owners, engine instances, and scheduling properties of multiple objects at a time. This command also allows you to commit and deploy multiple modified objects to the production environment at a time.

Features related to nodes

After you create a workflow, you can create different types of nodes for data development based on your requirements. For more information, see the "Select a data development node" section in the Overview topic. Different types of nodes provide similar features. In this section, an ODPS SQL node is used as an example to describe the features that are provided by DataWorks on the node configuration tab.



Section	Description
	Node development-related features in the top toolbar: • **Save** (): You can click this icon to save the code and configurations of the current node. • **Save as Ad-Hoc Query Node** (): You can click this icon to save the current code as an ad hoc query. Then, you can view the ad hoc query on the **Ad Hoc Query** tab. For more information, see Create an ad hoc query. • **Submit** (): You can click this icon to commit the current node. • **Unlock** (): You can click this icon to commit the current node and allow other users to modify the code of the node. • **Steal Lock** (): If you are not the owner of the node but you want to modify the node, click this icon. • **Run** (): You can click this icon to run the code of the current node. Values need to be assigned to the variables in the SQL statements only once. If you modify the code, the variables in the code still use the initial values that you assign. **Note** If no resource group for scheduling is specified for the node, DataWorks prompts you to select a resource group for scheduling after you click this icon. • **Run with Parameters** (): You can click this icon to run the code of the current node based

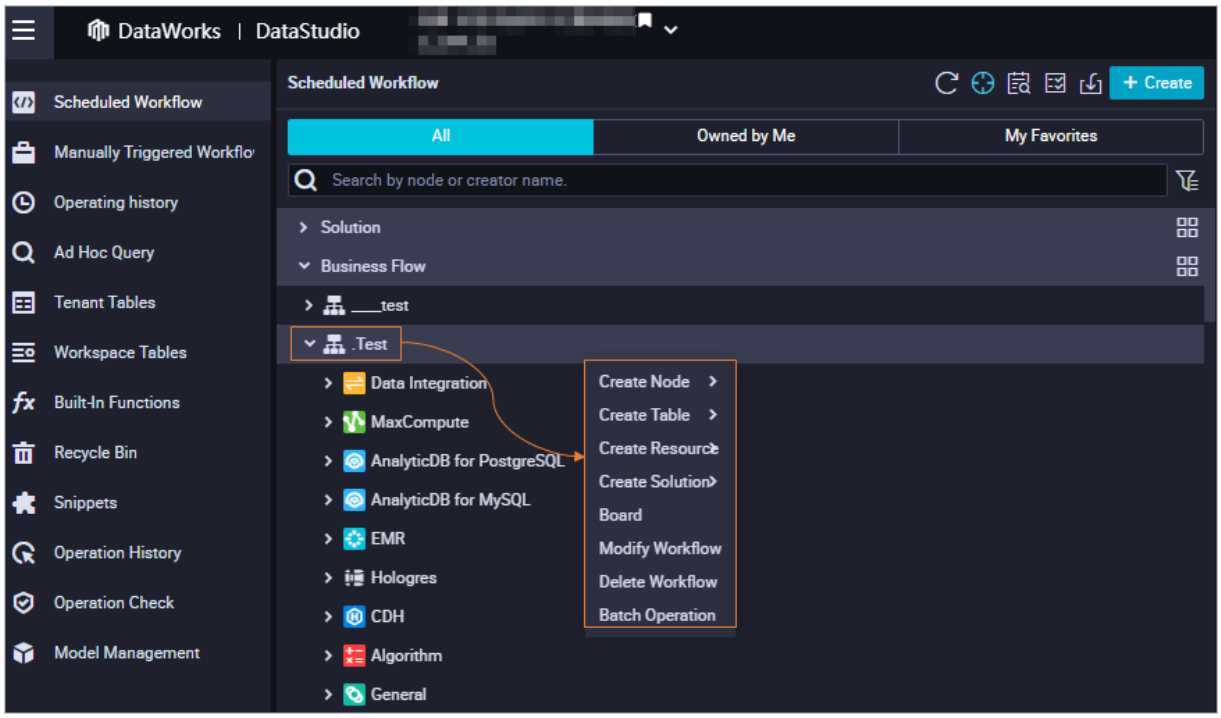
Section	Description
1	on the configured parameters. Each time you click the **Run with Parameters** icon, you must assign values to variables in SQL statements. DataWorks automatically obtains the initial values when you click the icon. After you assign values to custom parameters, DataWorks replaces the initial values with the values assigned to the custom parameters. **Note** If no resource group for scheduling is specified for the node, DataWorks prompts you to select a resource group for scheduling after you click this icon. • **Stop ( )**: You can click this icon to stop the node that is running. • **Reload ( )**: You can click this icon to refresh the current node configuration tab and return to the node configuration tab that is last saved. • **Perform Smoke Testing in Development Environment ( )**: You can click this icon to test the code of the current node in the development environment. Smoke testing in the development environment allows you to simulate the value replacement of scheduling parameters in the production environment. After you select a data timestamp, DataWorks replaces the values in the specified data timestamp with the values that you specified. This feature checks the result of value replacement for scheduling parameters. **Note** Each time you modify the scheduling parameters, you must save and commit the modification before you perform smoke testing in the development environment. Otherwise, the new values of scheduling parameters do not take effect. • **View Log of Smoke Testing in Development Environment ( )**: You can click this icon to view the operation logs of a node that runs in the development environment. • **Access Scheduling System in Development Environment ( )**: You can click this icon to go to Operation Center in the development environment and perform O&M operations. For more information, see View auto triggered node instances. • **Format Code ( )**: You can click this icon to sort the code of the current node. This prevents the code in a single line from being excessively long. • **Share ( )**: You can click this icon to share the current node with other users.

Section	Description
2	**Properties** tab: • General: In this section, you can view the name, ID, and type of the node and set the Owner and Description parameters for the node. • Parameters: In this section, you can add scheduling parameters for the node and dynamically assign values to the parameters. • Schedule: In this section, you can configure time properties for scheduling the node after the node is deployed in the production environment. The time properties include the instance generation mode, recurrence and scheduled time of instances, rerun properties, and timeout period for the node. • Resource Group: In this section, you can specify a resource group for scheduling for the node. • **Dependencies**: In this section, you can configure node dependencies. For more information, see Configure same-cycle scheduling dependencies and Configure previous-cycle scheduling dependencies. • Input and Output Parameters: In this section, you can use context-based parameters to pass the value of the output parameter of an ancestor node to a descendant node based on the assignment feature. Lineage tab: This tab displays the dependencies and auto-captured lineage between the current node and other nodes. Versions tab: A version is generated each time a node is committed and deployed. On this tab, you can view the historical versions and information about each version of the node. The information includes the user that committed the node, the time when the node was committed, change type, status, and remarks. The following content describes the different states of a node version: • **Yes**: The node has been committed to the development environment, but a deployment task has not been created for the node on the Create Deploy Task page. • **Deployed**: The node has been deployed in the production environment. You can view the node on the Cycle Task page in Operation Center in the production environment. For more information, see View auto triggered nodes. • **Not Deployed**: The node has been committed to the development environment but not deployed to the production environment. If you commit the node again, the previously committed version becomes a pending version. • **The deployment is cancelled**: If you commit a node but cancel the deployment of the committed node on the Create Deploy Task page, the state of this committed version becomes The deployment is cancelled. Code Structure tab: This tab uses SQL operators to display the code structure of the node.

Section	Description
3	**SQL Editor**: You can write SQL statements in the editor based on your business requirements. • You can click the icon to return to the first line of the SQL editor. • You can click the icon to view the SQL editor in full screen. • You can click the Run icon to quickly run a code snippet to test whether the code snippet is correctly written. For more information, see Debug a code snippet: Quickly run a code snippet. **Note** This icon is displayed only when you click a line of code.
4	**Features in the upper-right corner**: • Deploy: You can click Deploy to go to the Create Deploy Task page. On this page, you can view the deployment details of the node or perform O&M operations in the production environment after the node is deployed. • Operation Center: You can click this icon to go to Operation Center in the production environment and perform O&M operations.

Shortcut menu related to nodes

Move the pointer over a node and right-click the node. The following figure shows the shortcut menu that appears, and the following table describes the commands supported by the shortcut menu.



Command	Description
Rename	This command allows you to change the name of the node.

Command	Description
Add to Favorites	This command allows you to add the node to favorites. After you add the node to favorites, you can click **My Favorites** in the upper-right corner of the Scheduled Workflow tab to view the node. If you want to remove the node from favorites, right-click the node and select **Remove from Favorites**.
Move	This command allows you to move the node to another workflow.
Clone	This command allows you to clone the node. The new node is of the same type and has the same owner and resource properties as the original node but has a name that is different from that of the original node.
View Versions	This command allows you to view the historical versions and information about each version of the node. The information includes the user that committed the node, the time when the node was committed, change type, status, and remarks.
Locate in Operation Center	This command navigates you to **Operation Center** so that you can view information about the node. If the node is committed to both the development and production environments, you can select **Locate in Operation Center (Production Environment)** or **Locate in Operation Center (Development Environment)**.
Submit for Code Review	This command commits the code of the node for review. A node that is committed by a developer must pass the code review before it can be deployed.
Delete	This command deletes the node and the dependency configurations of its ancestor and descendant nodes. After you click Delete to delete a node that has been deployed to the production environment, you must go to the **Create Deploy Task** page, create a deployment task for the node, and then deploy the node. This way, the node is deleted from the production environment. For more information, see Delete a node.

3.workflow and Solutions

3.1. Create a solution

The data development mode of DataWorks is upgraded so that you can organize multiple workflows in a solution of a workspace.

Context

DataWorks upgrades the data development mode to allow you to organize various types of nodes based on the business category. You can organize workflows to analyze data by business.

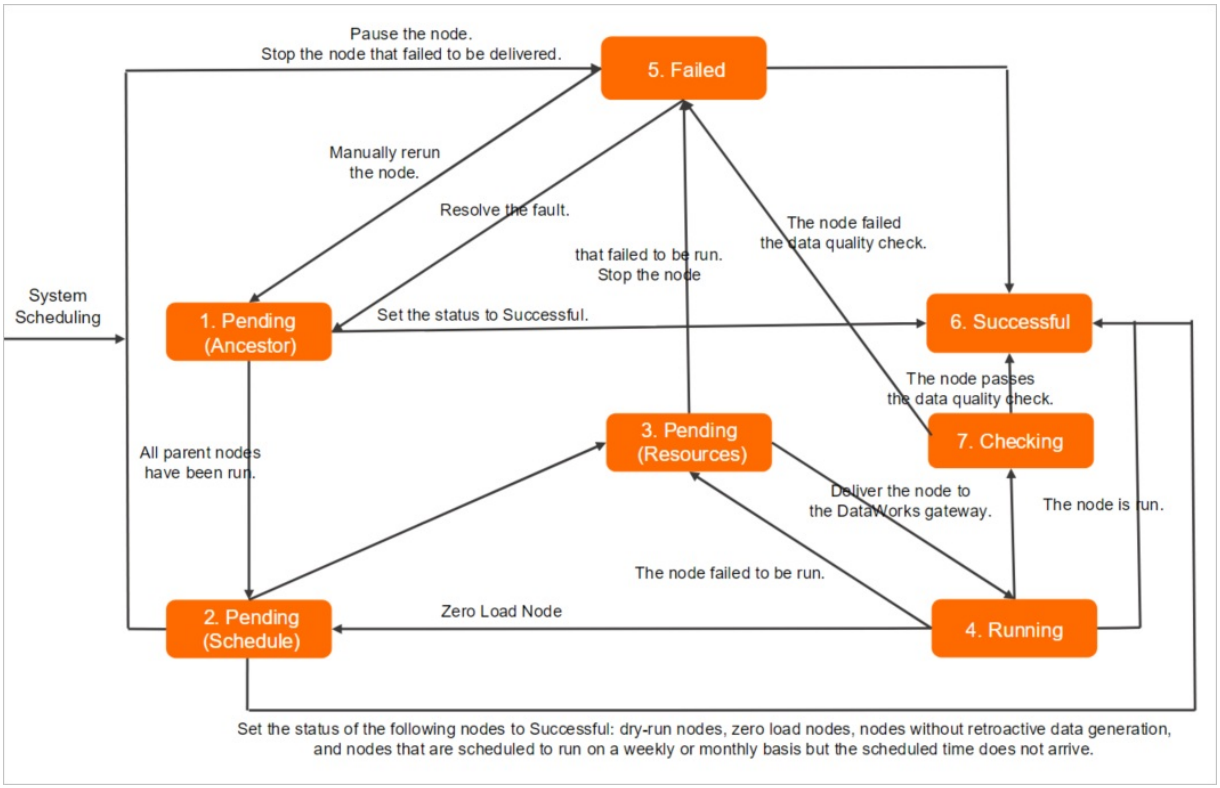
DataWorks defines a new data development process by using the data development mode that involves **workspaces**, **solutions**, and **workflows**. This helps improve user experience.

- A workspace is the basic organizational unit in which the development and O&M permissions of users are managed. Workspace members can collaborate to develop and manage all code in a workspace.
- A solution can contain one or more workflows. Solutions have the following benefits:
  - A solution can contain multiple workflows.
  - Multiple solutions can use the same workflow.
  - Workspace members can collaborate to create and manage all solutions in a workspace.
- A workflow is an abstract entity of business. A workflow allows you to develop data code from a business perspective. A workflow can be added to multiple solutions.

  Workflows have the following benefits:
  - Workflows facilitate business-oriented code development. Nodes in a workflow are organized by node type. The hierarchical directory structure is supported. We recommend that you create a maximum of four levels of subfolders. To create a subfolder, right-click the node type in the Scheduled Workflow pane of the DataStudio page and select **Create Folder**.
  - You can view and optimize each workflow from a business perspective.
  - You can view each workflow on a dashboard and develop code in an efficient manner.
  - You can deploy and manage nodes in each workflow as a whole.

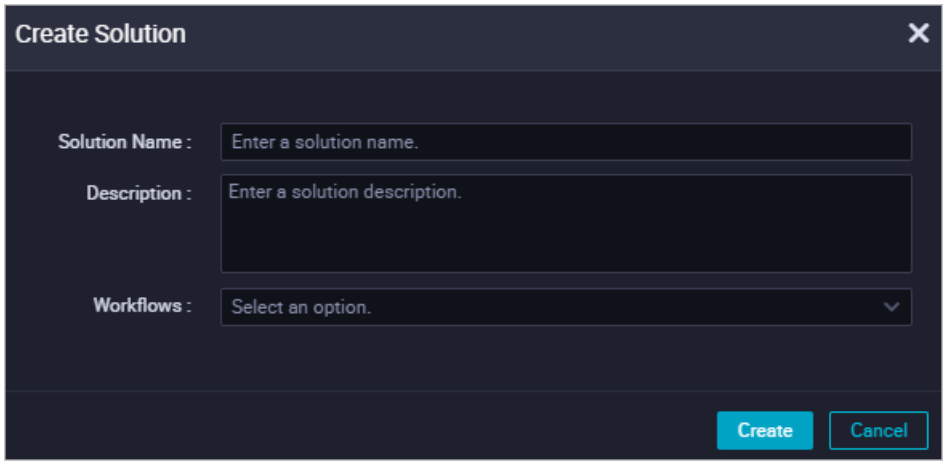
A workflow can be added to multiple solutions. After you develop a solution and add a workflow to the solution, other users can reference and modify this workflow in their own solutions for collaborative development.

The node status model defines six states of a node throughout the lifecycle of the node. The following figure shows the logic of status conversion.



Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the DataStudio page, click **+Create** and select **Create Solution**. In the Create Solution dialog box, configure the parameters.

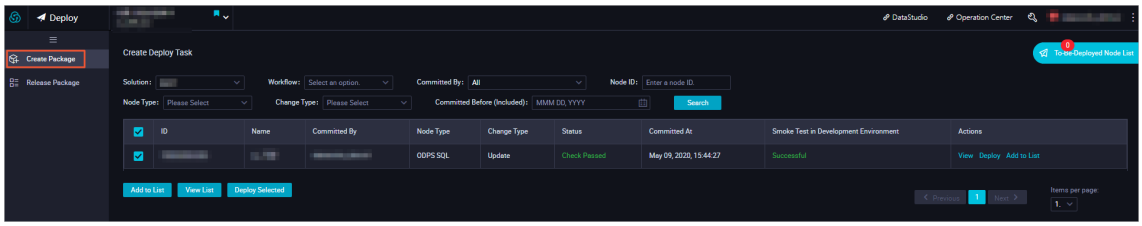


Parameter	Description
Solution Name	The name of the solution. The name cannot exceed 128 characters in length.
Description	The description of the solution. The description cannot exceed 256 characters in length.
Workflows	The workflow that you want to add to the solution. You can add multiple workflows to the solution.

3. After the solution is created, click **Solution** and perform the following operations:
   - Double-click the solution name to view all workflows in the solution. You can click a workflow name to view and edit the workflow. For more information, see Manage workflows.
   - Right-click the solution name and select **Solution Kanban** to view all workflows in the solution and all nodes in the workflows. You can click Change Solution to modify the configuration of the solution.
4. Deploy and manage the solution.

   Move the pointer over the solution name. The and icons appear.

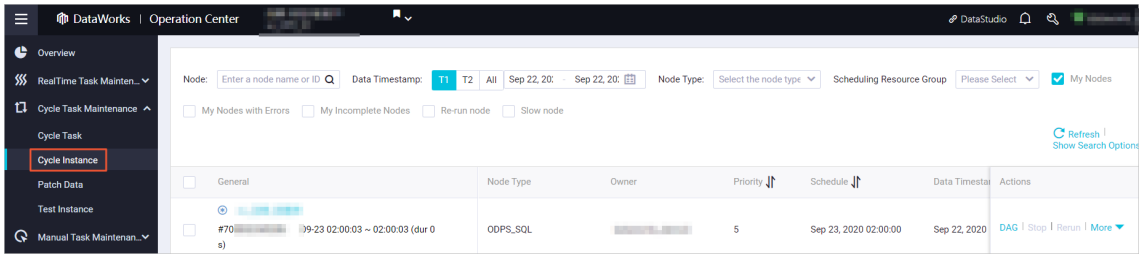
   - Click the icon. The **Create Deploy Task** page appears. On the Deploy Task page, you can view the nodes in the solution that have not been deployed, and deploy the nodes.

     For more information, see Deploy nodes.

     > **Note**
     > - When you deploy a solution, you deploy all workflows in the solution.
     > - A workflow may be reused in multiple solutions. If all workflows in a solution have been deployed in other solutions, you do not need to deploy the solution again.
     > - You cannot specify the order in which the workflows are run in a solution. You can configure scheduling properties for the nodes in workflows. The order in which the workflows are run is determined by the scheduling time of the nodes.

   - Click the icon to go to the **Cycle Instance** page under **Cycle Task Maintenance** in **Operation Center**. The Cycle Instance page displays the auto triggered instances of all nodes in the current solution.



3.2. Manage workflows

You can use a workflow to organize nodes based on business types. This way, you can develop code by the business type. This topic describes how to create, design, commit, and view a workflow and how to modify or delete multiple nodes in a workflow at a time.

Context

A workspace supports various types of compute engines and can contain multiple workflows. A workflow is a collection of multiple types of objects. The object types include Data Integration, Data Analytics, table, resource, function, and algorithm.

Each type of object corresponds to an independent folder. You can create subfolders in the folder. To facilitate the management of objects, we recommend that you create no more than four levels of subfolders. If you create more than four levels of subfolders, your workflow becomes excessively complex. In this case, we recommend that you split your workflow into two or more workflows and add the workflows to the same solution to improve work efficiency.

Design the organizational structure

You can organize your business based on **workspaces**, **solutions**, and **workflows**. You can plan and group workspaces based on enterprise departments, business projects, and data warehouse layers.

Concept	Description	Purpose
Workspace	You can specify administrators and members for each workspace based on your business requirements. The role settings of members and parameters for a compute engine instance are different among workspaces. For more information about workspace planning, see Plan workspaces.	Workspaces are basic units for managing permissions in DataWorks. You can create workspaces based on the organizational structure of your company. You can use a workspace to manage development permissions and O&M permissions. Workspace members can collaborate to develop and manage the code for all nodes in a workspace.
Solution	A solution is a group of workflows that are dedicated to a specific business goal. A workflow can be added to multiple solutions. After you develop a solution and add a workflow to the solution, other users can reference and modify the workflow in their solutions or workflows for collaborative development.	You can use a solution for business integration.

Concept	Description	Purpose
Workflow	A workflow is an abstract business entity that allows you to develop code based on your business requirements. Workflows and nodes in different workspaces are separately developed. Workflows can be displayed in a directory tree or in a panel. The display modes enable you to organize code from the business perspective and show the resource classification and business logic in a more efficient manner. • The directory tree allows you to organize your code by node type. • The panel shows the business logic in a workflow.	A workflow is a basic unit for code development and resource management.

DataStudio works based on nodes in a workflow. You can create one or more nodes in a specific workflow in the panel. In each workflow, nodes are grouped by engine type. In the section of a specific engine, nodes are classified into data synchronization nodes, tables, resources, and functions. These components can be used to meet a specific business goal. Only the components that are used in a workflow are displayed in the workflow.

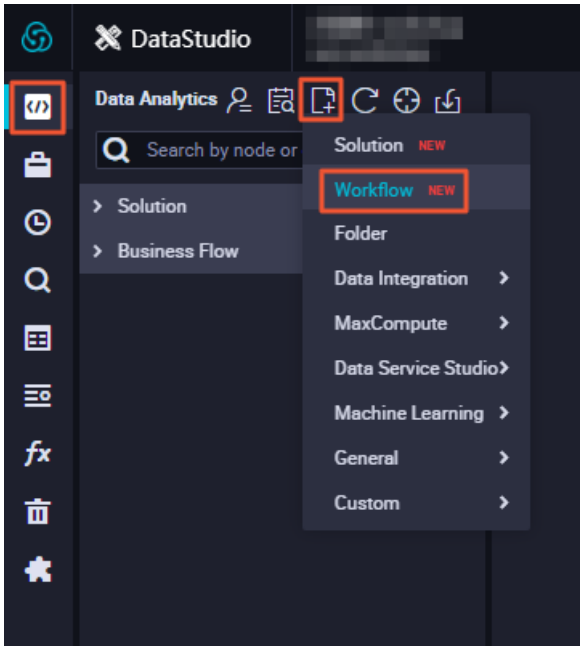
- To use DataStudio, you must create a workflow.
- If you change the code for a node in the production environment, you must modify node parameters on the DataStudio page. Then, commit and deploy the node.

> **Note**
>
> - If no compute engine is available in your workspace or the compute engine that you want to use is not displayed in the directory tree, check whether the service corresponding to the compute engine type is activated and whether the compute engine is associated with your workspace on the Workspace Management page. Only the compute engines that are associated with the workspace are displayed in the directory tree. For more information about how to associate a compute engine with a workspace, see Configure a workspace.
> - If you cannot use specific features or cannot find an entry used to create an object, go to the User Management page to check whether you have developer permissions. You have developer permissions if you use an Alibaba Cloud account or you log on to the DataWorks console as a RAM user that is assigned the developer role or workspace administrator role. You can also check whether the DataWorks edition that you adopted meets the requirements.
> - If you create more than four levels of subfolders, your workflow becomes excessively complex. In this case, we recommend that you split your workflow into two or more workflows and add the workflows to the same solution to improve work efficiency.

Create a workflow

In DataStudio, data development is implemented by using the components such as nodes in workflows. Before you create a node, create a workflow.

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Move the pointer over the **+Create** icon and click **Workflow**.



3. In the **Create Workflow** dialog box, set the **Workflow Name** and **Description** parameters.

> **Notice** The workflow name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Create**.

Design a workflow

Code development is implemented in workflows. To develop code in a workflow, you can create a node under a folder of a compute engine type in the directory tree. You can also double-click a workflow. On the workflow configuration tab, drag the components including nodes of different compute engine types to the canvas and connect the components to form a directed acyclic graph (DAG).

When you design a workflow, take note of the following items:

- We recommend that you create no more than 100 nodes in a workflow.

> **Note** If the total number of nodes in a workflow exceeds 1,000, the DAG of the workflow cannot be viewed.

- In the DAG, you can draw a line between two nodes to configure dependencies between the two nodes. You can also open the Properties panel on the configuration tab of a node and configure node dependencies in the panel. For more information, see Logic of same-cycle scheduling dependencies.
- If you create a node in the directory tree of a workflow, the node dependencies can be configured based on the lineage in the code. For more information, see Logic of same-cycle scheduling dependencies.

Design the business logic

DataWorks encapsulates the capabilities of different compute engines in different types of nodes. You can use nodes of different compute engine types to develop data without the need to run complex commands on compute engines. You can also use the general nodes of DataWorks to design complex logic.

In a workflow, you can configure components such as data integration nodes and data analytics nodes.

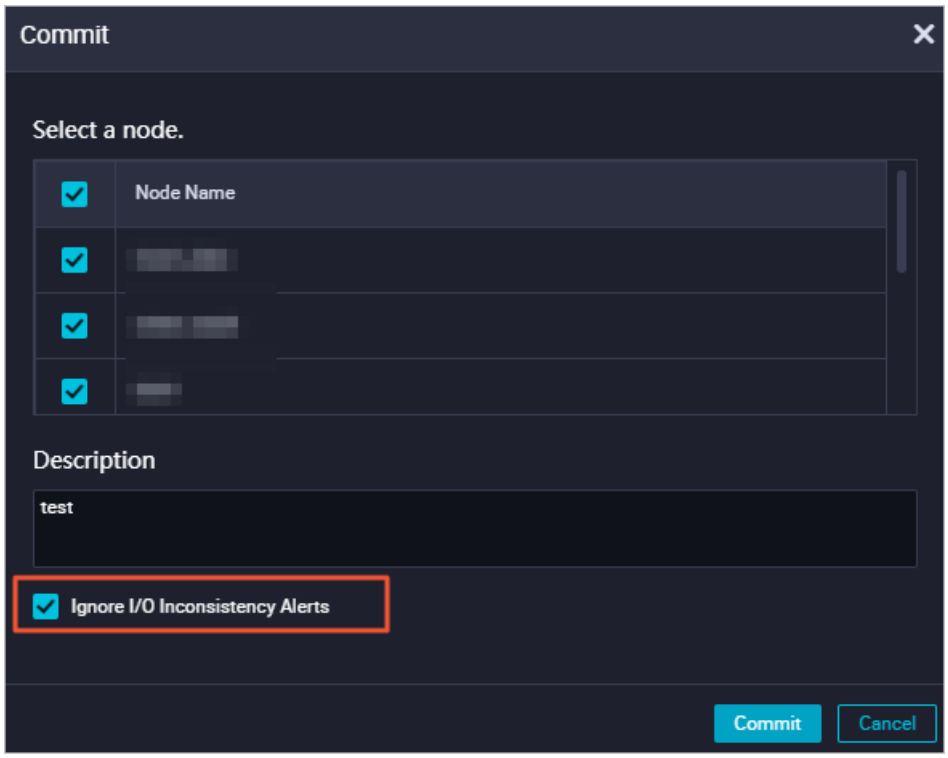
- You can configure data integration nodes including batch synchronization nodes and real-time synchronization nodes to synchronize data between databases.
- You can configure data analytics nodes for data cleansing. You can also add required resources and create required functions in a visualized mode.

> **Note**
> - For more information about the supported types of nodes that encapsulate the capabilities of different compute engines and the supported features for development in DataWorks, see Select a data development node.
> - For more information about how to configure scheduling dependencies, see Configure basic properties.

Commit a workflow

In a workspace in standard mode, the DataStudio page only allows you to develop and test nodes in the development environment. To commit the code to the production environment, you can commit multiple nodes in the workflow at a time and deploy them on the Deploy page.

1. After you design a workflow, click the icon in the toolbar.
2. In the **Commit** dialog box, select the nodes that you want to commit and enter your comments in the **Change description** field. Then, determine whether to select **Ignore I/O Inconsistency Alerts** based on your business requirements. If you do not select **Ignore I/O Inconsistency Alerts**, an error message is displayed if the system determines that the input and output that you set do not match with those identified in code lineage analysis. For more information, see When I commit a node, the system reports an error that the input and output of the node are not consistent with the data lineage in the code developed for the node. What do I do?.

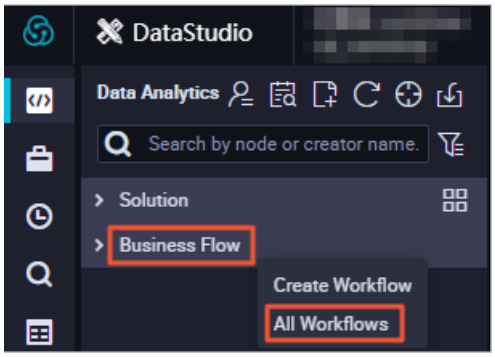


3. Click **Commit**.

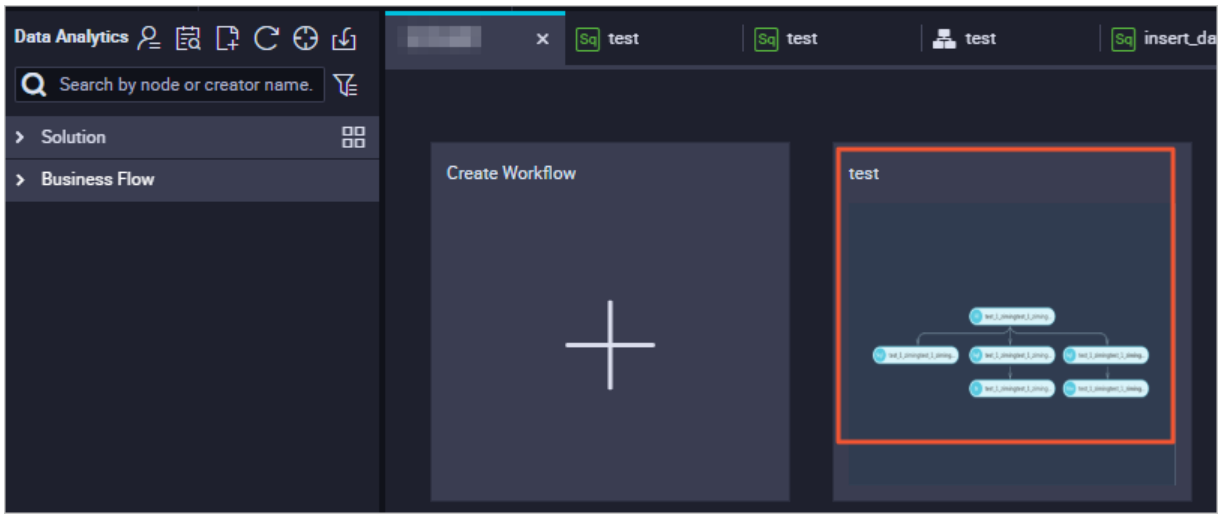
> **Note** If you have modified the code or properties of a node and committed the node on its configuration tab, you cannot select the node in the Commit dialog box. If you have modified the code or properties of a node but have not committed the node on its configuration tab, you can select the node in the Commit dialog box.

View all workflows

In the **Scheduled Workflow** pane, right-click **Business Flow** and select **All Workflows** to view all the workflows in the current workspace.



Click the card of a workflow. The configuration tab of the workflow appears.



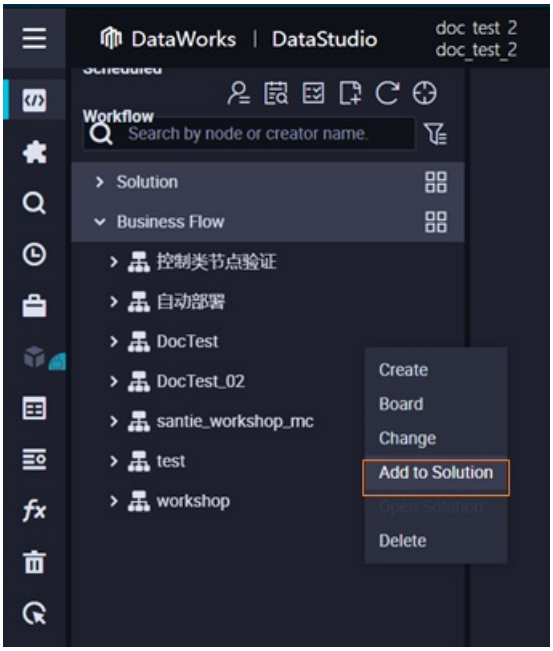
Manage workflows by using the solution feature

You can include one or more workflows in a solution. Solutions have the following benefits:

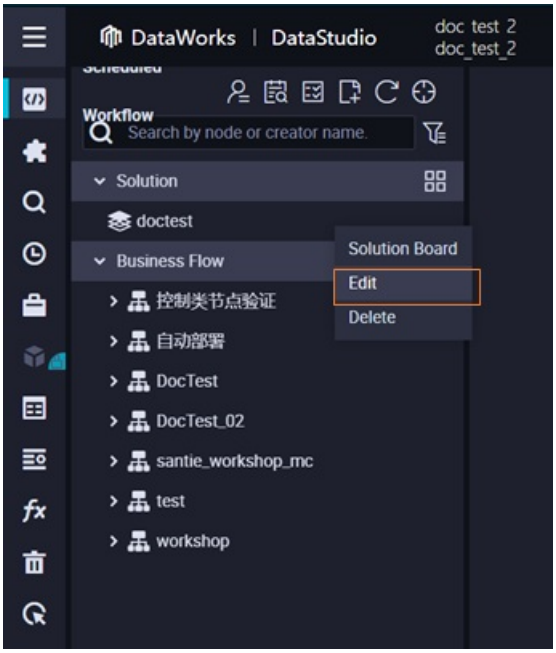
- A solution can contain multiple workflows.
- A workflow can be added to multiple solutions.
- Workspace members can collaboratively develop and manage all solutions in a workspace.

If you manage workflows by using solutions, you can perform the following operations:

- Add a workflow to a solution.



- Add multiple workflows to a solution at a time. To do so, right-click a solution, select Edit, and then modify the Workflows parameter in the Change Solution dialog box.



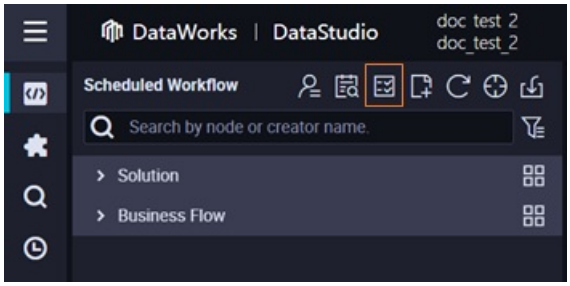
Modify or delete multiple nodes at a time

If you want to modify or delete multiple nodes of the same type, such as all batch synchronization nodes, in the current workspace at a time, you can use the parameters on the **Node** tab to find the nodes and modify or delete the nodes. The parameters include **Node type**, **Business processes**, and **Scheduling Resource Group**.

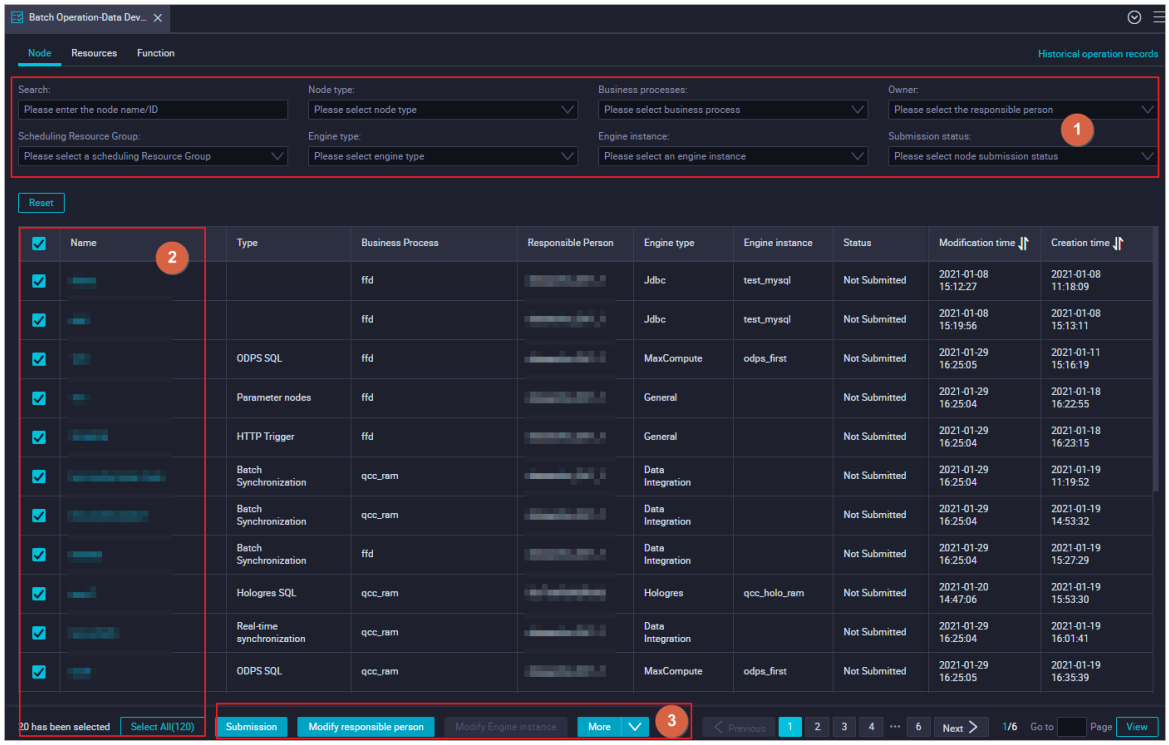
> **Note** You can modify only the owners and resource groups for scheduling of multiple nodes at a time.

1. On the **DataStudio** pane, click the icon in the upper-right corner of the **Scheduled Workflow**

pane to go to the **Node** tab.



2. Modify or delete nodes.



i. Configure filter conditions such as the node name, node ID, node type, and workflow to find the nodes that you want to modify or delete.

ii. Select partial or all nodes.

iii. Modify or delete the nodes.

- To modify the selected nodes, click **Modify responsible person** or **Modify scheduling Resource Group**. You can modify only the owners and resource groups for scheduling of multiple nodes at a time.

  If you set the **Mandatory modification** parameter to **Yes** in the dialog box that appears, you can modify all the selected nodes. If you set this parameter to **No**, you can modify only the nodes that are locked by yourself.

- To delete the selected nodes, choose **More > Delete** in the lower part of the Node tab.

  If you set the **Force delete** parameter to **Yes** in the **Delete node** dialog box, you can delete all the selected nodes. If you set this parameter to **No**, you can delete only the nodes that are locked by yourself.

Export a common workflow for replication

You can use the node group feature to quickly group all nodes in a workflow as a node group and then reference the node group in a new workflow. For more information, see Create and reference a node group.

Export multiple workflows from a DataWorks workspace at a time and import them to other DataWorks workspaces or open source engines

If you want to export multiple workflows in a workspace from DataWorks at a time and import them to other DataWorks workspaces or open source engines, you can use the Migration Assistant service of DataWorks. For more information, see Overview.

3.3. Manage manually triggered workflows

In a manually triggered workflow, all nodes must be manually triggered, and cannot be scheduled. Therefore, you do not need to specify a parent node or output for a node in a manually triggered workflow.

Create a manually triggered workflow

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the left-side navigation submenu, click the **Manually Triggered Workflows** icon.

   You can click the icon in the lower-left corner to show or hide the left-side navigation submenu.
3. Right-click **Manually Triggered Workflows** and select **Create Workflow**.
4. In the **Create Workflow** dialog box, set the **Workflow Name** and **Description** parameters.

   > **Notice** The workflow name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
5. Click **Create**.

GUI elements

The following table describes the GUI elements on the configuration tab of a manually triggered workflow.

No.	GUI element	Description
1	**Submit**	Commit all nodes in the manually triggered workflow.

No.	GUI element	Description
2	**Run**	Run all nodes in the manually triggered workflow. Nodes in this workflow do not have dependencies. Therefore, they can be run at the same time.
3	**Stop**	Stop running nodes.
4	**Deploy**	Go to the **Deploy** page. On this page, you can deploy specific or all nodes that are committed but not deployed to the production environment.
5	**Go to Operation Center**	ou can click this icon to go to Operation Center to view the O&M details of nodes.
6	**Switch Layout**	You can click this icon to switch the layout of the canvas to **Vertical Horizontal** or **Grid**.
7	**Box**	Draw a box to select required nodes to form a node group.
8	**Refresh**	Refresh the configuration tab of the manually triggered workflow.
9	**Format**	You can click this icon to horizontally align the nodes on the canvas.
10	**Adapt**	You can click this icon to adapt the current workflow layout to the size of the canvas.
11	**Center**	You can click this icon to center nodes on the canvas.
12	**1:1**	You can click this icon to adjust the scale of the directed acyclic graph (DAG) of nodes to 100%.
13	**Zoom In**	Zoom in the directed acyclic graph (DAG).
14	**Zoom Out**	Zoom out the DAG.
15	**Search**	You can click this icon and enter a keyword in the search box to search for a node whose name contains the keyword.
16	**Toggle Full Screen View**	Display nodes in the manually triggered workflow in the full screen.
17	**Hide Engine Information**	Show or hide engine information.
18	**Workflow Parameters**	Set parameters for the manually triggered workflow. Parameters set on this tab takes priority over those set on a node configuration tab. If the same parameter is set for both the workflow and a node, the parameter setting on the Workflow Parameters tab takes effect.

No.	GUI element	Description
19	**Change History**	View the operation records of all nodes in the manually triggered workflow.
20	**Versions**	View the deployment records of all nodes in the manually triggered workflow.

Composition of a manually triggered workflow

> **Note** We recommend that you create no more than 100 nodes in a manually triggered workflow.

A manually triggered workflow can contain a variety of nodes. After you create a manually triggered workflow based on the preceding procedure, you can create nodes in the workflow. For more information, see Manage workflows.

- **Data Integration**

  Double-click **Data Integration** under a manually triggered workflow to view all the data integration nodes in the workflow.

  Right-click **Data Integration** and choose **Create > Batch Synchronization** to create a batch sync node. For more information, see Batch Sync node.

- **MaxCompute**

  > **Notice** The **MaxCompute** folder appears on the page only after you add a MaxCompute compute engine on the **Workspace Management** page. For more information, see Configure a workspace.

  The MaxCompute compute engine allows you to create data analytics nodes, such as ODPS SQL, SQL Snippet, ODPS Spark, PyODPS 2, ODPS Script, ODPS MR, and PyODPS 2 nodes. You can also create and view tables, resources, and functions.

  - **Data Analytics**

    Click **MaxCompute** under the manually triggered workflow and right-click **Data Analytics** to create a data analytics node. For more information, see Create an ODPS SQL node, Create an SQL component node, Create a MaxCompute Spark node, Create a PyODPS 2 node, Create an ODPS Script node, Create an ODPS MR node, and Create a PyODPS 3 node.

  - **Table**

    Click **MaxCompute** under the manually triggered workflow and right-click **Table** to create a table. You can also view all the tables that are created in the current MaxCompute compute engine. For more information, see Create a MaxCompute table.

  - **Resource**

    Click **MaxCompute** under the manually triggered workflow and right-click **Resource** to create a resource. You can also view all the resources that are created in the current MaxCompute compute engine. For more information, see Create a MaxCompute resource.

- **Function**

  Click **MaxCompute** under the manually triggered workflow and right-click **Function** to create a function. You can also view all the functions that are created in the current MaxCompute compute engine. For more information, see Register a MaxCompute function.

- **EMR**

  > **Notice** The **EMR** folder appears on the page only after you add an E-MapReduce compute engine on the **Workspace Management** page. For more information, see Configure a workspace.

  The E-MapReduce compute engine allows you to create data analytics nodes, such as EMR Hive, EMR MR, EMR Spark SQL, EMR Spark, EMR Shell, EMR Spark Shell, EMR Presto, and EMR Impala nodes. You can also create and view E-MapReduce resources.

  - **Data Analytics**

    Click **EMR** under the manually triggered workflow and right-click **Data Analytics** to create a data analytics node. For more information, see Create an EMR Hive node, Create and use an EMR MR node, Create an EMR Spark SQL node, Create an EMR Spark node, and Create an EMR Presto node.

  - **Resource**

    Click **EMR** under the manually triggered workflow and right-click **Resource** to create a resource. You can also view all the resources that are created in the current E-MapReduce compute engine.

  - **Function**

    Click **EMR** under the manually triggered workflow and right-click **Function** to create a function. You can also view all the resources that are created in the current E-MapReduce compute engine.

- **Algorithm**

  Click the manually triggered workflow and right-click **Algorithm** to create an algorithm. You can also view all the Machine Learning experiment nodes that are created in the manually triggered workflow. For more information, see Create a Machine Learning (PAI) node.

- **General**

  Click the manually triggered workflow and right-click **General** to create relevant nodes. For more information, see Create a Shell node and Create a zero-load node.

- **UserDefined**

  Click the manually triggered workflow and right-click **UserDefined** to create relevant nodes. For more information, see Create a Data Lake Analytics node and Create an AnalyticDB for MySQL node.

4.Create and manage tables

4.1. Create a table

4.1.1. Create a MaxCompute table

This topic describes how to create a MaxCompute table.

Prerequisites

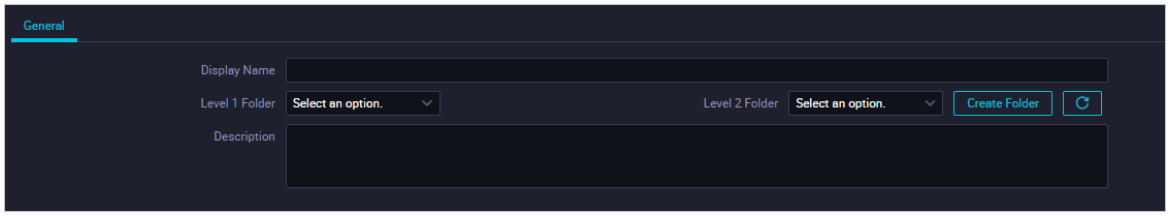
A MaxCompute compute engine instance is associated with the workspace in which you want to create a MaxCompute table. The MaxCompute folder is displayed on the DataStudio page only after you associate a MaxCompute compute engine instance with the workspace on the **Workspace Management** page. For more information, see Configure a workspace.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region where the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the page that appears, move the pointer over the +Create icon and choose **Create Table > MaxCompute > Table**.

   Alternatively, you can click the desired workflow in the Business Flow section, right-click **MaxCompute**, and then select **Create Table**.
3. In the **Create table** dialog box, select a path from the **Path** drop-down list, configure the **Name** parameter, and then click **Create**.

   > **Notice**
   > - The table name can be a maximum of 100 characters in length. The table name must start with a letter and cannot contain special characters.
   > - If multiple MaxCompute compute engine instances are associated with the current workspace, you must select one from the **Engine Instance** drop-down list.
4. In the **General** section, configure the parameters.



Parameter	Description
Display Name	The name of the MaxCompute table.
Level-1 Folder	The name of the level-1 folder where the table resides. **Note** The level-1 and level-2 folders show the table locations in DataWorks for you to manage tables with ease.
Level-2 Folder	The name of the level-2 folder where the table resides.
Create Folder	Click **Create Folder** to go to the **Folder Management** tab of the Table Management tab. On the Folder Management tab, you can create level-1 and level-2 folders for tables. After you create a folder, click next to Create Folder to synchronize the folder.
Description	The description of the table.

5. Create a table.

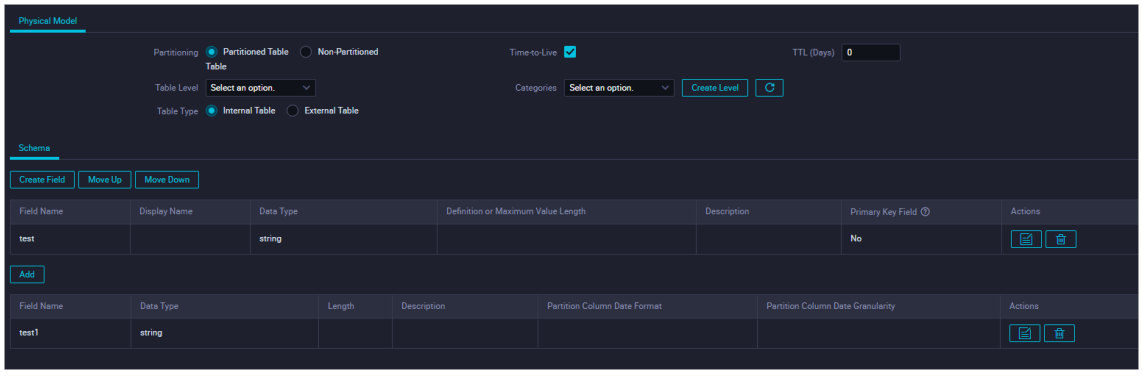
   You can use one of the following methods to create a table:

   - Create a table by using a DDL statement.

     Click **DDL Statement** in the toolbar. In the dialog box that appears, enter the table creation statement and click **Generate Table Schema**. Then, parameters in the **Physical Model** and **Schema** sections are automatically configured. For more information about the table creation statement, see the "Create a table" section in Table operations.

   - Create a table on the graphical user interface (GUI).

     DataWorks allows you to create tables on the GUI.



Category	Parameter	Description
	Partitioning	Valid values: **Partitioned Table** and **Non-Partitioned Table**.

Category	Parameter	Description
Physical model design	Time-to-Live	Specifies whether to enable the Time-to-Live feature for the table. If you enable the Time-to-Live feature, you must enter a number in the **TTL (Days)** field. If the table or partition is stored for more than the specified number of days, data that has not been updated is deleted.
	Layer	The layer where the table data is stored or processed. A data warehouse consists of the operational data store (ODS), common data model (CDM), and application data store (ADS) layers. You can customize a name for each layer.
	Category	The category of the table. Tables are categorized into basic services, advanced services, and other services. You can customize a name for each category. If you want to create a table category or layer, click **Create Level**. On the **Table Management** tab, click the Level Management tab. **Note** Categories are designed only to help you manage the table and do not affect underlying implementation.
Schema design	Create Field	The button for adding a field. To add a field, click **Create Field**, configure the field information, and then click the **save** icon in the Actions column.
	Move Up	The buttons for adjusting the field sequence of the table. If you want to adjust the sequence of fields in an existing table, you must delete the table and create another table with the same name. These operations are forbidden in the production environment.
	Move Down	
	Field Name	The name of the field. The name can contain letters, digits, and underscores (_).
	Display Name	The display name of the field.
	Data Type	The data type of the field. MaxCompute supports the following data types: TINYINT, SMALLINT, INT, BIGINT, FLOAT, DOUBLE, DECIMAL, VARCHAR, CHAR, STRING, BINARY, DATETIME, DATE, TIMESTAMP, BOOLEAN, ARRAY, MAP, and STRUCT. For more information, see Data types.
	Definition or Maximum Value Length	The length limit of the field. You must configure this parameter if the data type that you specify for the field has a length limit.
	Description	The description of the field.

Category	Parameter	Description
	Primary Key Field	Specifies whether the field serves as the primary key or part of a composite primary key.
	Edit icon	The **icon** for editing the field. After you save the field, you can click this icon to edit the field and then click the **save** icon to save the edited field.
	Delete icon	The icon for deleting the field. **Note** If you want to delete a field from an existing table and then commit the table, you must delete the table and create another table with the same name. These operations are forbidden in the production environment.
Partition field design **Note** This section appears only when you set Partitioning to Partitioned Table in the Physical Model section.	**Add**	The button for adding a partition. If you set the **Partitioning** parameter to **Partitioned Table** in the **Physical Model** section, you must configure a partition for the table. You can add a partition to the current table. If you want to add a partition to an existing table and then commit the table, you must delete the table and create another table with the same name. These operations are forbidden in the production environment.
	Data Type	The data type of the partition field. We recommend that you use the STRING type for all partition fields.
	Partition Column Date Format	The format of the date partition. If the partition field is a date, although the data type may be STRING, select or enter a date format, such as *yyyymmdd* or *yyyy-mm-dd*.
	Partition Column Date Granularity	The granularity of the date partition. The granularity can be second, minute, hour, day, month, quarter, or year. You can enter a partition granularity based on your business requirements. If you want to add multiple partitions with different granularities, note that a greater granularity corresponds to a higher partition level. For example, if you add three partitions whose granularities are day, hour, and month, respectively, to the table, the table contains three partitions: a level-1 partition (month), a level-2 partition (day), and a level-3 partition (hour).
	Delete icon	The icon for deleting the partition. If you want to delete a partition from an existing table and then commit the table, you must delete the table and create another table with the same name. These operations are forbidden in the production environment.

6. Click **Commit to Development Environment** and **Commit to Production Environment** in sequence.

   If you are using a workspace in basic mode, you need only to click **Commit to Production Environment**.

Button	Description
Load from Development Environment	If the table has been committed to the development environment, this button is clickable. After you click this button, the information about the table you create in the development environment overwrites the table information on the current page. **Note** This feature is supported only for MaxCompute tables.
Commit to Development Environment	Before you click this button, make sure that you have configured all required parameters on the current page. You cannot click this button if any parameter is not configured.
Load from Production Environment	After you click this button, the information about the table that is committed to the production environment overwrites the table information on the current page. **Note** This feature is supported only for MaxCompute tables.
Commit to Production Environment	After you click this button, the table is created in the workspace of the production environment.

What's next

After the table is created, you can query the table data, modify the table, or delete the table. For more information, see Manage tables.

4.1.2. Create an AnalyticDB for PostgreSQL table

This topic describes how to create an AnalyticDB for PostgreSQL table.

Prerequisites

- DataWorks Standard Edition or higher is activated.
- An exclusive resource group for scheduling or a custom resource group is added based on business needs. For more information, see Purchase an exclusive resource group for scheduling or Create and use a custom resource group for scheduling.

  The resource group added when you bind an AnalyticDB for PostgreSQL instance passes the connectivity test.
- An AnalyticDB for PostgreSQL instance is bound to the workspace where you want to create an ADB for PostgreSQL table. The AnalyticDB for PostgreSQL service is available in a workspace only after you bind an AnalyticDB for PostgreSQL instance to the workspace on the **Workspace Management**

page. For more information, see Configure a workspace.

- The metadata of the bound AnalyticDB for PostgreSQL instance is collected on the **DataMap** page.

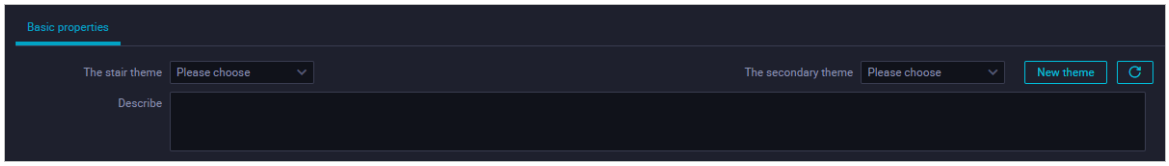
Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the **Data Development** tab, move the pointer over the **+Create** icon and choose **AnalyticDB > Table**.

   Alternatively, you can click a workflow in the Business process section, right-click **AnalyticDB for PostgreSQL**, and then choose **New > AnalyticDB for PostgreSQL Table**.
3. In the **New table** dialog box, set the **Table name** parameter.

   > **Notice**
   > - The table name is in the format of schema_name.table_name.
   > - The schema name and table name must be 1 to 63 characters in length, and can contain letters, digits, and underscores (_). They must start with a letter or underscore (_).
   > - If you have bound multiple AnalyticDB for PostgreSQL instances to the current workspace, you must select one based on your needs.
4. Click **Submit**. The table configuration tab appears.

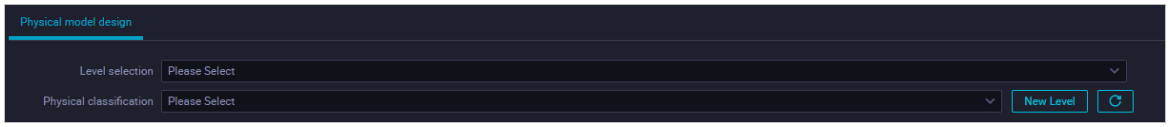
   The upper part of the table configuration tab displays the table name and AnalyticDB for PostgreSQL instance name.
5. In the **Basic properties** section, set the parameters as required.



Parameter	Description
The stair theme	The name of the level-1 folder where the table resides. **Note** Level-1 and level-2 folders show the table locations in DataWorks for you to manage tables more conveniently.

Parameter	Description
The secondary theme	The name of the level-2 folder where the table resides.
New theme	Click **New theme** to go to the **Theme management** tab. On this tab, you can create level-1 and level-2 folders for tables. After you create a folder, click the icon next to New theme to synchronize the folder.
Description	The description of the table.

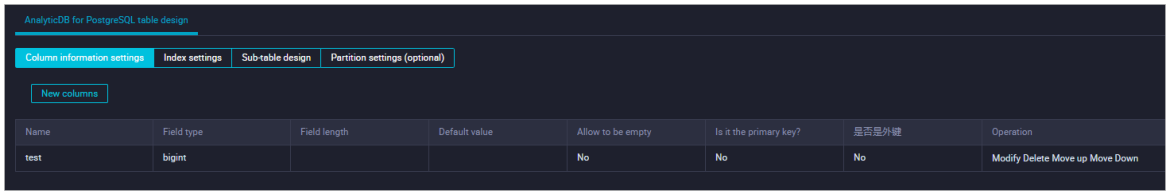
6. In the **Physical model design** section, set the parameters as required.



Parameter	Description
Level selection	The layer where the table data is stored or processed. Generally, a data warehouse consists of the operational data store (ODS), common data model (CDM), and application data store (ADS) layers. You can specify a custom name for each layer.
Physical classification	The category of the table. Tables are categorized into basic services, advanced services, and other services. You can specify a custom name for each category. **Note** Categories are designed only for your management convenience and do not involve underlying implementation.
New Level	The levels and categories to be created. To add levels and categories, click **New Level** to go to the **Hierarchical management** tab. After levels and categories are created, click the icon.

7. In the **AnalyticDB for PostgreSQL table design** section, set the parameters as required.

   You can configure the schema of an AnalyticDB for PostgreSQL table on the following tabs: **Column information settings**, **Index settings**, **Sub-table design**, and **Partition settings (optional)**.



Tab	Parameter	Description
Column information settings	**New columns**	Allows you to click the button and set the relevant parameters to create a field.
	Name	The name of the field.
	Field type	The data type of the field.
	Field length	The length of the field. You can specify the length for fields of only some specific data types.
	Default value	The default value of the field.
	Allow to be empty	Specifies whether the field can be empty.
	Is it the primary key?	Specifies whether the field serves as the primary key.
	Foreign key	Specifies whether the field serves as a foreign key.
	Operation	◦ You can perform the following operations on a new field: save, cancel, delete, move up, and move down. ◦ You can perform the following operations on an existing field: modify, delete, move up, and move down.
Index settings	**New columns**	Allows you to click the button and set the relevant parameters to create an index.
	Index name	The name of the index. Make sure that you specify a unique name.
	Include columns	The field on which the index is to be created. To select a field, click **Edit**. In the **Select at least one index** dialog box, click the **+** icon. All created fields appear in the Column information drop-down list. Select the field from the **Column information** drop-down list and click **Save**.
	Index type	The type of the index. Valid values: **Normal**, **Primary Key**, and **Unique**.
	Index mode	The mode for indexing data in the fields. Valid values: **B-tree**, **Bitmap**, and **GiST**.
	Operation	◦ You can perform the following operations on a new index: save, cancel, delete, move up, and move down. ◦ You can perform the following operations on an existing index: modify, delete, move up, and move down.

Tab	Parameter	Description
Sub-table design	**Hash (Recommended)**, **Copy Schema**, and **Random (Not Recommended)**	The way in which the partition key is generated. Take **Hash (Recommended)** as an example. Click **New columns** and select the target field from the **Name** drop-down list. The information about the selected field appears. Click **Save**. For more information, see the **Column information settings** section of this table.
Partition settings (optional)	**Partition settings (optional)**	The partitions of the table. You can configure the partitions as required.

8. Click **Submit to development environment** and **Submit to production environment** in sequence.

   If you are using a workspace in basic mode, you only need to click **Submit to production environment**.

9. In the **Submit changes** dialog box, confirm that the table creation statements are correct, select a resource group from the **Select a resource group** drop-down list, and then click **Confirm execution**.

> **Note**
> - You can select only an exclusive resource group for scheduling or a custom resource group.
> - The selected resource group must be the same as the one that passes the connectivity test when you bind an AnalyticDB for PostgreSQL instance.

What's next

After the AnalyticDB for PostgreSQL table is created, you can query the table data and modify or delete the table. For more information, see Manage tables.

4.1.3. Create an EMR table

This topic describes how to create an E-MapReduce (EMR) table.

Prerequisites

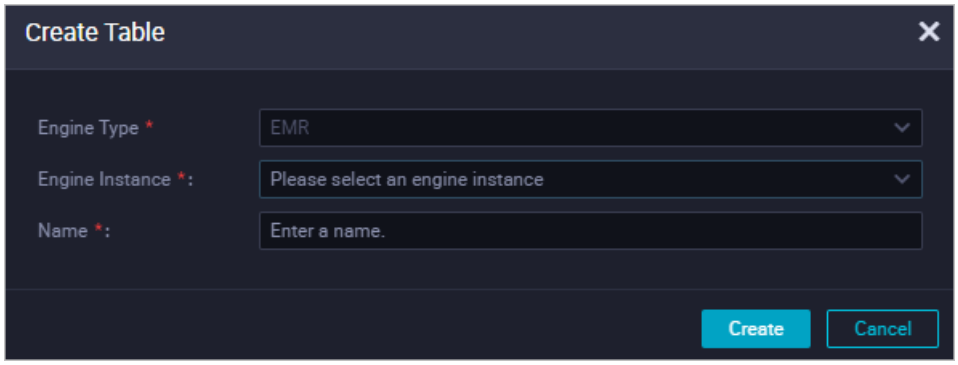
- An Alibaba Cloud EMR cluster is created, and an inbound rule that contains the following content is added to the security group to which the cluster belongs.
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- An EMR compute engine instance is associated with your workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Configure a workspace.

- The metadata of an EMR data source is collected in **Data Map** so that you can select an EMR database when you create a table. For more information, see Collect metadata from an EMR data source.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Move the pointer over the icon and choose **Create Table > EMR > table**.

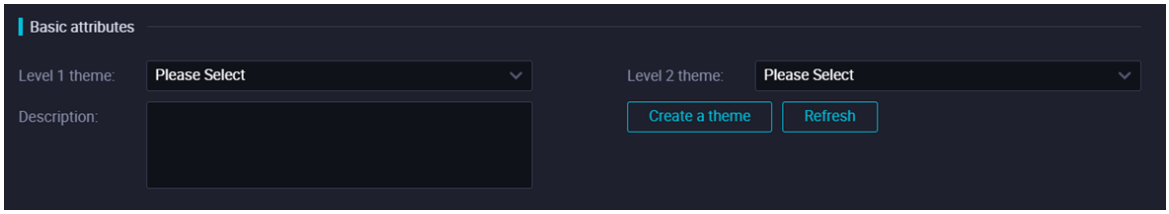
   You can also find the workflow in which you want to create an EMR table, right-click **EMR**, and then click **Create Table**.
3. In the **Create Table** dialog box, set the parameters as required.



Parameter	Description
Engine type	The default value is EMR, which cannot be modified.
Engine Instance	Select an engine instance from the drop-down list.
Name	The name of the EMR table.

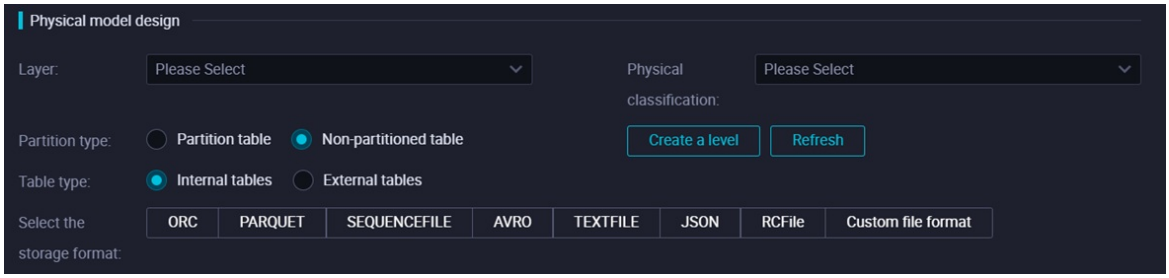
4. Click **Create**. The table configuration tab appears.

   The upper part of the tab shows the configurations that you specified in the **Create Table** dialog box. You can modify the database in which the EMR engine instance resides. To create a database, click **Create a database**. In the **Create a database** dialog box, set the parameters and click **OK**.
5. In the **Basic attributes** section, set the parameters as required.



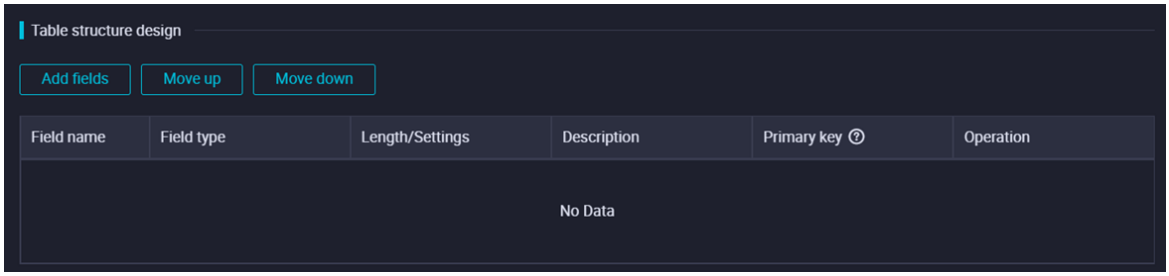
Parameter	Description
Level 1 theme	The name of the level-1 folder in which the table resides. **Note** Level-1 and level-2 folders show the table locations in DataWorks for you to manage tables more conveniently.
Level 2 theme	The name of the level-2 folder in which the table resides.
Create a theme	Click **Create a theme** to go to the **Folder Management** tab. On this tab, you can create level-1 and level-2 folders.
Refresh	After you create a folder, click **Refresh**.
Description	The description of the table.

6. In the **Physical model design** section, set the parameters as required.



Parameter	Description
Layer	Select the appropriate level and category from the drop-down list. To add levels and categories, contact the workspace administrator to click **Create a level** to go to the **Level Management** tab. After you create levels and categories, click **Refresh**.
Physical classification	
Partition type	Valid values: **Partition table** and **Non-partitioned table**.
Table type	Valid values: **Internal tables** and **External tables**.

7. In the **Table structure design** section, set the parameters as required.



Parameter or button	Description
Add fields	To add a field, click **Add fields**, configure the field information, and then click **Save**.
Move up	Adjust the sequence of fields in the table to be created. If you adjust the sequence of fields in an existing table, you are required to delete the table and create another table with the same name. These operations are forbidden in the production environment.
Move down	
Field name	The name of the field, which can contain letters, digits, and underscores (_).
Field type	The EMR table supports the following data types: TINYINT, SMALLINT, INT, BIGINT, FLOAT, DOUBLE, DECIMAL, VARCHAR, CHAR, STRING, BINARY, DATETIME, DATE, TIMESTAMP, BOOLEAN, ARRAY, MAP, and STRUCT.
Length/Settings	You must set this parameter if the data type that you specify for the field has a length limit.
Description	The description of the field.
Primary key	Specifies whether the field serves as the primary key. The primary key is a business concept and ensures that a record is unique for your business. DataWorks does not impose a limit on the primary key.
Edit	After you save a field, you can click **Edit** to edit the field and then click **Save**.
Delete	Delete a created field. **Note** If you delete a field from an existing table and then commit the table, you are required to delete the table and create another table with the same name. This operation is forbidden in the production environment.
Add	If you set the **Partition type** parameter to **Partition table** in the **Physical model design** section, you must configure a partition for the table. You can add a partition to the current table. If you add a partition to an existing table, you are required to delete the table and create another table with the same name. This operation is forbidden in the production environment.

8. Click the icon in the top toolbar to commit the EMR table to the production environment.

   If you are using a workspace in standard mode, commit the table to the development environment and the production environment in sequence.

> **Notice**
>
> - You cannot create an EMR table in data definition language (DDL) mode.
> - You must select a resource group for scheduling when you commit the EMR JAR resource. We recommend that you use an exclusive resource group for scheduling. If no exclusive resource groups for scheduling are available, you can purchase and configure one. For more information, see Create and use an exclusive resource group for scheduling.

4.2. View tenant tables

DataWorks allows you to view MaxCompute, AnalyticDB for PostgreSQL, Hologres, and E-MapReduce (EMR) tables in the workspaces of a tenant account. You can filter tenant tables by the type and name of the compute engine on the Tenant Tables tab.

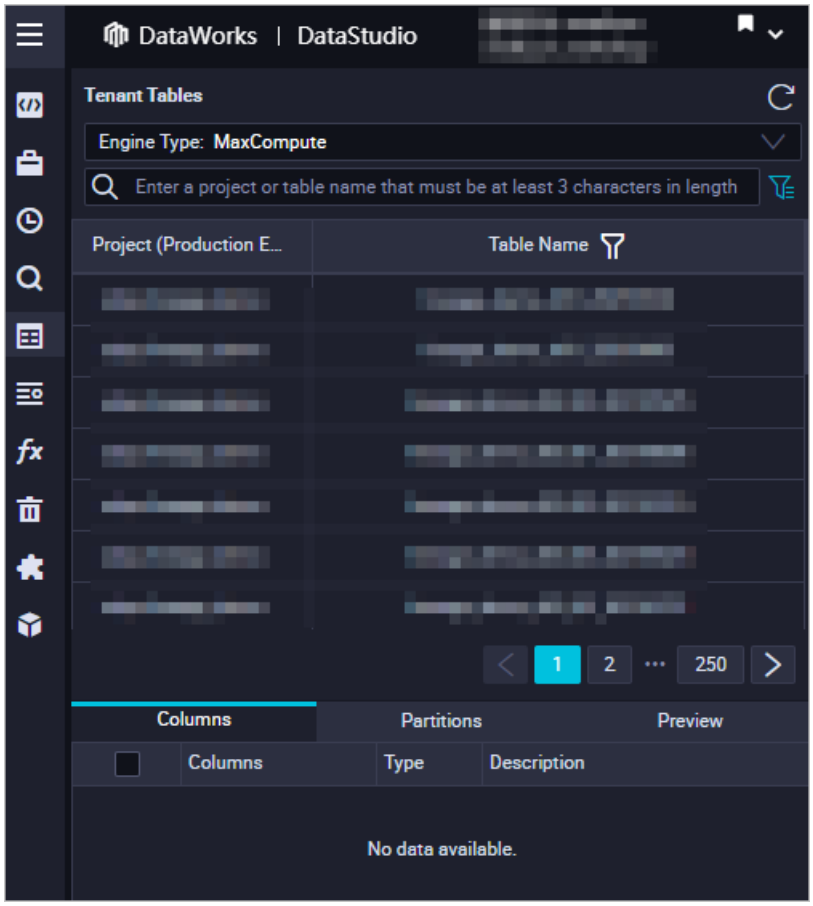
Prerequisites

- To view MaxCompute tenant tables, you must associate a MaxCompute compute engine instance with the current workspace. For more information, see Configure a workspace.
- To view AnalyticDB for PostgreSQL tenant tables, you must associate an AnalyticDB for PostgreSQL compute engine instance with the current workspace and collect the metadata of the AnalyticDB for PostgreSQL instance on the **DataMap** page in the DataWorks console. For more information, see Configure a workspace and Collect metadata from an AnalyticDB for PostgreSQL data source.
- To view Hologres tenant tables, you must associate a Hologres compute engine instance with the current workspace and collect the metadata of the Hologres instance on the **DataMap** page in the DataWorks console. For more information, see Configure a workspace.
- To view E-MapReduce (EMR) tenant tables, you must associate an E-MapReduce compute engine instance with the current workspace and collect the metadata of the E-MapReduce instance on the **DataMap** page in the DataWorks console. For more information, see Configure a workspace and Collect metadata from an EMR data source.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. In the left-side navigation pane, click **Tenant Tables**.

   Click the icon in the lower-left corner to show or hide the left-side navigation pane.
3. Specify the type and name of the compute engine to view the tenant tables.



The following table describes the parameters related to a MaxCompute instance.

Parameter	Description
Project	The name of the project in the related environment. You can click the icon next to the project or table name search box and select an environment. **Note** ◦ For a workspace in standard mode, tenant tables in both the **development** and **production** environments are displayed. For a workspace in basic mode, only tenant tables in the **production environment** are displayed. ◦ The current environment is highlighted in blue.
Table Name	The name of the table in the related environment.
Columns	The information of fields in the current table, including the number of fields, field type, and field description.

Parameter	Description
Partitions	The information and number of partitions of the current table. A maximum of 60,000 partitions can be created. If you have specified the time-to-live (TTL) for partitions, the number of partitions depends on the TTL. **Notice** The partition information is displayed only for MaxCompute tenant tables.
Preview	This parameter allows you to preview the data in the current table. **Notice** ○ Only the data of MaxCompute tenant tables can be previewed. ○ External tables do not support data preview.

4.3. External table

This topic describes how to use DataWorks to create and configure external tables. This topic also lists the data types supported in external tables.

Concepts

The following table lists the concepts you need to know before using external tables.

Concept	Description
Object Storage Service (OSS)	OSS supports storage classes including standard, infrequent access, and archive. It is applicable to various data storage scenarios. In addition, OSS can deeply integrate with the Hadoop open-source community and other products such as E-MapReduce (EMR), Batch Compute, MaxCompute, Machine Learning Platform for AI, and Function Compute.
MaxCompute	MaxCompute is an efficient and fully managed data warehousing solution. When used in conjunction with OSS, it enables you to analyze and process large amounts of data with reduced costs.
External tables of MaxCompute	Based on the computing framework V2.0 of MaxCompute, you can use external tables to directly query numerous files in OSS without loading data into MaxCompute tables. This reduces the time and labor required for data migration and lowers the storage costs.

Currently, MaxCompute supports external tables that store unstructured data, such as the tables in OSS and Table Store. The unstructured data processing framework allows MaxCompute to import data from and export data to external tables in OSS and Table Store. The following section takes the OSS external tables as an example to describe the processing logic:

1. Data stored in OSS is converted through the unstructured data processing framework and passed to user-defined interfaces using the InputStream Java class. You need to write code for the EXTRACT logic, that is, to read, parse, convert, and compute data from input streams. The format

of extracted data records must be supported by MaxCompute.

2. The extracted data records can be further processed by the SQL compute engine built in MaxCompute. More data records can be produced during the processing.
3. The produced records are passed to the user-defined output logic for further computing and conversion. Finally, the system uses the OutputStream Java class to export required data in the records to OSS.

You can create, search for, configure, process, and configure external tables by using a visual interface in the DataWorks console. You can also query, compute, and analyze data in external tables. DataWorks is powered by MaxCompute.

Network and access authorization

Since MaxCompute is separate from OSS, network connectivity between them on different clusters may affect the ability of MaxCompute to access the data stored in OSS. We recommend that you use the private endpoint which ends with -internal.aliyuncs.com when you access the data stored in OSS from MaxCompute.

MaxCompute requires a secure authorized channel to access OSS data. MaxCompute uses Resource Access Management (RAM) and Security Token Service (STS) of Alibaba Cloud to ensure data access security. MaxCompute applies for data access permissions from STS as the table creator. The permission settings of Table Store are the same as those of OSS.

1. Perform STS authorization.

   You need to authorize the account used for running MaxCompute jobs to access OSS data. STS is a token management service provided by the RAM service of Alibaba Cloud. With STS, authorized RAM users and cloud services can issue tokens with custom validity and permissions. Applications can use tokens to call Alibaba Cloud API operations to manipulate resources.

   You can grant the permissions by using either of the following two methods:

   - If the MaxCompute project and OSS bucket are owned by the same Alibaba Cloud account, log on to the DataWorks console and perform authorization.
     a. Open the editing page of a newly created table and find the **Physical Model** section.
     b. Set **Table Type** to **External Table**.
     c. Configure **Storage Address** and click **Authorize**.
     d. In the **Cloud Resource Access Authorization** dialog box, click **Confirm Authorization Policy**.
   - Grant MaxCompute the permission to access OSS data in the RAM console.
     a. Log on to the RAM console.

     > **Note** If you use different accounts for logon to MaxCompute and OSS, you must log on to the RAM console using the OSS account to perform the following operations.

     b. In the left-side navigation pane, click **RAM Roles**.
     c. In the **Create Role** panel, select **Alibaba Cloud Account** for the Select Trusted Entity parameter and click **Next**.

d. Configure **RAM Role Name** and **Note**.

> **Note** The role name must be set to AliyunODPSDefaultRole or AliyunODPSRoleForOtherUser.

e. Under **Select Trusted Alibaba Cloud Account**, select **Current Alibaba Cloud Account** or **Other Alibaba Cloud Account**.

> **Note** If you select **Other Alibaba Cloud Account**, enter the account ID.

f. Click **OK**.

g. Configure the **role details**.

On the **RAM Roles** page, click the target role in the **RAM Role Name** column. On the **Trusted Policy Management** tab, click **Edit Trust Policy** and enter a policy as required.

```
--If the MaxCompute project and OSS bucket belong to the same account, enter the
following content:
{
"Statement": [
{
"Action": "sts:AssumeRole",
"Effect": "Allow",
"Principal": {
"Service": [
"odps.aliyuncs.com"
      ]
    }
  }
],
"Version": "1"
}
```

```
--If the MaxCompute project and OSS bucket belong to different accounts, enter th
e following content:
{
"Statement": [
{
"Action": "sts:AssumeRole",
"Effect": "Allow",
"Principal": {
"Service": [
"ID of the Alibaba Cloud account that owns the MaxCompute project@odps.aliyuncs.c
om"
      ]
    }
  }
],
"Version": "1"
}
```

After the configuration is completed, click **OK**.

h. Associate an authorization policy with the role. Click **Add Permissions** and search for the AliyunODPSRolePolicy policy that is required for granting OSS access. Attach the AliyunODPSRolePolicy policy to the role. If you cannot find this policy in this way, click **Input and Attach** to attach the required permission to the role.

```
{
  "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. Use an OSS data store.

   If you have created and saved an OSS data store, view and use the data store under **Workspace Manage > Data Source**.

Create an external table

1. Use Data Definition Language (DDL) to create an external table.

   Go to the **Data Analytics** page and use a DDL statement to create an external table. The DDL statement must comply with the MaxCompute syntax. If you have completed STS authorization, you do not need to include the odps.properties.rolearn attribute in the DDL statement.

   The following section provides a sample DDL statement, in which EXTERNAL indicates an external table:

```
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' --Required. The STORED
BY statement specifies the name of a custom storage handler class or other file format.
with SERDEPROPERTIES (
'delimiter'='\\|', -- Optional. The SERDEPROPERTIES clause specifies the parameters use
d when serializing or deserializing data. These parameters can be passed into the EXTRA
CT logic through DataAttributes.
'odps.properties.rolearn'='acs:ram::xxxxxxxxxxxx:role/aliyunodpsdefaultrole'
)
LOCATION 'oss://oss-cn-shanghai-internal.aliyuncs.com/oss-odps-test/Demo/SampleData/Cus
tomTxt/AmbulanceData/'     --Required. The LOCATION parameter specifies the location of
the external table.
USING 'odps-udf-example.jar'; --Required if you use a custom format class. The USING pa
rameter specifies the JAR package where the custom format class resides.
```

STORED BY can be followed by a parameter indicating the storage handler built in for the Comma Separated Value (CSV) or Tab Separated Value (TSV) file. The details are as follows:

- `com.aliyun.odps.CsvStorageHandler` indicates that the table is stored as a CSV file. With this specification, the column delimiter is a comma (,) and the newline character is \n. One example is `STORED BY'com.aliyun.odps.CsvStorageHandler'` .
- `com.aliyun.odps.TsvStorageHandler` indicates that the table is stored as a TSV file. With this specification, the column delimiter is \t and the newline character is \n.

STORED BY can also be followed by a parameter indicating an **open-source external table**, such as ORC, Parquet, sequence file, Record Columnar File (RCFile), Avro, or text file. For example, you can specify the `org.apache.hive.hcatalog.data.JsonSerDe` class to save the table as a text file.

- org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe: The table is stored as a text file.
- org.apache.hadoop.hive.ql.io.orc.OrcSerde: The table is stored in the Optimized Row Columnar (ORC) format.
- org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe: The table is stored in the Parquet format.
- org.apache.hadoop.hive.serde2.avro.AvroSerDe: The table is stored in the Avro format.
- org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe: The table is stored as a sequence file.

The following section provides a DDL statement for creating an open-source external table:

```
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}/';
```

The following table lists attributes of the **SERDEPROPERTIES** clause.

Attribute	Valid value	Default value	Description
odps.text.option.gzip.input.enabled	true or false	false	Indicates whether the file to be read is compressed.
odps.text.option.gzip.output.enabled	true or false	false	Indicates whether the file to be written is compressed.
odps.text.option.header.lines.count	N (a non-negative integer)	0	Skip the first N lines of the file.
odps.text.option.null.indicator	String	Empty string	The string that represents NULL.
odps.text.option.ignore.empty.lines	true or false	true	Specifies whether to ignore blank lines.
odps.text.option.encoding	UTF-8, UTF-16, or US-ASCII	UTF-8	The encoding of the file.

> **Note** MaxCompute can also read CSV and TSV files in GZIP format by using a built-in extracter. You can select whether the file is GZIP compressed, which determines attributes you need to specify.

The LOCATION parameter is in the *oss://oss-cn-shanghai-internal.aliyuncs.com/Bucket name/Directory name* format. You can obtain the OSS directory in the graphical user interface (GUI). You do not need to add a filename next to the directory.

You can find tables created by using DDL statements in the node tree on the Workspace Tables tab. You can modify the values of level 1 folder and level 2 folder to change the table locations.

2. Create a Table Store external table.

   The following section describes a statement used to create a Table Store external table:

```
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';
```

The parameters in the statement are described as follows:

- com.aliyun.odps.TableStoreStorageHandler specifies the storage handler built in MaxCompute for processing data stored in Table Store.
- SERDEPROPERTIES provides options for parameters. You must specify tablestore.columns.mapping and tablestore.table.name when using TableStoreStorageHandler.
  - tablestore.columns.mapping: Required. This parameter describes the columns of the table in Table Store that MaxCompute accesses, including the primary key columns and property columns. A primary key column is indicated with the colon sign (:) at the beginning of the column name. In this example, primary key columns are `:o_orderkey` and `:o_orderdatee`. The others are property columns.

    Table Store supports up to four primary key columns. The data types include STRING, INTEGER, and BINARY. The first primary key column is the partition key. You must specify all primary key columns of the table in Table Store when specifying the mapping. You only need to specify the property columns that MaxCompute accesses instead of specifying all property columns.
  - tablestore.table.name: the name of the table to access in Table Store. If the table name does not exist in Table Store, an error is returned. MaxCompute does not create a table in Table Store.
- LOCATION: specifies the name and endpoint of the Table Store instance.

3. Create a table in the GUI.

   Go to the **Data Analytics** page and create a table in the GUI. An external table has the following attributes:

   - **General**
     - Table name (entered when you create a table)
     - Display name
     - Level 1 folder and level 2 folder
     - Description
   - **Physical Model**
     - **Table Type**: Select **External Table**.
     - **Partitioning**: Table Store external tables do not support partitioning.

- **Storage Address**: In the **Physical Model** section, you can enter the value of the LOCATION parameter as the storage address. You can also click **Click to Select** to select a storage address. Then, click **Authorize**.
- **Storage Format**: Select the file format as required. CSV, TSV, ORC, Parquet, sequence file, RCFile, Avro, text file, and custom file formats are supported. If you select a custom file format, you need to select the corresponding resource. After you commit a resource, DataWorks automatically parses out included class names and displays them in the Class Name drop-down list.
- rolearn: If you have completed STS authorization, you can leave it unspecified.

- **Schema**

Parameter	Description
Field Type	MaxCompute 2.0 supports fields of TINYINT, SMALLINT, INT, BIGINT, VARCHAR, and STRING types.
Actions	The add, change, and delete actions are supported.
Definition or Maximum Value Length	You can set the maximum length of the VARCHAR type fields. For complex data types, enter the definition.

Data types

The following table describes the data types supported in external tables.

Type	New	Example	Description
TINYINT	Yes	1Y and -127Y	A signed eight-bit integer in the range of -128 to 127.
SMALLINT	Yes	32767S and -100S	A signed 16-bit integer in the range of -32768 to 32767.
INT	Yes	1000 and -15645787	A signed 32-bit integer in the range of -2~31 to 2~31 - 1.
BIGINT	No	100000000000L and -1L	A signed 64-bit integer in the range of -2~63 + 1 to 2~63 - 1.

Type	New	Example	Description
FLOAT	Yes	None.	The 32-bit binary floating point type.
DOUBLE	No	3.1415926 1E+7	An eight-byte double precision floating-point number (a 64-bit binary floating-point number).
DECIMAL	No	3.5BD and 99999999999.9999999BD	A decimal exact number. The integer part is in the range of -10~36 + 1 to 10~36 - 1, and the fractional part is accurate to 10 to 18 decimal places.
VARCHAR(n)	Yes	None.	A variable-length character string, which can be 1 to 65535 characters in length.
STRING	No	"abc", 'bcd', and "alibaba"	A string. Currently, the maximum length is 8 MB.
BINARY	Yes	None.	A binary number. Currently, the maximum length is 8 MB.
DATETIME	No	DATETIME '2017-11-11 00:00:00'	The data type for dates and times. UTC-8 is used as the standard time of the system. The range is from 0000-01-01 to 9999-12-31, accurate to milliseconds.
TIMESTAMP	Yes	TIMESTAMP '2017-11-11 00:00:00.123456789'	The TIMESTAMP data type, which is independent from time zones. The range is from 0000-01-01 to 9999-12-31 23.59:59.999,999,999, accurate to nanoseconds.
BOOLEAN	No	TRUE and FALSE	The value must be TRUE or FALSE.

The following table lists complex field types supported in external tables.

Type	Definition	Constructor
ARRAY	array< int >; array< struct< a:int, b:string >>	array(1, 2, 3); array(array(1, 2); array(3, 4))
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))

If you want to use data types newly supported by MaxCompute 2.0 including TINYINT, SMALLINT, INT, FLOAT, VARCHAR, TIMESTAMP, BINARY, and complex data types, you must include `set odps.sql.type.system.odps2=true;` before the statement used to create a table. Commit the set statement together with the table creation statement. If compatibility with Hive is required, we recommend that you include the `odps.sql.hive.compatible=true;` statement.

View and process external tables

Go to the **Data Analytics** page and click **Workspace Tables** in the left-side navigation pane to view external tables. External tables are processed in the same way as internal tables.

4.4. Manage tables

DataWorks allows you to view, modify, and delete your MaxCompute, AnalyticDB for PostgreSQL, and E-MapReduce (EMR) tables. You can view and manage these tables by type on the Workspace Tables tab of the DataStudio page.

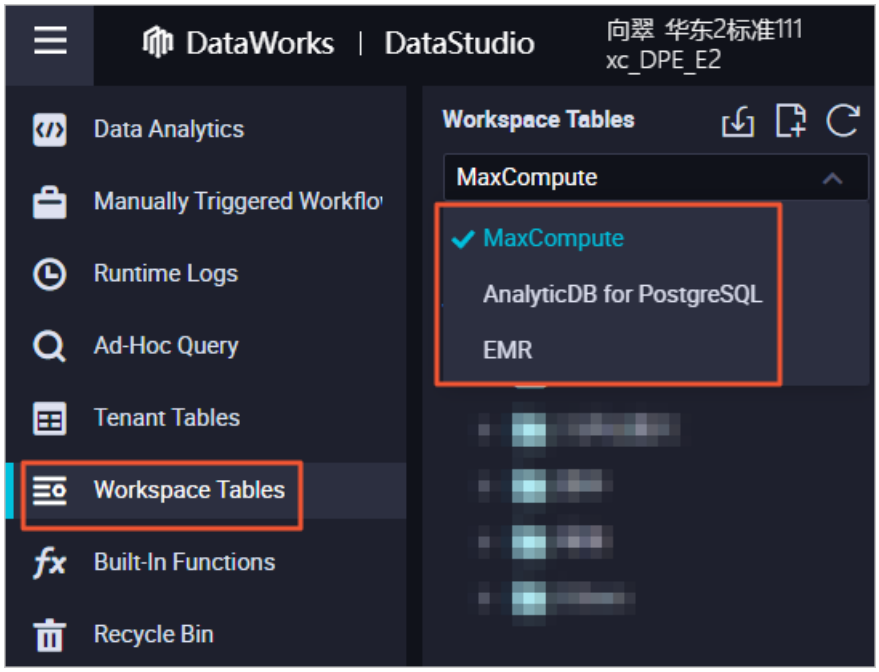
Prerequisites

- To manage MaxCompute tables, you must associate a MaxCompute compute engine instance with your workspace. For more information, see Configure a workspace.
- To manage AnalyticDB for PostgreSQL tables, you must associate an AnalyticDB for PostgreSQL compute engine instance with your workspace and collect the metadata of the AnalyticDB for PostgreSQL compute engine instance on the **DataMap** page in the DataWorks console. For more information, see Configure a workspace and Collect metadata from an AnalyticDB for PostgreSQL data source.
- To manage EMR tables, you must associate an EMR compute engine instance with your workspace and collect the metadata of the EMR compute engine instance on the **DataMap** page in the DataWorks console. For more information, see Configure a workspace and Collect metadata from an EMR data source.

Manage tables

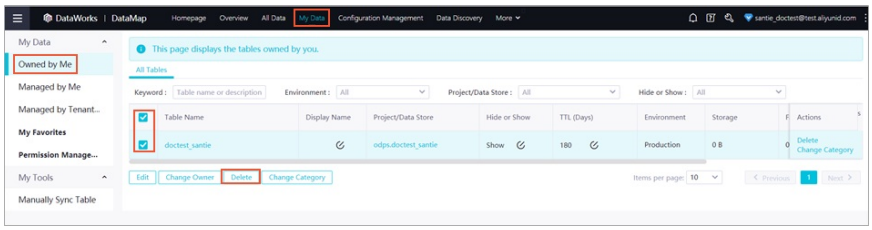
1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. In the left-side navigation pane, click the **Workspace Tables** icon.

   You can click the icon in the lower-left corner to show or hide the names of tabs in the left-side navigation pane.
3. Select an engine type from the drop-down list in the upper part of the Workspace Tables tab.



In this example, MaxCompute is selected. The following table describes how to view, modify, and delete a MaxCompute table. For more information about how to create a table, see Create a MaxCompute table.

Operation	Description
View a table	You can click the icon next to the search box and select an environment. **Note** ◦ For a workspace in standard mode, tables in both the development and production environments are displayed. For a workspace in basic mode, only tables in the production environment are displayed. ◦ The current environment is highlighted in blue. Double-click a table. The configuration tab of the table appears and displays the details of the table.
Rename a table	In a workspace in standard mode, right-click a table and select **Rename Table**. In the **Rename Table** dialog box, enter a name and click **OK**. **Note** You cannot right-click a table in the production environment and rename the table. Therefore, tables in workspaces in basic mode do not support this operation.
Import data to a table	Right-click a table and select **Import Data**. For more information, see Create tables and import data.

Operation	Description
Delete a table	You can delete a table in either the development or production environment. ○ Delete a table in the development environment: a. Before you delete a table, make sure that you have permissions to delete the table. Right-click a table and select **Delete Table**. b. In the **Delete Table** dialog box, select the **I have been aware of the risk and confirm the deletion** check box. c. Click **OK**. **Note** You cannot right-click a table in the production environment and delete the table. Therefore, tables in workspaces in basic mode do not support this operation. If the name of a table starts with tmp_pyodps, the table is of the PyODPS type. For more information, see Create a PyODPS 2 node. ○ Delete a table in the production environment: a. Method 1: Create an ODPS SQL node, enter the command that is used to delete the table, and then commit and deploy the node to the production environment to run the node. For more information, see Create an ODPS SQL node. b. Method 2: Delete the table on the My Data tab of the DataMap page. 

Divide a data warehouse into layers

In the **Physical Model** section of the configuration tab of a table, you can define table layers for a data warehouse. This enables you to have better planning and control over your data.

In most cases, a data warehouse consists of the following layers:

- The ODS layer stores raw data in the data warehouse. The data structure is basically consistent with that in the source system. The ODS layer serves as the data staging area of the data warehouse. It imports basic data to MaxCompute and records historical changes of basic data.
- The CDM layer, which is also called the general data model layer, consists of the dimension data (DIM), data warehouse detail (DWD), and data warehouse service (DWS) layers. The CDM layer processes and integrates the data of the ODS layer to define conformed dimensions, create reusable detailed fact tables for analysis and statistics, and aggregate common metrics.
  - The DIM layer defines conformed dimensions for an enterprise based on the concepts of dimensional modeling. It reduces the risk of inconsistent statistical criteria and algorithms.

    Tables at the DIM layer are also called logical dimension tables. Generally, each dimension corresponds to a logical dimension table.

- The DWS layer is driven by analyzed subjects during data modeling. Based on the metric requirements of upper-layer applications and products, the DWS layer creates fact tables to aggregate common metrics and builds a physical data model by using wide tables. The DWS layer creates statistical metrics in compliance with uniform naming conventions and statistical criteria, provides common metrics for the upper layer, and generates aggregate wide tables and detailed fact tables.

  Tables at the DWS layer are also called logical aggregate tables, which are used to store derived metrics.

- The DWD layer is driven by business processes during data modeling. It creates detailed fact tables at the finest granularity based on each specific business process. In combination with the data usage habits of an enterprise, you can duplicate some key attribute fields of dimensions in detailed fact tables to create wide tables.

  Tables at the DWD layer are also called logical fact tables.

- The ADS layer stores personalized statistical metrics of data products. It processes the data of the CDM and ODS layers.

5.Create and manage nodes

5.1. Create a batch sync node

Batch sync nodes support various types of data stores, including MaxCompute, MySQL, Distributed Relational Database Service (DRDS), SQL Server, PostgreSQL, Oracle, MongoDB, Db2, Tablestore, Object Storage Service (OSS), FTP, HBase, LogHub, Hadoop Distributed File System (HDFS), and Stream. This topic describes how to create a batch sync node.

Context

When you enter a table name, an object list appears, which displays all matched tables. Only exact match is supported. Therefore, you must enter a complete table name. Tables are labeled as unsupported if they are not supported by batch sync nodes.

If you move the pointer over a table in the list, the details of the table are displayed, including the database, IP address, and owner of the table. The details help you select the correct table. After you select and click a table, the column information is automatically entered. You can add, move, or delete columns.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Move the pointer over the **+Create** icon and choose **Data Integration > Batch Synchronization**.

   Alternatively, you can find the required workflow, right-click **Data Integration**, and then choose **Create > Batch Synchronization**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. Configure the batch sync node. For more information, see Configure a synchronization node by using the codeless UI.
6. Commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iii. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the do-while node.

7. Test the node. For more information, see View auto triggered nodes.

5.2. Create, configure, commit, and manage real-time sync nodes

DataWorks supports real-time synchronization. This topic describes how to create, configure, commit, and manage real-time sync nodes.

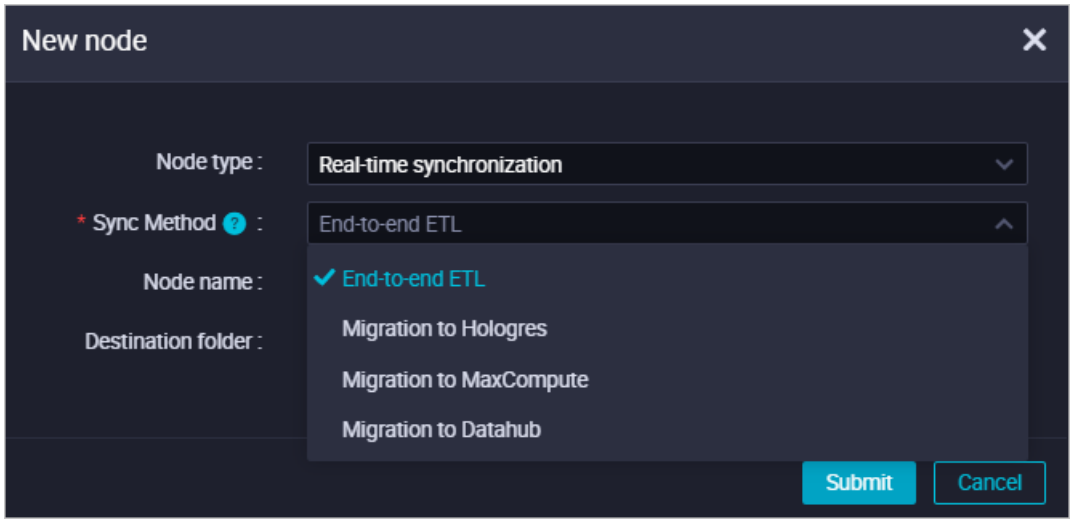
Prerequisites

The real-time synchronization feature is in public preview. This feature is available in the following regions: China (Hangzhou), China (Shanghai), China (Beijing), China (Zhangjiakou), China (Shenzhen), and China (Chengdu).

Create a real-time sync node

1. Log on to the DataWorks console.
2. In the left-side navigation pane, click **Workspaces**.
3. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **Data Development** in the Actions column.
4. On the Data Development tab, move the pointer over the **+Create** icon and choose **Data integration > Real-time synchronization**.

   Alternatively, you can click a workflow in the Business process section, right-click **Data Integration**, and then choose **New > Real-time synchronization**. For more information about the data stores that the real-time synchronization feature supports, see Plug-ins for data sources that support real-time synchronization.
5. In the **New node** dialog box, set the parameters as required.



Parameter	Description
Node type	The type of the node. Default value: **Real-time synchronization**.

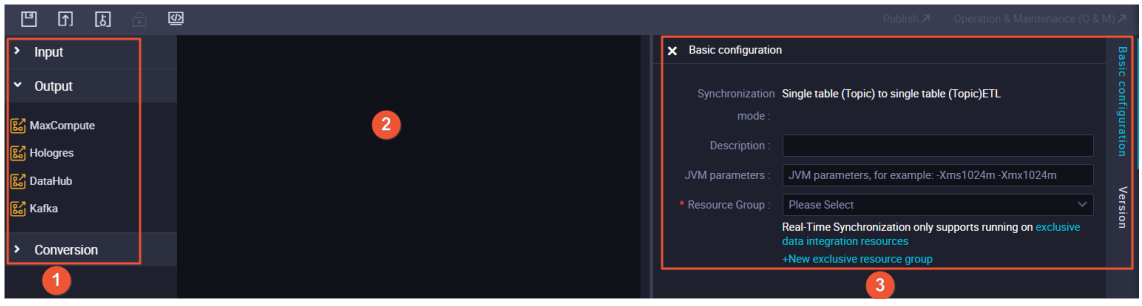
Parameter	Description
Sync Method	The method for synchronizing data. Valid values: ○ **End-to-end ETL**: synchronizes data in one table to one or more tables. Data transformation is supported during the synchronization process. ○ **Migration to Hologres**: synchronizes all or some tables in a database to Hologres. Destination tables can be automatically created in Hologres. ○ **Migration to MaxCompute**: synchronizes all or some tables in a database to MaxCompute. ○ **Migration to DataHub**: synchronizes all or some topics in a database to DataHub.
Node name	The name of the node. The node name must be 1 to 128 characters in length, and can contain letters, digits, underscores (_), and periods (.).
Destination folder	The folder where the node resides.

6. Click **Submit**.

Configure the real-time sync node

The operations that you can perform on the configuration tab of the real-time sync node vary based on the synchronization method you selected.

- To configure the real-time sync node for which **Sync Method** is set to **End-to-end ETL**, perform the following steps:
  i. Double-click the real-time sync node. On the node configuration tab that appears, click the **Basic configuration** tab in the right-side navigation pane. On the Basic configuration tab, select the desired resource group from the **Resource Group** drop-down list.

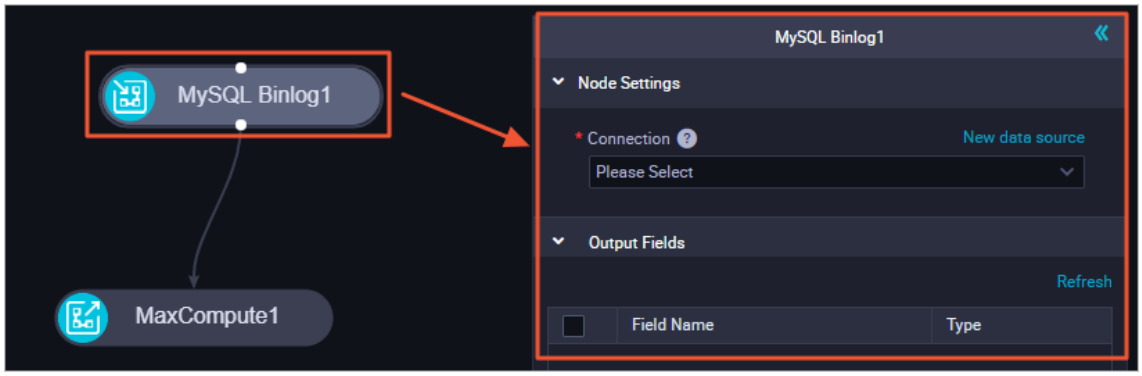


No.	Description
1	The left-side navigation tree. This pane consists of the **Input**, **Output**, and **Conversion** sections.
2	The configuration canvas of the real-time sync node. You can drag components from the navigation tree to the canvas.

No.	Description
3	The property configuration pane of the real-time sync node. This pane appears after you click a node on the canvas or click the **Basic configuration** tab in the right-side navigation pane. **Notice** You must select a resource group before you commit the node. Otherwise, the system returns an error when you commit the node. Real-time sync nodes can be run only on an exclusive resource group for Data Integration. For more information, see Create and use an exclusive resource group for Data Integration.

ii. Drag components from the navigation tree to the canvas, and drag directed lines to connect the nodes on the canvas. Data will be synchronized from upstream nodes to downstream nodes based on the connection.

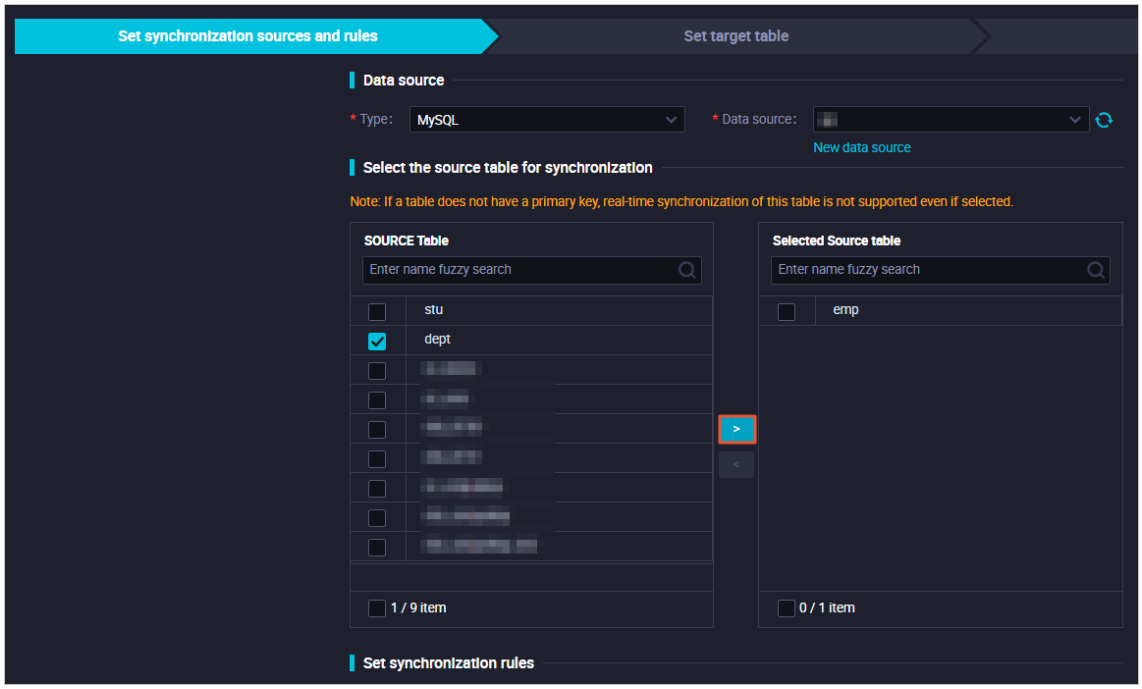
iii. Click each node. In the configuration pane that appears, set the required parameters in the **Node configuration** section. For more information, see Supported data stores.



iv. Click the icon in the toolbar.

- To configure the real-time sync node for which **Sync Method** is set to **Migration to Hologres**, perform the following steps:

  i. Double-click the real-time sync node. On the node configuration tab that appears, click the **Basic configuration** tab in the right-side navigation pane. On the Basic configuration tab, select the desired resource group from the **Resource Group** drop-down list.



> **Notice** You must select a resource group before you commit the node. Otherwise, the system returns an error when you commit the node. Real-time sync nodes can be run only on an exclusive resource group for Data Integration. For more information, see Create and use an exclusive resource group for Data Integration.

ii. In the **Data source** section, set the **Type** and **Data source** parameters.

iii. In the **Select the source table for synchronization** section, select the tables to be synchronized in the **SOURCE Table** list and click the > icon to move the tables to the **Selected Source table** list.

The SOURCE Table list displays all the tables in the source database. You can select all or some tables to synchronize them at a time.

> **Notice** If a selected table does not have a primary key, the table cannot be synchronized in real time.

iv. (Optional)In the **Set synchronization rules** section, click **Add rule** and select an option to configure naming rules for destination tables.

Supported options include **Table name conversion rules** and **Target table name rule**.

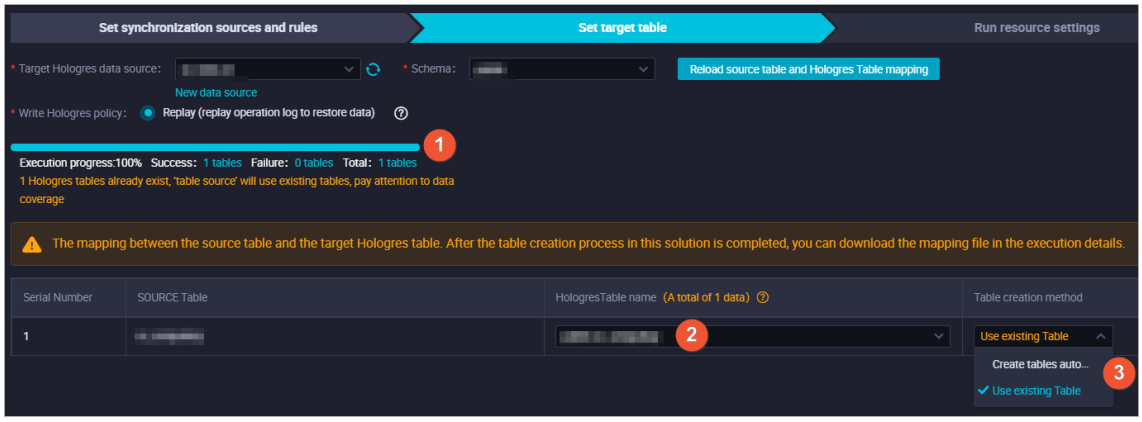
- **Table name conversion rules**: the rules for converting the names of source tables to that of destination tables.
- **Target table name rule**: the rule for adding a prefix and suffix to the converted names of destination tables.

v. Click **Next Step**.

vi. In the **Set target table** step, set the **Target Hologres data source** and **Schema** parameters.

vii. Click **Reload source table and Hologres Table mapping** to configure the mappings between the source tables and destination Hologres tables.

viii. Check the source and destination tables after the mappings are created, and click **Next Step**.



No.	Description
1	The mapping progress between the source and destination tables. **Note** The mapping may take a long period of time if the number of source tables to be synchronized is large.
2	The destination tables to which data is written. The tables can be existing ones or the ones that are automatically created. **Note** An error message appears if the selected source table does not have a primary key. The synchronization can be performed if one of the selected source tables has a primary key. Source tables without primary keys are ignored during the synchronization.
3	The method of creating a destination table. The message that appears in the **Hologres Table name** column varies depending on the method that you select. ■ If you select **Create tables automatically**, the **Create tables automatically** dialog box appears after you click **Next Step**. Click **Start table building** in the dialog box, and then click **Close** after the table is created. You can click the table name to view and modify the table creation statements. ■ If you select **Use existing Table**, you must select a table from the drop-down list in the **Hologres Table name** column.

ix. In the **Run resource settings** step, set the **Maximum number of connections supported by source read** and **Number of concurrent writes on the target side** parameters and then click the icon in the toolbar.

- To configure the real-time sync node for which **Sync Method** is set to **Migration to MaxCompute**, perform the following steps:
  i. Double-click the real-time sync node. On the node configuration tab that appears, click the **Basic configuration** tab in the right-side navigation pane. On the Basic configuration tab, select the desired resource group from the **Resource Group** drop-down list.

ii. In the **Data source** section, set the **Type** and **Data source** parameters.

iii. In the **Select the source table for synchronization** section, select the tables to be synchronized in the **SOURCE Table** list and click the > icon to move the tables to the **Selected Source table** list.

The SOURCE Table list displays all the tables in the source database. You can select all or some tables to synchronize them at a time.

> **Notice** If a selected table does not have a primary key, the table cannot be synchronized in real time.

iv. In the **Set synchronization rules** section, click **Add rule** and select an option to configure naming rules for destination tables.

Supported options include **Table name conversion rules** and **Target table name rule**.

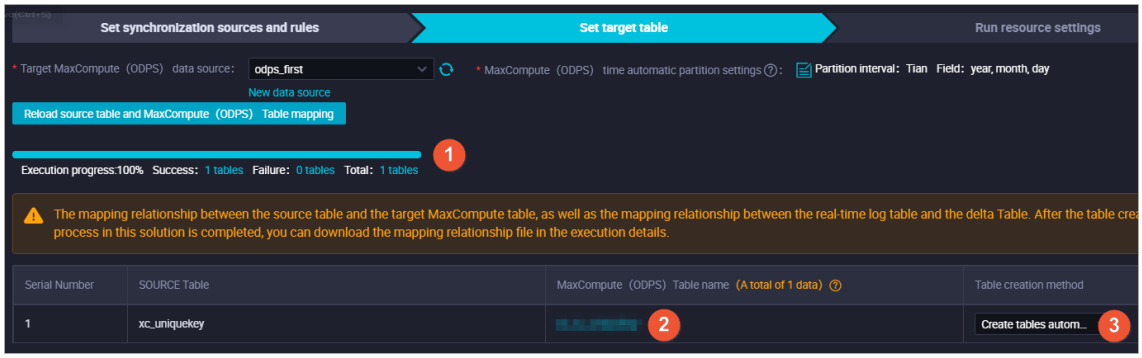
- **Table name conversion rules**: the rules for converting the names of source tables to that of destination tables.
- **Target table name rule**: the rule for adding a prefix and suffix to the converted names of destination tables.

v. Click **Next Step**.

vi. In the **Set target table** step, select a connection from the **Target MaxCompute (ODPS) data source** drop-down list and click the icon next to **MaxCompute (ODPS) time automatic partition settings**. In the **Edit** dialog box, set the partition interval of tables in MaxCompute to day or hour.

vii. Click **Reload source table and MaxCompute (ODPS) Table mapping** to configure the mappings between the source tables and destination MaxCompute tables.

viii. Check the source and destination tables after the mappings are created, and click **Next Step**.



No.	Description
1	The mapping progress between the source and destination tables. **Note** The mapping may take a long period of time if the number of source tables to be synchronized is large.

No.	Description
2	The destination tables to which data is written. The tables can be existing ones or the ones that are automatically created. **Note** An error message appears if the selected source table does not have a primary key. The synchronization can be performed if one of the selected source tables has a primary key. Source tables without primary keys are ignored during the synchronization.
3	The method of creating a destination table. The message that appears in the **MaxCompute (ODPS) Table name** column varies depending on the method that you select. ■ If you select **Create tables automatically**, the **Create tables automatically** dialog box appears after you click **Next Step**. Click **Start table building** in the dialog box, and then click **Close** after the table is created. You can click the table name to view and modify the table creation statements. ■ If you select **Use existing Table**, you must select a table from the drop-down list in the **MaxCompute (ODPS) Table name** column.

ix. In the **Run resource settings** step, set the **Maximum number of connections supported by source read** and **Number of concurrent writes on the target side** parameters and then click the icon in the toolbar.

- To configure the real-time sync node for which **Sync Method** is set to **Migration to DataHub**, perform the following steps:
  i. Double-click the real-time sync node. On the node configuration tab that appears, click the **Basic configuration** tab in the right-side navigation pane. On the Basic configuration tab, select the desired resource group from the **Resource Group** drop-down list.
  ii. In the **Data source** section, set the **Type** and **Data source** parameters.
  iii. In the **Select the source table for synchronization** section, select the tables to be synchronized in the **SOURCE Table** list and click the icon to move the tables to the **Selected Source table** list.

  The SOURCE Table list displays all the tables in the source database. You can select all or some tables to synchronize them at a time.

  Notice If a selected table does not have a primary key, the table cannot be synchronized in real time.

  iv. In the **Set synchronization rules** section, click **Add rule** and then select an option to configure naming rules for destination tables.

  Supported options include **SOURCE table name and Topic conversion rules** and **Target Topic rules**.

  v. Click **Next Step**.
  vi. In the **Set target table** step, select a connection from the **Target DataHub data source**

drop-down list and then click **Reload source table and DataHub Topic mapping** to configure the mappings between the source tables and destination DataHub topics.

vii. Check the source tables and destination topics after the mappings are created, and click **Next Step**.

No.	Description
1	The mapping progress between the source tables and destination topics. **Note** The mapping may take a long period of time if the number of source tables to be synchronized is large.
2	The destination topics to which data is written. The topics can be existing ones or the ones that are automatically created.
3	The method of creating a destination topic. The message that appears in the **Topic** column varies depending on the method that you select. ■ If you select **Create tables automatically**, the **Create tables automatically** dialog box appears after you click **Next Step**. Click **Start table building** in the dialog box, and then click **Close** after the topic is created. ■ If you select **Use existing Topic**, you must select a topic from the drop-down list in the **Topic** column.

viii. In the **Run resource settings** step, set the **Maximum number of connections supported by source read** and **Number of concurrent writes on the target side** parameters and then click the icon in the toolbar.

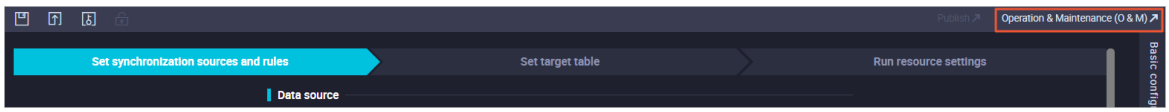
Commit the real-time sync node

1. On the configuration tab of the real-time sync node, click the icon in the toolbar.
2. In the **Submit New version** dialog box, enter your comments in the **Change description** field.
3. Click **OK**.

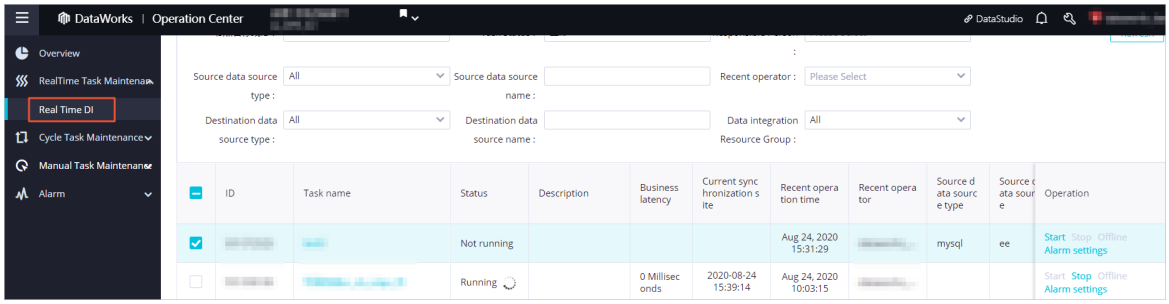
   In a workspace in standard mode, you must click **Publish** in the upper-right corner after you commit the real-time sync node. For more information, see Deploy a node.

Manage the real-time sync node

1. After you commit or deploy the real-time sync node, click **Operation & Maintenance (O & M)** in the upper-right corner of the node configuration tab to manage the node on the **Real Time DI** page.



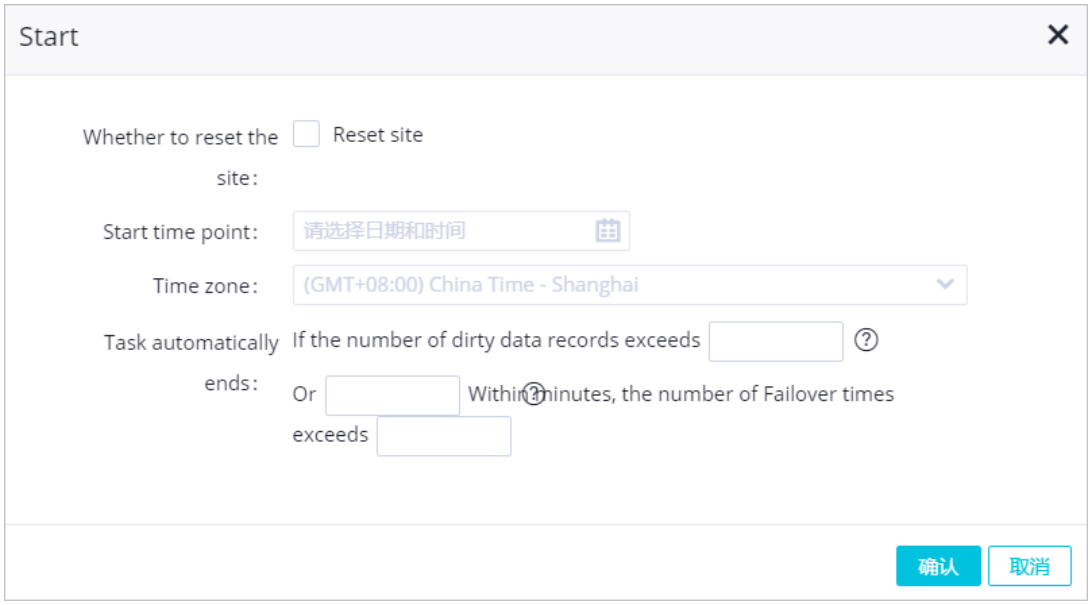
2. On the **Real Time DI** page, find the real-time sync node, click the node name, and then view the O&M details about the node.



On this page, you can start, stop, undeploy, or configure alert settings for the real-time sync node.

- To start a node that is not running, perform the following steps:
  a. Find the node and click **Start** in the Operation column.

b. In the **Start** dialog box, set the parameters as required.



Parameter	Description
Whether to reset the site	Specifies whether to set the time point for next startup. If you select Reset site, the **Start time point** and **Time zone** parameters are required.
Start time point	The date and time for starting the real-time sync node.
Time zone	The time zone where the source data store resides. Select a time zone from the **Time zone** drop-down list.
Task automatically ends	■ The condition for automatically terminating the real-time sync node. You can specify the maximum number of dirty data records allowed. If you set the value to 0, no dirty data records are allowed. If the value is empty, the node continues no matter whether dirty data records exist. ■ You can also specify the maximum number of failover times. If the value is empty, the node is automatically terminated if the node fails for 100 times every 5 minutes. This avoids resource occupation caused by frequent startup.

c. Click **OK**.

- To stop a running node, perform the following steps:
  a. Find the node and click **Stop** in the Operation column.
  b. In the message that appears, click **Stop**.
- To bring offline a node that is not running, perform the following steps:
  a. Find the node and click **Offline** in the Operation column.
  b. In the message that appears, click **Offline**.
- Find the node and click **Alarm settings** in the Operation column. On the page that appears, you

can view alert event information and alert rules on the **Alert event** and **Alarm rules** tabs.

- To configure alert settings for a node, perform the following steps:
  a. Select the node and click **New Alarm** in the lower part of the page.
  b. In the **New rule** dialog box, set the parameters as required.

Parameter	Description
Name	Required. The name of the rule to create.
Description	The description of the rule.
Indicators	The indicators in the rule to create. Valid values: **Task Status**, **Business latency**, **Failover**, **Dirty Data**, and **DDL error**.
Threshold	The threshold for reporting an alert. The default value is 5 minutes for both **WARNING** and **CRITICAL** alerts.
Alarm interval	The interval for reporting alerts. The default value is 5 minutes.
WARNING	The methods for sending alerts. Valid values: **Mail**, **SMS**, and **DingTalk**. **Note** Only Singapore, Malaysia(Kuala Limpur), and Germany(Frankfurt) support the SMS reminding method. To use the SMS reminding method in other regions, submit a ticket to contact DataWorks technical support.
CRITICAL	
Receiver	The person who receives alerts. Select a receiver from the **Receiver** drop-down list.

  c. Click **OK**.

- To modify alert settings for a node, perform the following steps:
  a. Select the node and click **Operation alarm** in the lower part of the page.
  b. In the **Operation alarm** dialog box, set the **Operation type** and **Alarm indicators** parameters.

     DataWorks automatically modifies all the rules for the selected alert types at a time.
  c. Click **OK**.

5.3. MaxCompute

5.3.1. Create an ODPS SQL node

ODPS SQL nodes can process terabytes of data in distributed scenarios that do not require real-time processing by using the SQL-like syntax. This topic describes how to create an ODPS SQL node.

Prerequisites

A MaxCompute compute engine instance is associated with your workspace. The **MaxCompute** service

is available in a workspace only after you associate a **MaxCompute** compute engine instance with the workspace on the **Workspace Management** page. For more information, see Configure a workspace.

Context

In most cases, a long period of time is required from preparation to job committing. To reduce the time required, you can use ODPS SQL nodes to process thousands to tens of thousands of transactions. ODPS SQL nodes are online analytical processing (OLAP) applications that are designed to process large amounts of data..

Limits

ODPS SQL nodes have the following limits:

- You cannot use a SET or USE statement separately in the code of an ODPS SQL node. They must be executed with other SQL statements. For example, you can use a SET statement together with a CREATE TABLE statement.

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

- You cannot add comments to statements that contain keywords in the code of an ODPS SQL node. The statements include the SET and USE statements. For example, a comment is not allowed in the following code:

```
create table name(id string);
set a=b; --Comment   // You cannot add a comment.
create table name1(id string);
```

- You cannot add comments to the end of a complete statement in the code of an ODPS SQL node. Examples:

> **Note** If a semicolon (;) is added to the end of an SQL statement, the SQL statement is considered complete.

```
select *   --Comment   // This statement is incomplete. You can add a comment.
from dual;--Comment   // This statement is complete. You cannot add a comment.
show tables;
```

- The running of an ODPS SQL node during data development and the scheduled running of an ODPS SQL node have the following differences:
  - Running during data development: All the statements that contain keywords are combined in the node code and are executed before you execute other SQL statements.
  - Scheduled running: All SQL statements are executed in sequence.

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

The following table describes the differences between the two execution modes for the preceding SQL statements.

SQL statement	Running during data development	Scheduled running

First SQL statement	`set a=b;`
	`set c=d;`
	`create table name1(id string);`
		`set a=b;`
		`create table name1(id string);`

Second SQL statement	`set a=b;`
	`set c=d;`
	`create table name2(id string);`
		`set c=d;`
		`create table name2(id string);`

- You must configure a scheduling parameter in the format of `key=value`. Do not add spaces before or after the equal sign (=). Examples:

```
time={yyyymmdd hh:mm:ss} // The format is invalid.
a =b // The format is invalid.
```

- If you use keywords such as bizdate and date as scheduling parameters, you must specify the values in the format of yyyymmdd. If you want to use other time formats, do not use the preceding keywords as scheduling parameters. Example:

```
bizdate=201908 // The format is invalid.
```

- You can use only statements that start with SELECT, READ, or WITH to query the result data of a node during data development. Otherwise, no results are returned.
- If you want to execute multiple SQL statements, separate the SQL statements with semicolons (;). Specify each SQL statement in a separate line.
  - Incorrect example

    ```
    create table1;create table2
    ```

  - Correct example

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

- If new data types are used for the additional functions of MaxCompute V2.0, you must add `set odps.sql.type.system.odps2=true;` before the SQL statements that use the functions, and commit and execute the code together with the SQL statements.
- If you want to add comments to SQL statements, do not use semicolons (;) in comments.

  Incorrect example:

```
create table1;  // Create a table named table1; then, create a table named table2.
create table2;
```

- The code of an ODPS SQL node can be a maximum of 200 KB in size and can contain a maximum of 200 SQL statements.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. In the Scheduled Workflow pane, move the pointer over the **+Create** icon and choose **Create Node > MaxCompute > ODPS SQL**.

   Alternatively, you can click a workflow in the Scheduled Workflow pane, right-click **MaxCompute**, and then choose **Create Node > ODPS SQL**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the configuration tab of the ODPS SQL node, write and run code in the code editor.

   After the node is created, write the code of the ODPS SQL node. The code must conform to the SQL syntax. For more information about SQL syntax, see Overview of MaxCompute SQL.

   > **Note** Due to the adjustment made by the International Organization for Standardization (ISO) on the UTC+8 time zone, differences exist between the actual time and the output time when you execute related SQL statements in DataWorks. In a year from 1900 to 1928, the time difference is 352 seconds. Before the year of 1900, the time difference is 9 seconds.

   You cannot use a `SET` statement separately in the node code. The SET statement must be executed together with other SQL statements. For example, you can execute a SET statement together with a SELECT statement.

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

   For more information about SET statements, see SET operations.

   Example: Create a table, insert data into the table, and then query data in the table.

   i. Create a table named test1.

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

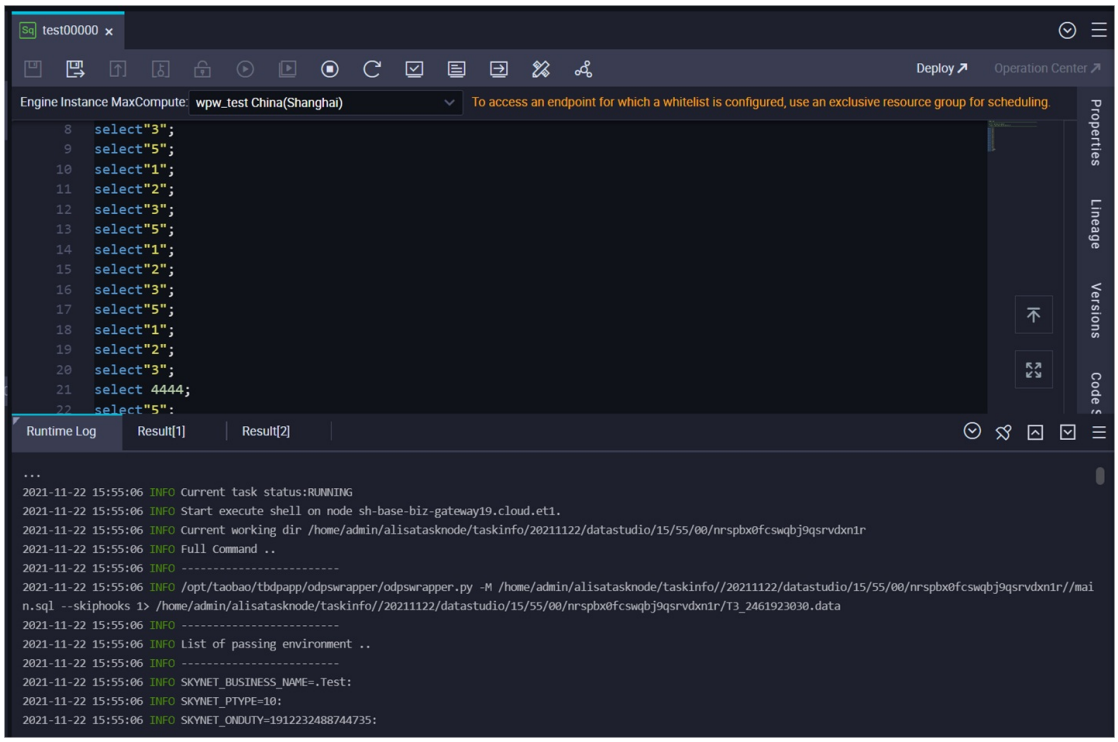
ii. Insert data into the table.

```
INSERT INTO test1 VALUES (1,'Adam',43,'Male');
INSERT INTO test1 VALUES (1,'Carlos',32,'Male');
INSERT INTO test1 VALUES (1,'Claire',27,'Female');
INSERT INTO test1 VALUES (1,'Alan',24,'Male');
INSERT INTO test1 VALUES (1,'Alice',35,'Female');
INSERT INTO test1 VALUES (1,'Sofia',22,'Female');
INSERT INTO test1 VALUES (1,'Raju',55,'Male');
```

iii. Query data in the table.

```
select * from test1;
```

iv. After you enter the preceding SQL statements in the code editor, click the icon in the toolbar. DataWorks executes the SQL statements from top to bottom and then displays logs.



> **Note**
>
> - If multiple MaxCompute compute engine instances are associated with the current workspace, select a desired MaxCompute compute engine instance before you click the Run icon.
> - If the MaxCompute compute engine instance that you selected uses a shared resource group charged based on the pay-as-you-go billing method, click the icon in the toolbar to estimate the cost. You can view your bill for the accurate expenses that are generated when you run this node.

If you execute an `INSERT INTO` statement, unexpected data duplication may occur. DataWorks does not re-execute the `INSERT INTO` statement. However, DataWorks may rerun the nodes that contain the statement. We recommend that you do not use the `INSERT INTO` statement. If DataWorks runs a node that contains the INSERT INTO statement, the following information appears in run logs:

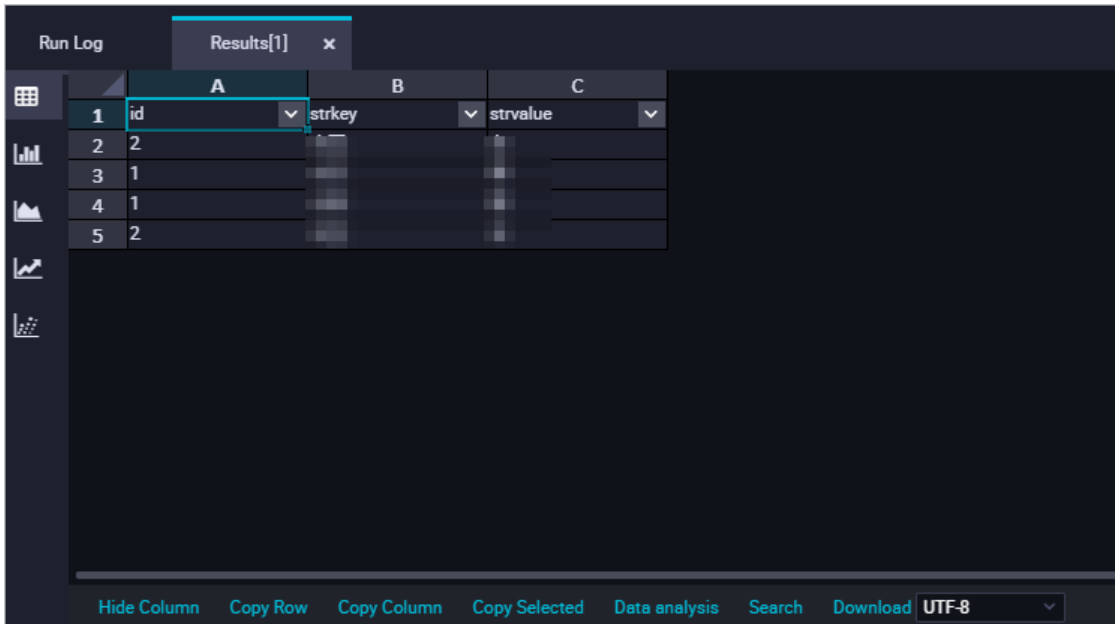
```
The INSERT INTO statement may cause repeated data insertion. DataWorks does not re-execute the INSERT INTO statement, but it may rerun nodes that contain the statement. We recommend that you do not use the INSERT INTO statement.
If you continue to use the INSERT INTO statement, you are deemed to be aware of the risks and are willing to take the consequences of potential data duplication.
```

v. After the SQL statements are executed, click the icon in the toolbar to save the SQL code.

vi. View the query result.

DataWorks displays the query result in a workbook.

You can view or manage the query result in the workbook, or copy the query result to an Excel file on your on-premises machine.



Button	Description
Hide Column	Select one or more columns and click **Hide Column** in the lower part to hide the selected columns.
Copy Row	Select one or more rows and click **Copy Row** in the lower part to copy the selected rows.
Copy Column	Select one or more columns and click **Copy Column** in the lower part to copy the selected columns.
Copy Selected	Select one or more cells in the workbook and click **Copy Selected** in the lower part to copy the selected cells.
Data Analysis	Click **Data Analysis** in the lower part to go to the **workbook editing** page.
Search	Click **Search** in the lower part to search for data in the workbook. After you click the button, a search box appears in the upper-right corner of the Result tab.
Download	Download the result data in the **GBK** or **UTF-8** encoding format.

6. On the node configuration tab, click the **Properties** tab in the right-side navigation pane and configure scheduling properties for the node. For more information, see Configure basic properties.
7. Commit the node.

> **Notice** You can commit the node only after you configure the **Rerun** and **Parent Nodes** parameters.

i. Click the icon in the toolbar.

ii. In the **Commit Node** dialog box, configure the **Change description** parameter.

If an alert appears, indicating that the input and output that you configured do not match with those identified in the code lineage analysis process, you can choose to ignore the alert, or click **Cancel** and then modify the dependencies.

iii. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit a node. For more information, see Deploy nodes.

8. Test the node. For more information, see View auto triggered nodes.

5.3.2. Create an SQL component node

An SQL component is an SQL script template that contains multiple input and output parameters. You can create and run an SQL component node to filter source table data, join source tables, and aggregate source tables to generate a result table.

Prerequisites

- DataWorks Standard Edition or higher is activated.
- A MaxCompute compute engine is bound to the workspace where you want to create an SQL component node. The MaxCompute service is available in a workspace only after you bind a MaxCompute compute engine to the workspace on the **Workspace Management** page. For more information, see Configure a workspace.
- SQL script templates are prepared. For more information, see Create a script template.

Context

When a new version is released for a script template, you can decide whether to update the version of the script template used in your nodes to the new version.

The script template update feature allows developers to update script template versions. This feature helps improve the process execution efficiency and optimize the business performance.

Assume that User A uses a script template that is released by User B in Node C. After User B updates the version of the script template, User A receives an update notification. User A can decide whether to update the version of the script template in Node C.

To update the version of the SQL script template in an SQL component node, perform the following steps: Go to the configuration tab of the node, click **Update code version** in the upper-right corner of the code editor, and then check whether the parameter settings of the old-version SQL script template are valid in the new version. If the parameter settings are invalid in the new version, modify the settings based on the description of the new-version SQL script template. Then, save the node and commit it for deployment.

Procedure

1. Go to the **DataStudio** page.

i. Log on to the DataWorks console.

i. Log on to the DataWorks console.

ii. In the left-side navigation pane, click **Workspaces**.

iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.

2. On the Data Development tab, move the pointer over the +Create icon and choose **MaxCompute > SQL component node**.

   Alternatively, you can click a workflow in the Business process section, right-click **MaxCompute**, and then choose **New > SQL component node**.

3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

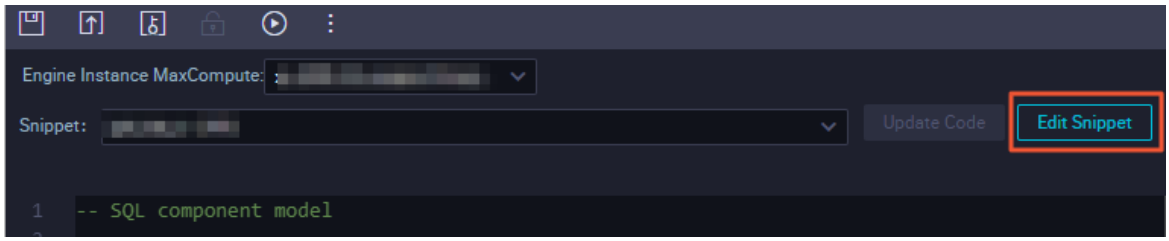
   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.

5. On the node configuration tab, select an SQL script template from the **Select code components** drop-down list.

   If multiple MaxCompute compute engines are bound to the current workspace, select one from the **MaxCompute Engine instance** drop-down list.

   After you select an SQL script template, you can click **Open component** to go to the details page of the template.

   To improve development efficiency, you can create data analytics nodes by using the script templates that are provided by workspace members and tenants.

   - The script templates that are provided by members of the current workspace are available on the **Components** tab.
   - The script templates that are provided by tenants are available on the **Common components** tab.

6. Click the **Parameter configuration** tab in the right-side navigation pane and set parameters for the SQL script template.

7. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iv. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.

8. Test the node. For more information, see View auto triggered nodes.

5.3.3. Create a MaxCompute Spark node

A MaxCompute Spark node processes data by using Java and Python. This topic describes how to create and configure a MaxCompute Spark node.

Context

Python resources are referenced in the user-defined functions (UDFs) of Python. However, Python resources have limited usage because you can obtain only a few dependent third-party packages.

PyODPS 2 and PyODPS 3 nodes support Python resources. For more information, see Create a PyODPS 2 node and Create a PyODPS 3 node.

This topic describes how to create JAR and Python resources and upload them based on your business requirements after you create a MaxCompute Spark node.

Create a JAR resource

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the DataStudio page, move the pointer over the **Create** icon and choose **MaxCompute > Resource > JAR**.

   Alternatively, you can find your desired workflow, right-click the workflow name, and then choose **Create > MaxCompute > Resource > JAR**.
3. In the **Create Resource** dialog box, specify **Resource Name** and **Location**.

   > **Note**
   > - If multiple MaxCompute compute engine instances are bound to the current workspace, you must select one from the **Engine Instance MaxCompute** drop-down list.
   > - If the selected JAR package has been uploaded from the MaxCompute client, clear **Upload to MaxCompute**. If you do not clear it, an error occurs during the upload process.
   > - The resource name can be different from the name of the uploaded file.
   > - The resource name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). The name is not case-sensitive. A JAR resource name must end with .jar, and a Python resource name must end with .py.
4. Click **Upload** and select the file to upload.
5. Click **Create**.

6. Click the icon in the top toolbar to commit the code.
7. In the **Commit Node** dialog box, enter your comments in the **Change description** field and click **OK**.

Create a Python resource

1. On the **DataStudio** page, move the pointer over the **+Create** icon and choose **MaxCompute > Resource > Python**.

   Alternatively, you can find your desired workflow, right-click the workflow name, and then choose **Create > MaxCompute > Resource > Python**.
2. In the **Create Resource** dialog box, specify **Resource Name** and **Location**.

   > **Note**
   > - If multiple MaxCompute compute engine instances are bound to the current workspace, you must select one from the **Engine Instance MaxCompute** drop-down list.
   > - The resource name can contain letters, digits, periods (.), underscores (_), and hyphens (-), and must end with .py.
   > - The created Python resource can be run by using only the code of Python 2.X or 3.X.
3. Click **Create**.
4. On the node configuration tab, enter the Python code.

   In the following example, the Python code defines the logic for checking whether parameter values are correct instead of the logic for processing data.

```
# -*- 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. Click the icon in the top toolbar to commit the code.
6. In the **Commit Node** dialog box, enter your comments in the **Change description** field and click **OK**.

Create a MaxCompute Spark node

1. On the **DataStudio** page, move the pointer over the **+Create** icon and choose **MaxCompute > ODPS Spark**.

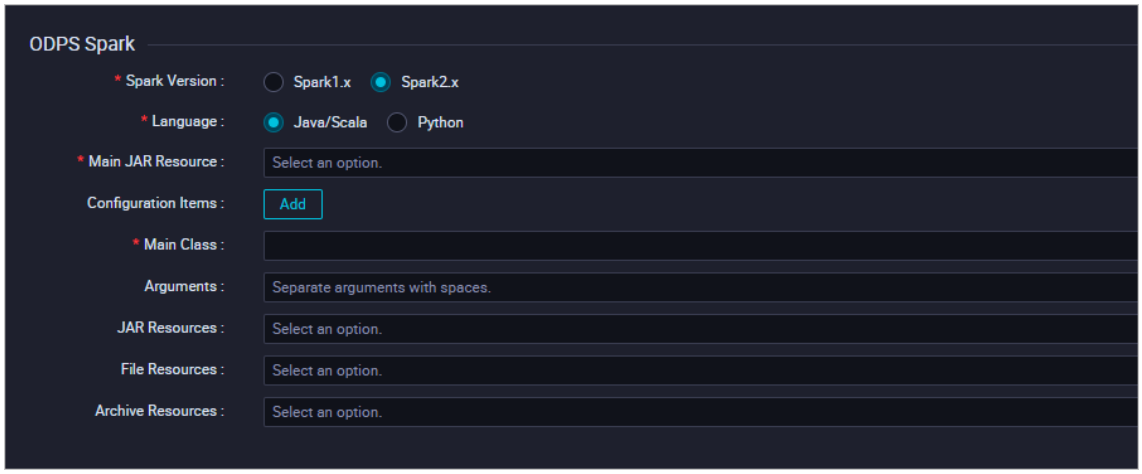
   Alternatively, you can find your desired workflow, right-click the workflow name, and then choose **Create > MaxCompute > ODPS Spark**.
2. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

3. Click **Commit**.
4. On the configuration tab of the MaxCompute Spark node, configure the parameters. For more information about MaxCompute Spark nodes, see Overview.

   Two options are available for the **Spark Version** and **Language** parameters of the MaxCompute Spark node. The parameters on this tab vary based on the value of the **Language** parameter. You can configure the parameters as prompted.

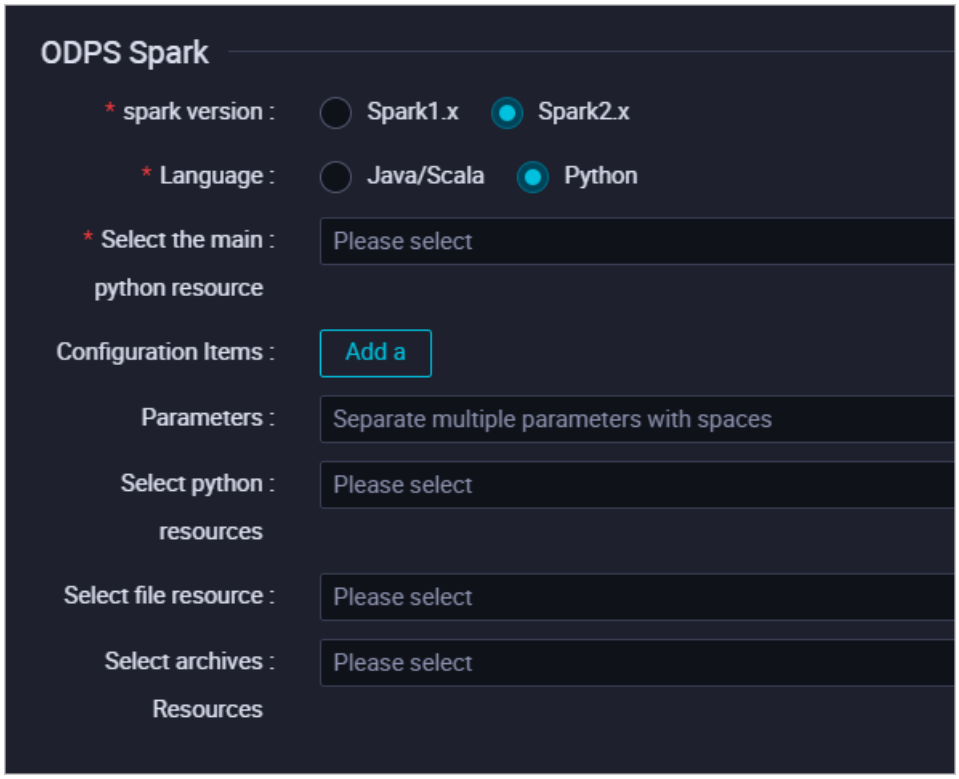
   - The following table describes the parameters that appear on this tab when you set **Language** to **Java/Scala**.



Parameter	Description
Spark Version	The Spark version of the node. Valid values: **Spark1.x** and **Spark2.x**.
Language	The programming language used by the node. Select **Java/Scala**.
Main JAR Resource	The main JAR resource referenced by the node. Select the desired JAR resource from the drop-down list.
Configuration Items	The configuration items of the node. Click **Add**, and enter a key and a value to add a configuration item. Alternatively, you can directly select a key from the field that appears. In this case, a value is automatically entered for the key.
Main Class	The class name of the node.

Parameter	Description
Arguments	The parameters of the node. Separate multiple parameters with spaces. You can configure scheduling parameters. If you want to perform such an operation, click Properties in the right-side navigation pane. For more information, see Overview of scheduling parameters. **Note** After you configure the scheduling parameters, you must continue to configure the parameters for the node. The scheduling parameters and node parameters are run in sequence.
JAR Resources	The JAR resource referenced by the node. The system displays all the uploaded JAR resources. Select the desired JAR resource from the drop-down list.
File Resources	The file resource referenced by the node. The system displays all the uploaded file resources. Select the desired file resource from the drop-down list.
Archive Resources	The archive resource referenced by the node. The system displays all the uploaded archive resources that are compressed. Select the desired archive resource from the drop-down list.

- The following table describes the parameters that appear when you set **Language** to **Python**.



Parameter	Description
Spark Version	The Spark version of the node. Valid values: **Spark1.x** and **Spark2.x**.
Language	The programming language used by the node. Select **Python**.
Main Python Resource	The main Python resource referenced by the node. Select the desired Python resource from the drop-down list.
Configuration Items	The configuration items of the node. Click **Add**, and enter a key and a value to add a configuration item. Alternatively, you can directly select a key from the field that appears. In this case, a value is automatically entered for the key.
Arguments	The parameters of the node. Separate multiple parameters with spaces.
Python Resources	The Python resource referenced by the node. The system displays all the uploaded Python resources. Select the desired Python resource from the drop-down list.
File Resources	The file resource referenced by the node. The system displays all the uploaded file resources. Select the desired file resource from the drop-down list.
Archive Resources	The archive resource referenced by the node. The system displays all the uploaded archive resources that are compressed. Select the desired archive resource from the drop-down list.

5. On the node configuration tab, click **Properties** in the right-side navigation pane. On the Properties tab, configure the scheduling properties for the node. For more information, see Configure basic properties.
6. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.
7. Test the node. For more information, see View auto triggered nodes.

5.3.4. Create a PyODPS 2 node

DataWorks supports PyODPS 2 nodes, which are integrated with MaxCompute SDK for Python. You can edit Python code in PyODPS 2 nodes of DataWorks to process data in MaxCompute.

Context

MaxCompute provides SDK for Python. You can use the SDK for Python to process data in MaxCompute. For more information, see SDK for Python.

> **Note**
> - The Python version of PyODPS 2 nodes is 2.7.
> - Each PyODPS 2 node can process a maximum of 50 MB data and can occupy a maximum of 1 GB memory. Otherwise, the PyODPS 2 node stops running. Avoid writing excessive data processing code for a PyODPS 2 node.
> - For more information about the hints parameter, see SET operations.

PyODPS 2 nodes are designed to use MaxCompute SDK for Python. If you want to run pure Python code, you can create a Shell node to run the Python scripts that are uploaded to DataWorks. For information about how to reference a third-party package in a PyODPS 2 node, see Use a PyODPS node to reference a third-party package.

Create a PyODPS 2 node

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Move the pointer over the **+Create** icon and choose **MaxCompute > PyODPS 2**.

   Alternatively, you can click the required workflow in the **Business Flow** section, right-click **MaxCompute**, and then choose **Create > PyODPS 2**.

   For more information about how to create a workflow, see Create a workflow.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. Edit the PyODPS 2 node.
   i. Use the MaxCompute entry.

   In DataWorks, each PyODPS 2 node includes the global variable odps or o, which is the MaxCompute entry. Therefore, you do not need to manually specify the MaxCompute entry.

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

ii. Execute SQL statements.

PyODPS 2 supports MaxCompute SQL queries and allows you to obtain the query results. The return value of the execute_sql or run_sql method is running instances.

Not all SQL statements that you can execute on the MaxCompute client are supported by PyODPS 2. To execute statements other than data definition language (DDL) and data manipulation language (DML) statements, use other methods.

For example, to execute a GRANT or REVOKE statement, use the run_security_query method. To run a Machine Learning Platform for AI (PAI) command, use the run_xflow or execute_xflow method.

```
o.execute_sql('select * from dual')  # Execute the statement in synchronous mode. O
ther nodes are blocked until the SQL statement is executed.
instance = o.run_sql('select * from dual') # Execute the statement in asynchronous
mode.
print(instance.get_logview_address()) # Obtain the Logview URL.
instance.wait_for_success() # Block other nodes until the SQL statement is executed
.
```

iii. Set runtime parameters.

You can use the hints parameter to set the runtime parameters. The type of the hints parameter is dict.

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

If you set the sql.settings parameter for the global configuration, you must set the runtime parameters each time you run the code.

```
from odps import options
options.sql.settings = {'odps.sql.mapper.split.size': 16}
o.execute_sql('select * from PyODPS_iris')  # The hints parameter is set based on g
lobal configuration.
```

iv. Obtain SQL query results.

You can perform the open_reader operation on the instance that executes the SQL statement in the following scenarios:

- The SQL statement returns structured data.

```
with o.execute_sql('select * from dual').open_reader() as reader:
for record in reader:  # Process each record.
```

- SQL statements such as DESC are executed. In this case, you can use the reader.raw property to obtain raw SQL query results.

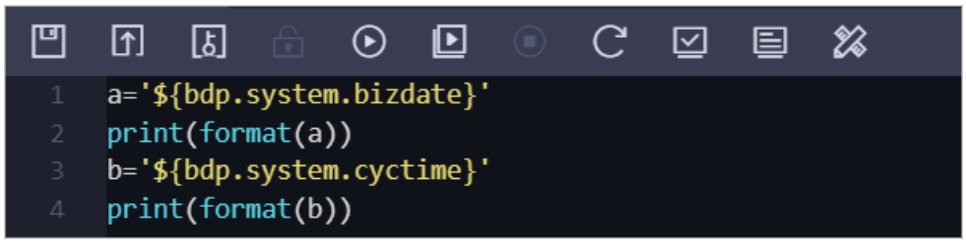
```
with o.execute_sql('desc dual').open_reader() as reader:
print(reader.raw)
```

> **Note** If you use a custom scheduling parameter and run the PyODPS 2 node on the page, you must fix the parameter to a time. PyODPS nodes do not support direct replacement.

6. On the node configuration tab, click **Properties** in the right-side navigation pane. Set the scheduling properties for the node. For more information, see Configure basic properties.

   PyODPS 2 nodes can use built-in scheduling parameters and custom scheduling parameters:

   - If the PyODPS 2 node uses built-in scheduling parameters, you can assign values on the node configuration tab.

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

     > **Note** The shared resource group cannot be connected to the Internet. If you want to connect to the Internet, we recommend that you use a custom resource group or exclusive resources for scheduling. Only DataWorks Professional Edition supports custom resource groups. You can purchase exclusive resources for scheduling in all DataWorks editions. For more information, see DataWorks exclusive resources.

   - You can also set custom scheduling parameters in the **Properties > General** section.

     > **Note** You must specify a custom parameter in the format of args['Parameter name'], for example, `print (args['ds'])` .

7. Commit the node.

   > **Notice** Before you commit the node, you must set the **Rerun** and **Parent Nodes** parameters.

   i. Click the icon in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node.

8. Test the node. For more information, see View auto triggered nodes.

Built-in services for a PyODPS node

A PyODPS node contains the following built-in services:

- setuptools
- cython
- psutil
- pytz
- dateutil
- requests
- pyDes

- numpy
- pandas
- scipy
- scikit_learn
- greenlet
- six
- Other built-in services in Python 2.7, such as smtplib

5.3.5. Create a PyODPS 3 node

This topic describes the usage limits of PyODPS 3 nodes and how to create a PyODPS 3 node in DataWorks.

Limits

- Python 3 defines bytecode differently in its different subversions such as Python 3.7 and Python 3.8.

  MaxCompute is compatible with Python 3.7. A MaxCompute client that uses another subversion of Python 3 will return an error when it executes code with specific syntax. For example, a MaxCompute client that uses Python 3.8 will return an error when it executes code with the finally block syntax. We recommend that you use Python 3.7.
- Each PyODPS 3 node can process a maximum of 50 MB data and can occupy a maximum of 1 GB memory. Otherwise, DataWorks terminates the PyODPS 3 node. Do not write code that will process an extra large amount of data in a PyODPS 3 node.
- PyODPS 3 nodes can run on a shared resource group and an exclusive resource group for scheduling that is purchased after April 2020. If you create an exclusive resource group for scheduling before April 2020, submit a ticket to upgrade the resource group.

Create a PyODPS 3 node

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Move the pointer over the **+Create** icon and choose **MaxCompute > PyODPS 3**.

   Alternatively, click the required workflow under **Business Flow**, right-click **MaxCompute**, and choose **Create > PyODPS 3**.

   For more information about how to create a workflow, see Create a workflow.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. Edit and run the PyODPS 3 node.

For example, if you want to use the execute_sql() method, you must specify runtime parameters for the SQL statements.

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

If you want to use a user-defined function (UDF) for DataFrame, such as df.map, df.map_reduce, df.apply, and df.agg, specify the following settings:

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

PyODPS determines the runtime environment of the UDF and commits SQL statements based on the Python version that the client uses. Assume that a public Python UDF is used to call DataFrame. When the client uses Python 3, statements are interpreted to Python 3. If the UDF executes a print statement specific to Python 2, the client returns the ScriptError error. For more information about how to reference a third-party package in a PyODPS 2 node, see Use a PyODPS node to reference a third-party package.

6. Click the **Properties** tab in the right-side navigation pane and set the scheduling properties for the node. For more information, see Configure basic properties.
7. Save and commit the node.

   Notice You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.
8. Test the node. For more information, see View auto triggered nodes.

5.3.6. Create an ODPS Script node

You can create an ODPS Script node to develop an SQL script by using the code editor provided by the MaxCompute V2.0 SQL engine.

Context

An ODPS Script node allows DataWorks to compile an SQL script as a whole, instead of compiling the SQL statements in the script one by one. This way, the SQL script is committed and run as a whole. This ensures that an execution plan is queued and executed only once, making full use of MaxCompute computing resources.

ODPS Script nodes allow you to write SQL statements based on your business logic in a way similar to that of using a common programming language. You do not need to consider how to organize the SQL statements.

```
-- SET statements
set odps.sql.type.system.odps2=true;
[set odps.stage.reducer.num=***;]
[...]
-- Data definition language (DDL) statements
create table table1 xxx;
[create table table2 xxx;]
[...]
-- Data manipulation language (DML) statements
@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 nodes have the following limits:

- ODPS Script nodes support SET statements, DML statements, and some DDL statements. The DDL statements, such as `DESC` and `SHOW` statements, that are used to return data are not supported.
- A complete script consists of SET statements, DDL statements, and DML statements in sequence. You can write one or more statements of each type, or skip a type without writing any statements of that type. However, you cannot mix different types of statements together. You must strictly follow this sequence: SET statements > DDL statements > DML statements.
- The at signs (@) preceding several statements indicate that these statements are connected by using variables.
- You can write only one statement, for example, a SELECT statement, that returns data in a script. If you write multiple such statements in a script, an error occurs. We recommend that you do not use SELECT statements in a script.
- You can write only one `CREATE TABLE AS` statement in a script, and this statement must be the last statement in the script. We recommend that you write CREATE TABLE statements and INSERT statements in different sections to separate them.
- If one statement in a script fails, the whole script fails.
- A job is generated to process data only after all the input data is prepared for a script.
- If you specify a statement for writing data to a table and then a statement for reading data from the table in the same script, an error occurs. For example, if you write the following statements in a script, an error occurs:

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

To avoid the error, write the statements in the following format:

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

Sample script:

```
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;
```

ODPS Script nodes are applicable to the following scenarios:

- You can use an ODPS Script node to rewrite a single statement with nested subqueries, or a script that must be split into multiple statements to make it simpler.
- Data from different data stores may be ready at different time points, and the time difference may be large. For example, the data from one data store can be ready at 01:00, whereas that from the other data store can be ready at 07:00. In this case, table variables are not suitable for connecting statements. You can use an ODPS Script node to combine the statements to a script.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the Data Development tab, move the pointer over the **+Create** icon and choose **MaxCompute > ODPS Script**.

   Alternatively, you can click a workflow in the Business process section, right-click **MaxCompute**, and then choose **New > ODPS Script**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.

5. On the node configuration tab, write the SQL script of the ODPS Script node. For more information, see Develop and submit an SQL script.
6. On the node configuration tab, click the **Scheduling configuration** tab in the right-side navigation pane. On the Scheduling configuration tab, set the scheduling properties for the node. For more information, see Configure basic properties.
7. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.
8. Test the node. For more information, see View auto triggered nodes.

5.3.7. Create an ODPS MR node

MaxCompute supports the MapReduce API. You can create and commit ODPS MR nodes that call the Java API operations of MapReduce to develop MapReduce programs for processing data in MaxCompute.

Prerequisites

Required resources are uploaded and committed.

For more information about how to edit and use an ODPS MR node, see WordCount example.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Create a JAR resource.
   i. On the Data Development tab, move the pointer over the Create icon and choose **MaxCompute > Resource > JAR**.

      Alternatively, you can click a workflow in the Business process section, right-click **MaxCompute**, and then choose **New > Resource > JAR**.

ii. In the **New resource** dialog box, set the **Resource Name** and **Destination folder** parameters.

> **Note**
> - If multiple MaxCompute compute engines are bound to the current workspace, you must select one from the **MaxCompute Engine instance** drop-down list.
> - If the selected JAR package has been uploaded from the MaxCompute client, clear **Upload as an ODPS resource**. If you do not clear it, an error will occur during the upload process.
> - The resource name can be different from the name of the uploaded file.
> - A resource name can contain letters, digits, underscores (_), and periods (.), and is not case-sensitive. It must be 1 to 128 characters in length. A JAR resource name must end with .jar, and a Python resource name must end with .py.

iii. Click **Click Upload**, select a local JAR package, and then click **Open**.

In this example, upload the mapreduce example.jar package.

iv. Click **Confirm**.

v. Click the and icons in the toolbar to save and commit the resource to the development environment.

3. Create an ODPS MR node.

i. On the Data Development tab, move the pointer over the Create icon and choose **MaxCompute > ODPS MR**.

Alternatively, you can click a workflow in the Business process section, right-click **MaxCompute**, and then choose **New > ODPS MR**.

ii. In the **New node** dialog box, set the **Node name** and **Destination folder** parameters.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

iii. Click **Submit**.

4. On the node configuration tab, enter the following sample code:

```
-- Create an input table.
CREATE TABLE if not exists jingyan_wc_in (key STRING, value STRING);
-- Create an output table.
CREATE TABLE if not exists jingyan_wc_out (key STRING, cnt BIGINT);
    --- Create the dual table.
    drop table if exists dual;
    create table dual(id bigint); -- Create the dual table if no dual table exists in t
he current workspace.
    --- Initialize the dual table.
    insert overwrite table dual select count(*)from dual;
    --- Insert the sample data to the wc_in table.
    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;
-- Reference the created JAR resource. You can find the JAR resource in the resource li
st, right-click the JAR resource, and then select Reference Resources to reference the
resource.
--@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
```

Code description:

- `--@resource_reference`: references a resource. Find the resource to be referenced in the resource list, right-click it, and then select **Reference Resources** to generate the reference statement.
- `-resources`: the name of the referenced JAR resource.
- `-classpath`: the path of the referenced JAR resource. You need to enter only *./Resource name* because the resource has been referenced.
- `com.aliyun.odps.mapred.open.example.WordCount`: the name of the main class in the JAR resource to be called during node running. The main class name must be the same as that in the JAR resource.
- `jingyan_wc_in`: the name of the input table of the ODPS MR node. The input table is created by using the preceding code.
- `jingyan_wc_out`: the name of the output table of the ODPS MR node. The output table is created by using the preceding code.
- If you reference multiple JAR resources in an ODPS MR node, separate the resource paths with commas (,), for example, `-classpath ./xxxx1.jar,./xxxx2.jar`.

5. On the node configuration tab, click the **Scheduling configuration** tab in the right-side navigation pane. On the Scheduling configuration tab, set the scheduling properties for the node. For more information, see Configure basic properties.
6. Save and commit the node.

> **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

i. Click the icon in the toolbar to save the node.

ii. Click the icon in the toolbar.

iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iv. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.

7. Test the node. For more information, see View auto triggered nodes.

5.4. EMR

5.4.1. Overview

After you associate an E-MapReduce (EMR) cluster with a DataWorks workspace, you can create EMR nodes such as EMR Hive, EMR MR, EMR Presto, and EMR Spark SQL nodes. Then, you can configure properties for these EMR nodes and run the nodes. This facilitates metadata management and data generation for EMR users. This topic describes the notes on how to use EMR in DataWorks. We recommend that you read this topic before you use EMR in DataWorks.

Limits

The following table describes the limits of different versions or configurations of EMR clusters for using EMR in DataWorks.

EMR-related feature in DataWorks	Requirements for the versions or configurations of an EMR cluster
View the output information of metadata tables and use the automatic recommendation feature of the Data Map service of DataWorks.	The version of the EMR cluster is later than V3.33.0 or V4.6.
Create and use an EMR JAR resource, Create an EMR table, and Create an EMR function.	The Kerberos or LDAP-based security mode is **disabled** for the EMR cluster.
Create and use nodes except for EMR Hive nodes and EMR Spark nodes.	The Kerberos or LDAP-based security mode is **disabled** for the EMR cluster. If the Kerberos or LDAP-based security mode is enabled, you can create and use only EMR Hive nodes, EMR Spark nodes, EMR Spark SQL nodes, and EMR Spark Streaming nodes.
• DataWorks does not support Flink tasks for EMR. • You can use only EMR Hive nodes in DataWorks to collect EMR metadata and lineage information. • If your EMR clusters or exclusive resource groups for scheduling of DataWorks were created before August 1, 2021, you must submit a ticket to update the plug-in for using EMR in DataWorks to the latest version.	

Preparations

Before you run an EMR node in DataWorks, make the following preparations:

- Purchase and configure an exclusive resource group for scheduling.

  Before you run an EMR node, you must purchase an exclusive resource group for scheduling and connect it to the VPC in which the EMR cluster resides. For more information about how to purchase and configure an exclusive resource group for scheduling, see Overview.

- Check the configurations of the EMR cluster.

  Before you run an EMR node in DataWorks, you must check whether the configurations of the EMR cluster meet the following requirements. Otherwise, an error may occur when you run an EMR node in DataWorks.

  - An EMR cluster is created. An inbound rule that contains the following content is added to the security group to which the EMR cluster belongs:
    - Action: Allow
    - Protocol type: Custom TCP
    - Port range: 8898/8898
    - Authorization object: 100.104.0.0/16
  - If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned when you run EMR nodes.
    a. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

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

    > **Note** In the code, `ALISA.*` and `SKYNET.*` are configurations in DataWorks.

    b. After the whitelist configurations are modified, restart the Hive service to make the configurations take effect. For more information, see Restart a service.
  - Set the hadoop.http.authentication.simple.anonymous.allowed parameter to true on the HDFS page in the EMR console. Then, restart the HDFS and YARN components.

- Associate the EMR cluster with a DataWorks workspace.

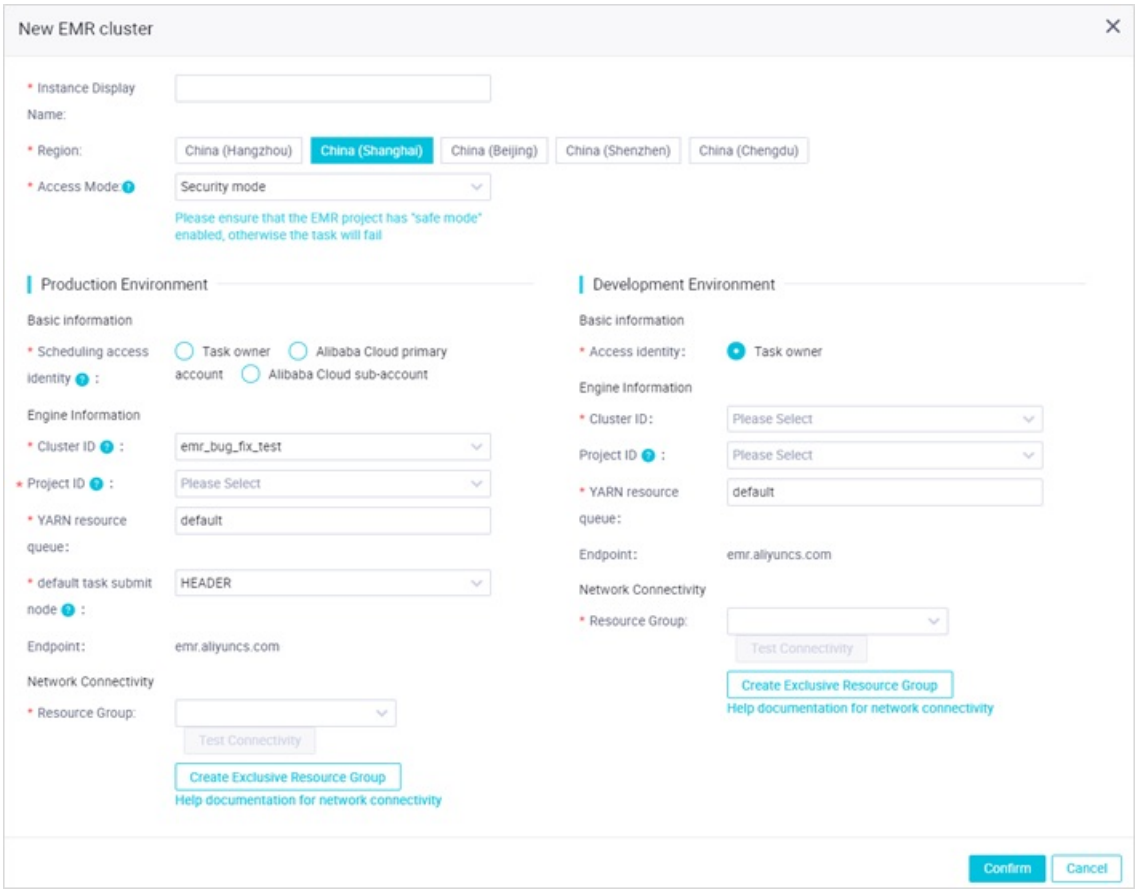
  You can create and develop an EMR node in DataWorks only after you associate an EMR cluster with a DataWorks workspace. When you associate an EMR cluster with a DataWorks workspace, select a mode to access the EMR cluster based on the configurations of the EMR cluster. Take note of the following items when you select the access mode. For more information, see Associate an EMR cluster with a DataWorks workspace.

  - If the LDAP authentication is disabled for the EMR cluster, select Shortcut mode when you associate the EMR cluster with a DataWorks workspace.

- If the LDAP authentication is enabled for the EMR cluster, select Security mode when you associate the EMR cluster with a DataWorks workspace.

  In this scenario, you must perform the following operations:

  - In the EMR console, restart the related components if the LDAP authentication is enabled for them.
  - Turn on Security Mode for the project in which the EMR cluster resides.
  - Set the Access Mode parameter to Security mode when you associate an EMR cluster with a DataWorks workspace.



  - Configure mappings between RAM users and LDAP accounts on the EMR Cluster Configuration page. This operation is important.
  - If the LDAP authentication is enabled for the Impala component, perform the following operations before you run an EMR node in DataWorks:

    a. Download the Impala JDBC driver.

       After the LDAP authentication is enabled, you must provide LDAP authentication credentials when you access Impala by using JDBC. In addition, you must download the Impala JDBC driver from the official website of Cloudera and add the driver to the /usr/lib/hive-current/lib directory. To download the Impala JDBC driver, click DJBC Driver for Impala.

    b. After you download the JAR package, copy the JAR package to the */usr/lib/flow-agent-current/zeppelin/interpreter/jdbc/* directory of the header or gateway node in the EMR cluster.

    c. Restart the FlowAgent component by clicking Flow Agent Daemon in the EMR console.

Create and debug an EMR node

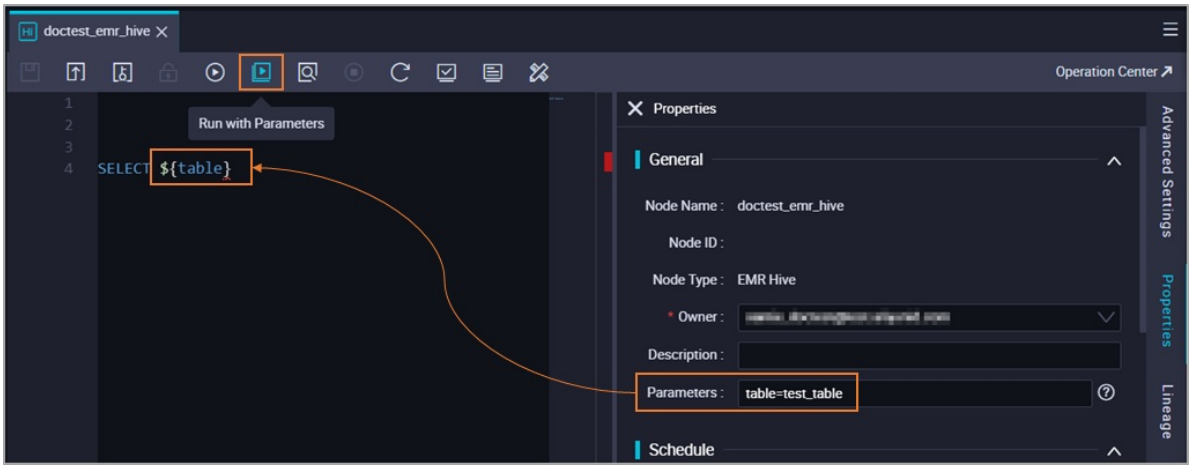
If the preparations are complete, you can create, compile, and run an EMR node in DataWorks. Take note of the following items when you create and debug an EMR node:

- Advanced parameters
  - "USE_GATEWAY":true: If you set this parameter to true, the node is automatically committed to the master node of an EMR gateway cluster.
  - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters that are required to run Spark jobs. You can configure multiple parameters in the --conf xxx=xxx format.
  - "queue": the scheduling queue to which jobs are committed. Default value: default.

    > **Note** Compared with the queue that you configure when you associate an EMR cluster with a DataWorks workspace, the queue that you configure here has a higher priority. Later versions will support the parameters of the default queue.

  - "vcores": the number of CPU cores. Default value: 1. We recommend that you use the default value.
  - "memory": the memory that is allocated to the launcher. Default value: 2048. We recommend that you use the default value.
  - "priority": the priority. Default value: 1.
  - "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. A value of false indicates that only one SQL statement is executed at a time. A value of true indicates that multiple SQL statements are executed at a time.
- Debugging

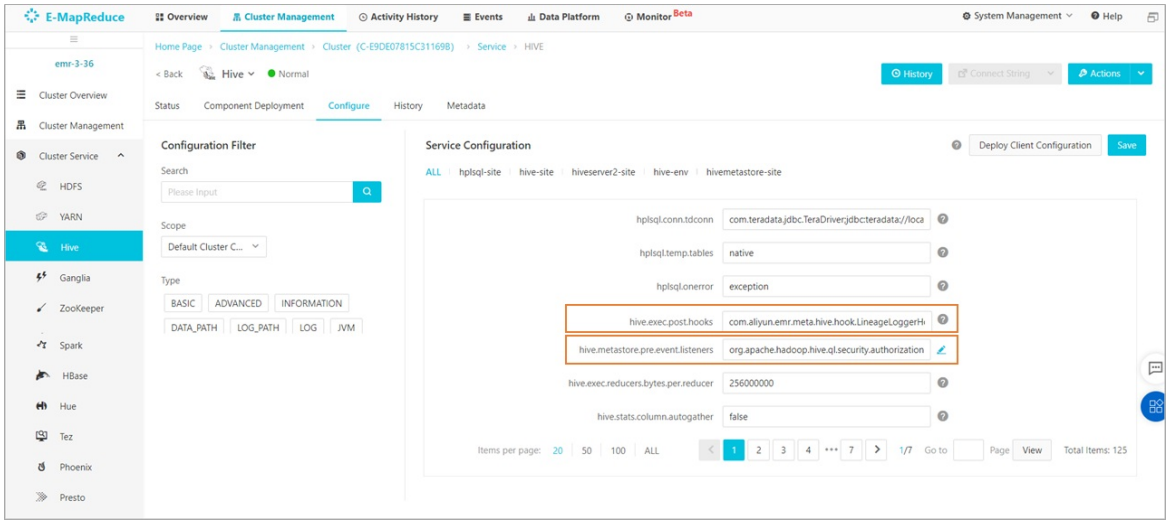
  If parameters are used in the code of the node, you must declare these parameters in the **Parameters** field on the **Properties** pane and click **Advanced run** to start debugging.



Data Map

Before you use DataWorks to collect metadata from EMR, you must check whether the configurations of the EMR cluster meet the requirements.

1. In the EMR console, check whether the configurations for the **hive.metastore.pre.event.listeners** and **hive.exec.post.hooks** fields take effect.



2. Set the parameters that are required to collect metadata on the Data Map page in the DataWorks console. For more information, see Collect and view metadata.

FAQ: Why does the operation to commit an EMR job fail?

- Problem description

  The following error message is returned after an EMR job failed to be committed in DataWorks.

  > **Note** The error message does not indicate that the job failed to be run.

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

- Possible causes

  The FlowAgent component of the EMR cluster is required to integrate EMR into DataWorks. If the preceding error message is returned, the cause may lie in the FlowAgent component. Restart the component to fix the problem.

- Solution:

  You can restart the FlowAgent component to fix the problem.

  i. Visit the FlowAgent page.

  By default, the FlowAgent page cannot be directly visited from the EMR console. Therefore, you need to visit the page of a component, and then modify the URL to visit the FlowAgent page.

  Take the HDFS Components page as an example. After you visit the page, the URL of the HDFS Components page is shown as `https://emr.console.aliyun.com/#/ cn-hangzhou/cluster/C-XXXXXXXXXXXX/ service/HDFS`. You need to change the component name HDFS at the end of the URL to EMRFLOW. The complete URL is shown as `https://emr.console.aliyun.com/#/ cn-hangzhou/cluster/C-XXXXXXXXXXXXXX/ service/EMRFLOW`.

  ii. Restart the FlowAgent component.

In the upper-right corner of the FlowAgent page, choose **Actions > Restart All Components**.

5.4.2. Associate an EMR cluster with a DataWorks workspace

This topic describes how to associate an E-MapReduce (EMR) cluster with a DataWorks workspace. In DataWorks, you can create nodes such as EMR Hive, EMR MR, EMR Presto, and EMR Spark SQL nodes based on an EMR compute engine instance and configure EMR workflows. You can also schedule the nodes and manage metadata. This improves your data output. You must associate an EMR cluster with a DataWorks workspace before you create an EMR node to develop data. This topic describes how to associate an EMR cluster with a DataWorks workspace.

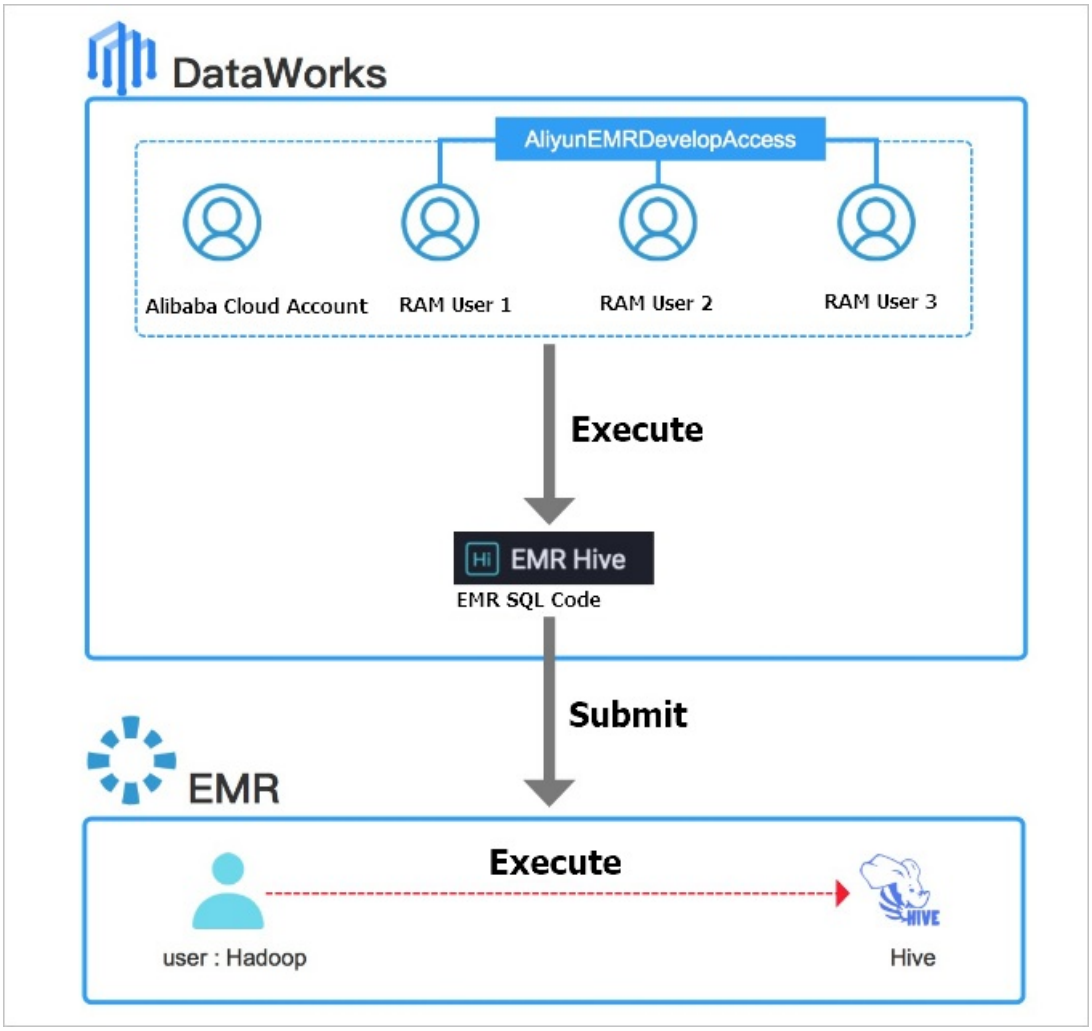
DataWorks provides two modes for you to associate an EMR cluster with a workspace: **Shortcut mode** and **Security mode**. The two modes can meet the security requirements of various enterprises. If you associate an EMR cluster with a workspace by using the **Shortcut mode**, you can create and run EMR nodes to generate data. If you associate an EMR cluster with a workspace by using the **Security mode**, you can create and run EMR nodes to generate data and manage permissions on the data to ensure higher security.

Shortcut mode

In **Shortcut mode**, if you run or schedule EMR nodes in DataWorks by using your Alibaba Cloud account or as a RAM user, the code is committed to the EMR cluster and run by the Hadoop user of the EMR cluster.

> **Notice**
>
> - The Hadoop user has all the permissions on the EMR cluster. Proceed with caution when you use the Shortcut mode to associate an EMR cluster with a workspace.
> - Before you use the **Shortcut mode** to associate an EMR cluster with a workspace, you must attach the AliyunEMRDevelopAccess policy to workspace roles such as developers and administrators. This way, the roles can be used to create and run EMR nodes in DataStudio.
>   - The AliyunEMRDevelopAccess policy is attached to Alibaba Cloud accounts by default.
>   - To run EMR nodes as a RAM user, you must attach the AliyunEMRDevelopAccess policy to the RAM user. For more information, see Grant permissions to RAM users.

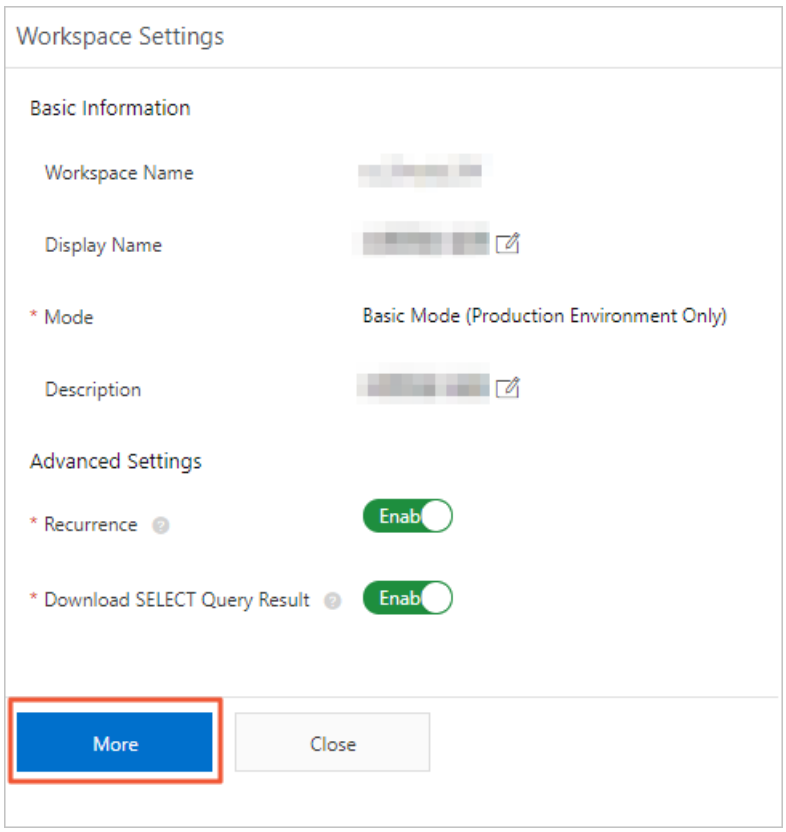
The **Shortcut mode** is suitable for workspaces that do not require strict permission management for users who run nodes.



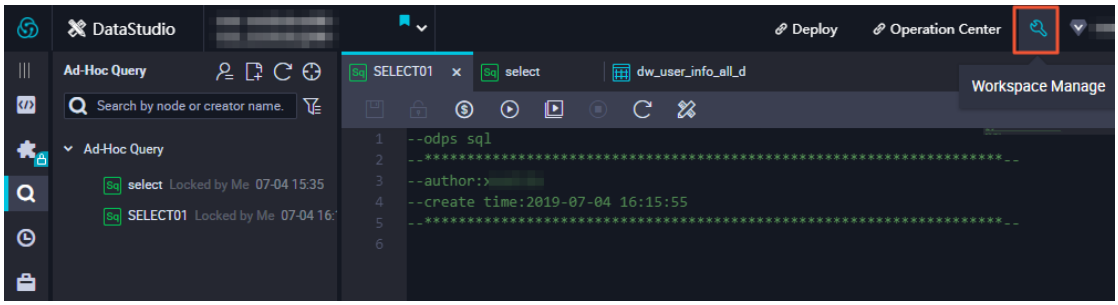
To associate an EMR cluster with a workspace in **Shortcut mode**, perform the following steps:

1. Go to the **Workspace Management** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.

iii. Select the region in which the workspace resides. Then, find the workspace and click **Workspace Settings** in the Actions column. In the **Workspace Settings** panel, click **More**. The **Workspace Management** page appears.

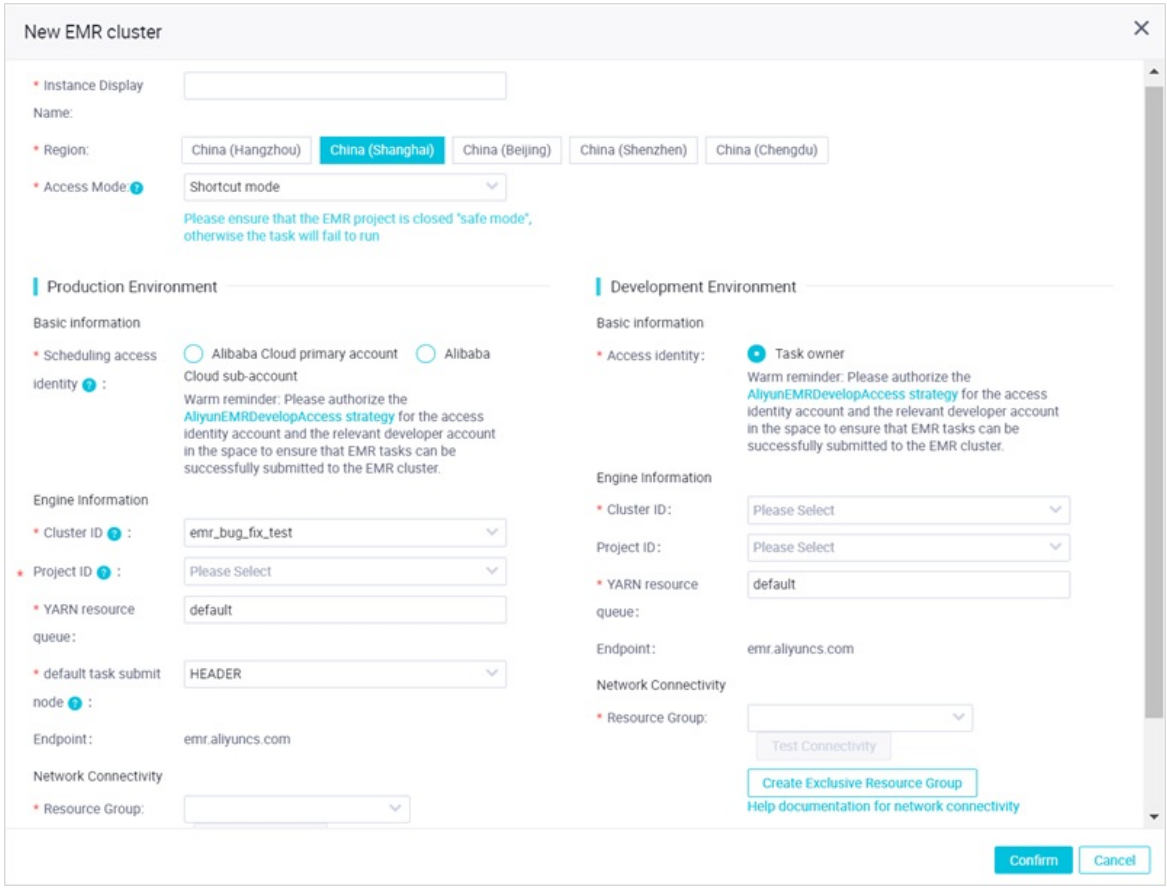


Alternatively, you can find the workspace and click **Data Analytics** in the Actions column. On the **DataStudio** page, click the icon in the upper-right corner. The **Workspace Management** page appears.



2. In the **Compute Engine Information** section, click the **E-MapReduce** tab.
3. On the **E-MapReduce** tab, click **Add Instance**.
4. In the **New EMR cluster** dialog box, set the parameters.

Parameters in the New EMR cluster dialog box vary based on the mode in which your DataWorks workspace runs. The following table describes the parameters for a DataWorks workspace in standard mode. You must set the parameters for both the production environment and the development environment.



Parameter	Description
Instance Display Name	The display name of the EMR compute engine instance.
Region	The region of the current workspace. The value of this parameter cannot be changed.
Access Mode	The access mode of the EMR cluster. Select **Shortcut mode** from the drop-down list.

Parameter	Description

Scheduling access identity	The identity that is used to commit the code of an EMR node to the EMR cluster. The code is committed when the node is committed to the scheduling system of DataWorks in the production environment. Valid values: **Alibaba Cloud primary account** and **Alibaba Cloud sub-account**.
	Note
	- This parameter is available only for the production environment.
	- Before you use the **Shortcut mode** to associate an EMR cluster with a workspace, you must attach the AliyunEMRDevelopAccess policy to workspace roles such as developers and administrators. This way, the roles can be used to create and run EMR nodes in DataStudio.
	  - The AliyunEMRDevelopAccess policy is attached to Alibaba Cloud accounts by default.
	  - To run EMR nodes as a RAM user, you must attach the AliyunEMRDevelopAccess policy to the RAM user. For more information, see Grant permissions to RAM users.

Parameter	Description

Access identity	The identity that is used to commit the code of an EMR node in the development environment to the EMR cluster. Default value: **Task owner**.
	
	Note
	- This parameter is available only for the development environment of a workspace in standard mode.
	- **Task owner** can be an Alibaba Cloud account or a RAM user.
	  Before you use the **Shortcut mode** to associate an EMR cluster with a workspace, you must attach the AliyunEMRDevelopAccess policy to workspace roles such as developers and administrators. This way, the roles can be used to create and run EMR nodes in DataStudio.
	  ■ The AliyunEMRDevelopAccess policy is attached to Alibaba Cloud accounts by default.
	  ■ To run EMR nodes as a RAM user, you must attach the AliyunEMRDevelopAccess policy to the RAM user. For more information, see Grant permissions to RAM users.

Cluster ID	The ID of the EMR cluster that you want to associate with the workspace. Select an ID from the drop-down list. The EMR cluster with the selected ID is used as the runtime environment of EMR nodes.
Project ID	The ID of the EMR project that you want to associate with the workspace. Select an ID from the drop-down list. The EMR project with the selected ID is used as the runtime environment of EMR nodes. **Note** The IDs of the EMR projects that are not in **Security mode** are not displayed and cannot be selected.
YARN resource queue	The name of the resource queue in the EMR cluster. Unless otherwise specified, set this parameter to *default*.

Parameter	Description
default task submit node	The node from which an EMR job is submitted to the EMR cluster. If your EMR cluster is associated with a gateway cluster, set this parameter to the associated gateway cluster. Otherwise, set this parameter to the default value **HEADER**. **Note** If you set this parameter to the associated gateway cluster, all EMR nodes in the workspace that is associated with the current EMR cluster use the associated gateway cluster to submit EMR jobs by default. If a node does not need to submit EMR jobs by using the gateway cluster, you can add the **USE_GATEWAY** parameter to the advanced configurations of the node and set the parameter to **False**. For more information about the advanced configurations of EMR nodes, see the topics related to EMR nodes.
Endpoint	The endpoint of the EMR cluster. The value of this parameter cannot be changed.

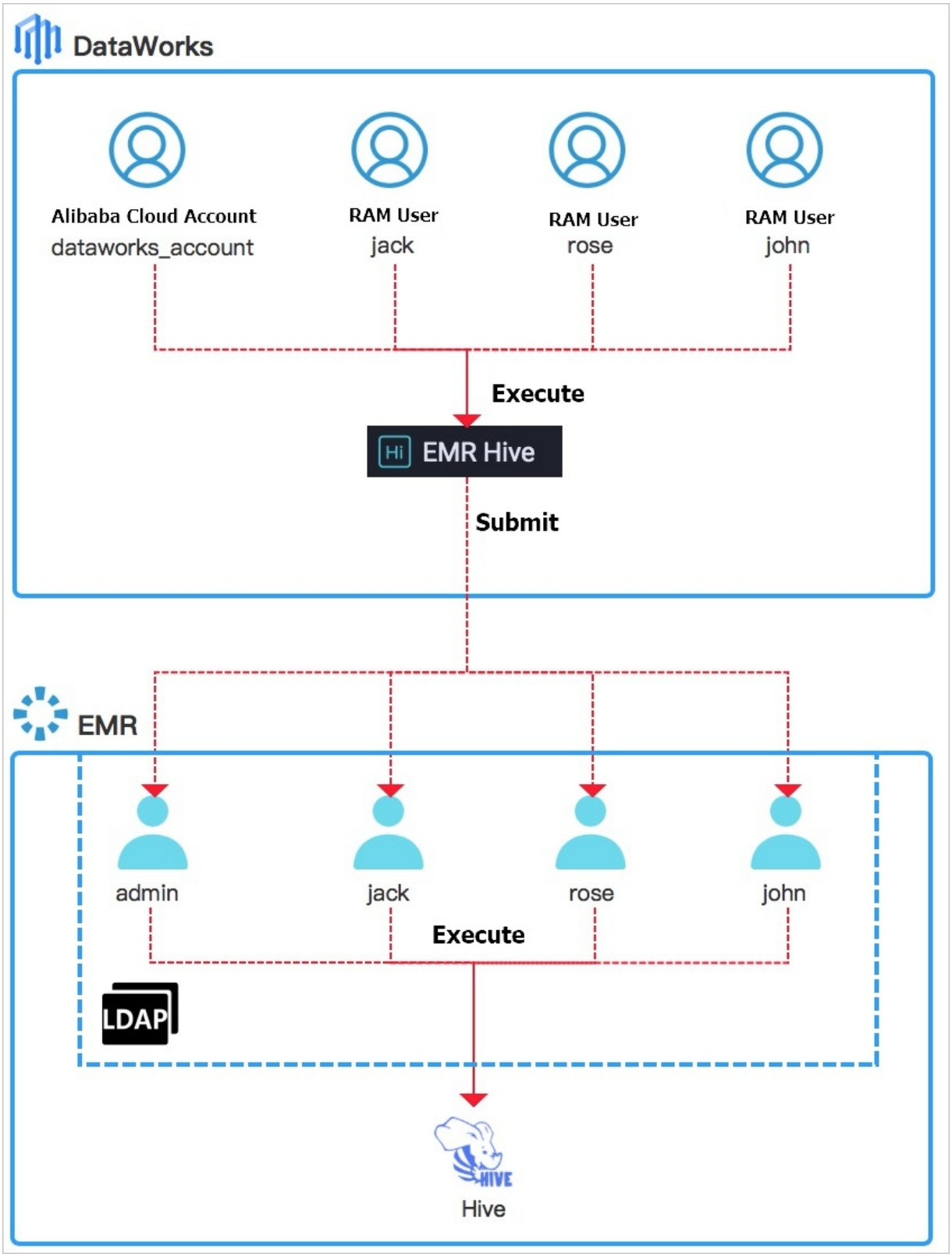
5. Select a resource group.
   i. Select an exclusive resource group for scheduling that connects to the DataWorks workspace. If no exclusive resource group for scheduling is available, create one. For more information about how to create an exclusive resource group for scheduling and configure network connectivity, see Create and use an exclusive resource group for scheduling.
   ii. Click **Test Connectivity** to verify the network connectivity between the exclusive resource group for scheduling and the EMR cluster.
6. Click **Confirm**.

Security mode

In **Security mode**, if you commit the code of EMR nodes by using an Alibaba Cloud account or as a RAM user to an EMR cluster, the code is run by a user that has the same name as the Alibaba Cloud account or RAM user. EMR Ranger can be used to manage the permissions of each Hadoop user in the EMR cluster. This ensures that different Alibaba Cloud accounts, node owners, or RAM users have different data permissions when they run EMR nodes in DataWorks. This provides higher data security.

> **Note**
>
> Before you use the **Security mode** to associate an EMR cluster with a workspace, you must add the credentials of workspace roles such as developers and administrators to the Lightweight Directory Access Protocol (LDAP) directory of the EMR cluster. In addition, you must attach the AliyunEMRDevelopAccess or AliyunEMRFullAccess policy and grant relevant data permissions to the workspace roles. This way, the roles can be used to create and run EMR nodes in DataStudio.
>
> - The credentials of Alibaba Cloud accounts are in the LDAP directory of the EMR cluster by default. The AliyunEMRDevelopAccess and AliyunEMRFullAccess policies are also attached to Alibaba Cloud accounts by default.
> - To run EMR nodes as a RAM user, you must add the credential of the RAM user to the LDAP directory of the EMR cluster. For more information, see the *Add the credentials of specific RAM users to the LDAP directory of the EMR cluster* step. In addition, you must attach the AliyunEMRDevelopAccess or AliyunEMRFullAccess policy to the RAM user. For more information, see Grant permissions to RAM users.

The **Security mode** is suitable for workspaces that require strict management and isolation of data permissions for users who run nodes.



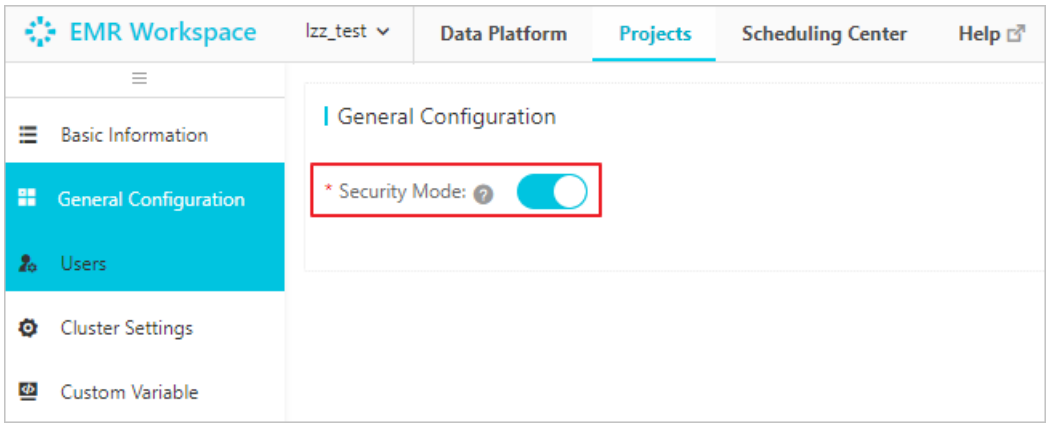
To use the **Security mode** to associate an EMR cluster with a workspace, perform the following steps:

1. Turn on Security Mode for the EMR project.
   i. Log on to the EMR console.
   ii. In the top navigation bar, click **Data Platform**.

iii. In the **Projects** section, find the project for which you want to enable the Security mode and click **Edit Job** in the Actions column.

iv. On the page that appears, click the **Projects** tab in the top navigation bar.

v. In the left-side navigation pane, click **General Configuration**. On the General Configuration page, turn on **Security Mode**.



2. Add the credentials of specific RAM users to the LDAP directory of the EMR cluster.

i. Return to the homepage of the EMR console. In the top navigation bar, click **Cluster Management**.

ii. Find the cluster that you want to manage and click **Details** in the Actions column.

iii. In the left-side navigation pane, click **Users**.

iv. On the **Users** page, click **Add User**.

v. In the **Add User** dialog box, set the parameters.

We recommend that you add the credentials of the following RAM users to the LDAP directory of the EMR cluster:

- RAM users that create, test, and run EMR nodes in DataStudio
- RAM users that create, commit, and deploy EMR nodes in DataStudio

vi. Click **OK**.

3. Configure EMR Ranger and manage the permissions of the Hadoop users that correspond to your Alibaba Cloud account and RAM users. For more information, see Integrate Ranger UserSync with an LDAP server and Integrate components with Ranger.

4. Associate the EMR cluster with the current DataWorks workspace.

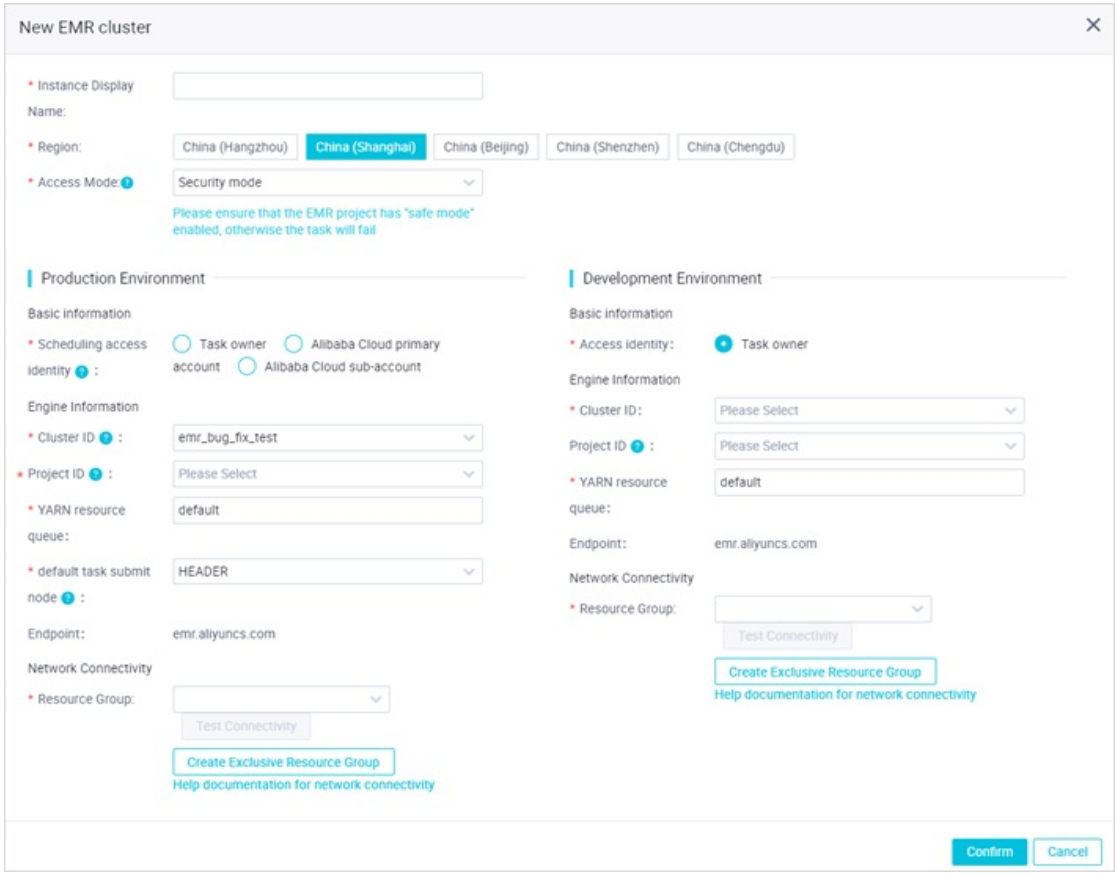
i. Go to the Workspace Management page.

ii. In the **Compute Engine Information** section, click the **E-MapReduce** tab.

iii. On the **E-MapReduce** tab, click **Add Instance**.

iv. In the **New EMR cluster** dialog box, set the parameters.

Parameters in the New EMR cluster dialog box vary based on the mode in which your DataWorks workspace runs. The following table describes the parameters for a DataWorks workspace in standard mode. You must set the parameters for both the production environment and the development environment.



Parameter	Description
Instance Display Name	The display name of the EMR compute engine instance.
Region	The region of the current workspace.
Access Mode	The access mode of the EMR cluster. Select **Security mode** from the drop-down list and click **Confirm** in the **Please note** message. **Note** You cannot use multiple modes to associate an EMR cluster with a DataWorks workspace at the same time. Proceed with caution when you change the access mode of the EMR cluster because a mode change leads to permission changes.

Parameter	Description
Scheduling access identity	The identity that is used to commit the code of an EMR node to the EMR cluster. The code is committed when the node is committed and deployed to the DataWorks scheduling system in the production environment. The Hadoop user that corresponds to this identity runs the code. Valid values: **Task owner**, **Alibaba Cloud primary account**, and **Alibaba Cloud sub-account**. ■ **Task owner**: commits and runs the code of an EMR node as the node owner. If you select this value, the data permissions of Hadoop users are isolated. **Task owner** can be an Alibaba Cloud account or a RAM user. ■ **Alibaba Cloud primary account**: commits the code of an EMR node to the EMR cluster by using an Alibaba Cloud account. ■ **Alibaba Cloud sub-account**: commits the code of an EMR node to the EMR cluster as a RAM user. **Note** ■ This parameter is available only for the production environment. ■ The credentials of Alibaba Cloud accounts are in the LDAP directory of the EMR cluster by default. The AliyunEMRDevelopAccess and AliyunEMRFullAccess policies are also attached to Alibaba Cloud accounts by default. ■ To run EMR nodes as a RAM user, you must add the credential of the RAM user to the LDAP directory of the EMR cluster. For more information, see the *Add the credentials of specific RAM users to the LDAP directory of the EMR cluster* step. In addition, you must attach the AliyunEMRDevelopAccess or AliyunEMRFullAccess policy to the RAM user. For more information, see Grant permissions to RAM users.

Parameter	Description
Access identity	The identity that is used to commit the code of an EMR node in the development environment to the EMR cluster. Default value: **Task owner**. The Hadoop user that corresponds to the user who runs the node runs the code. **Note** ■ This parameter is available only for the development environment of a workspace in standard mode. ■ Make sure that the credential of the user who runs the node is added to the LDAP directory of the EMR cluster. In addition, make sure that the AliyunEMRDevelopAccess or AliyunEMRFulAccess policy is attached to the user and relevant data permissions are granted to the user. This way, the user can run EMR nodes in DataStudio. **Task owner** can be an Alibaba Cloud account or a RAM user. ■ The credentials of Alibaba Cloud accounts are in the LDAP directory of the EMR cluster by default. The AliyunEMRDevelopAccess and AliyunEMRFullAccess policies are also attached to Alibaba Cloud accounts by default. ■ To run EMR nodes as a RAM user, you must add the credential of the RAM user to the LDAP directory of the EMR cluster. For more information, see the *Add the credentials of specific RAM users to the LDAP directory of the EMR cluster* step. In addition, you must attach the AliyunEMRDevelopAccess or AliyunEMRFullAccess policy to the RAM user. For more information, see Grant permissions to RAM users.
Cluster ID	The ID of the EMR cluster that you want to associate with the workspace. Select an ID from the drop-down list. The EMR cluster with the selected ID is used as the runtime environment of EMR nodes.
Project ID	The ID of the EMR project that you want to associate with the workspace. Select the ID of an EMR project in Security mode from the drop-down list. **Note** The IDs of the EMR projects that are not in **Security mode** are not displayed and cannot be selected.
YARN resource queue	The name of the resource queue in the EMR cluster. Unless otherwise specified, set this parameter to *default*.

Parameter	Description
default task submit node	The node from which an EMR job is submitted to the EMR cluster. If your EMR cluster is associated with a gateway cluster, set this parameter to the associated gateway cluster. Otherwise, set this parameter to the default value **HEADER**. **Note** If you set this parameter to the associated gateway cluster, all EMR nodes in the workspace that is associated with the current EMR cluster use the associated gateway cluster to submit EMR jobs by default. If a node does not need to submit EMR jobs by using the gateway cluster, you can add the **USE_GATEWAY** parameter to the advanced configurations of the node and set the parameter to **False**. For more information about the advanced configurations of EMR nodes, see the topics related to EMR nodes.
Endpoint	The endpoint of the EMR cluster. The value of this parameter cannot be changed.

v. Select a resource group.

a. Select an exclusive resource group for scheduling that connects to the DataWorks workspace. If no exclusive resource group for scheduling is available, create one. For more information about how to create an exclusive resource group for scheduling and configure network connectivity, see Create and use an exclusive resource group for scheduling.

b. Click **Test Connectivity** to verify the network connectivity between the exclusive resource group for scheduling and the EMR cluster.

vi. Click **Confirm**.

5. Configure mappings between Alibaba Cloud accounts or RAM users and LDAP accounts.

After the EMR cluster in **Security mode** is associated with your DataWorks workspace, the LDAP account that maps the identity specified by the Access identity parameter is used to run EMR nodes. The Access identity parameter is set when you associate the EMR cluster with your DataWorks workspace. You must configure mappings between Alibaba Cloud accounts or RAM users and LDAP accounts on the **EMR Cluster Configuration** page.

i. After the EMR cluster is associated with your DataWorks workspace, the **Please note** message appears. In the Please note message, click **Configure Development Environment** and **Configure Production Environment** to configure mappings between Alibaba Cloud accounts or RAM users and LDAP accounts.

ii. On the **EMR Cluster Configuration** page, click **Edit** in the upper-right corner of the EMR cluster that is associated with your DataWorks workspace.

iii. In the **Edit EMR Cluster Configuration** dialog box, configure mappings between Alibaba Cloud accounts or RAM users and LDAP accounts. You can use one of the following methods to configure the mappings:

- Reference an existing mapping: You can reference an existing mapping in the current workspace.
- Create a mapping: In the **Configure Account Mapping** section, specify the Alibaba Cloud account and LDAP account between which you want to configure a mapping, and enter the password of the LDAP account.

> **Note**
>
> - An Alibaba Cloud account or a RAM user to whom the **AliyunEMRFullAccess** policy is attached can configure mappings for all members of this workspace. Other members of the workspace can configure mappings only for themselves.
> - You can add mappings between multiple Alibaba Cloud accounts and LDAP accounts. A single LDAP account can be mapped to multiple Alibaba Cloud accounts in DataWorks.

iv. Click **Confirm**. The mappings are configured.

5.4.3. Create an EMR Presto node

This topic describes how to create an E-MapReduce (EMR) Presto node. EMR Presto nodes allow you to perform interactive analysis and queries on large amounts of structured and unstructured data.

Prerequisites

- An Alibaba Cloud EMR cluster is created, and an inbound rule that contains the following content is added to the security group to which the cluster belongs.
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- An EMR compute engine instance is associated with your workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Configure a workspace.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR Hive nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned when you run EMR Hive nodes.
  i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

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

> **Note** In the preceding code, `ALISA.*` and `SKYNET.*` are specific to DataWorks.

ii. After the whitelist configurations are modified, you must restart the Hive service to make the configurations take effect. For more information, see Restart a service.

- An exclusive resource group for scheduling is created, and the resource group is associated with the virtual private cloud (VPC) where the EMR cluster resides. For more information, see Create and use an exclusive resource group for scheduling.

> **Note** You can use only exclusive resource groups for scheduling to run EMR Hive nodes.

Limits

- You are not allowed to create EMR Presto nodes on Alibaba Gov Cloud and Alibaba Finance Cloud platforms.
- You can use only exclusive resource groups for scheduling to run EMR Presto nodes.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the DataStudio page, move the pointer over the **Create** icon and choose **EMR > EMR Presto**.

   Alternatively, find your workflow, right-click the EMR folder in the workflow, and then choose **Create > EMR Presto**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the configuration tab of the EMR Presto node, write code for the node.

   Sample SQL statement for the node:

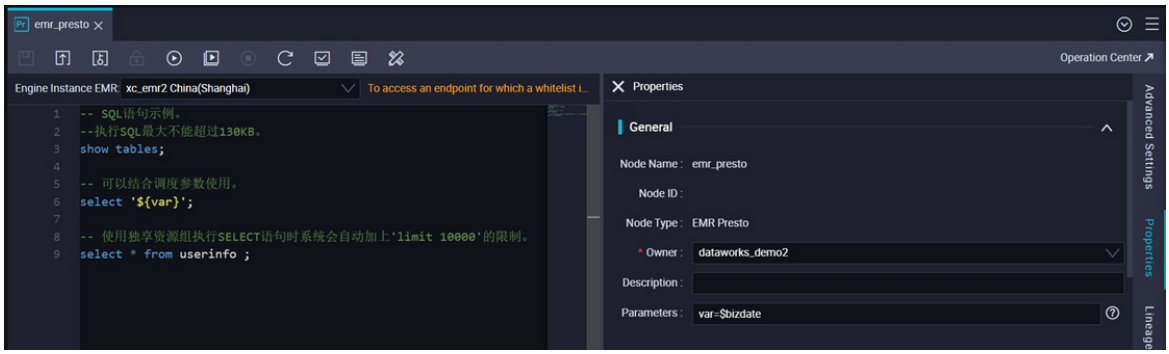
```
show tables;
select '${var}'; // You can assign a specific scheduling parameter to the var variable.

select * from userinfo ;
```

> **Note**
>
> - The size of the SQL statement for the node cannot exceed 130 KB.
> - If you use an EMR Presto node to query data, a maximum of 10,000 data records can be returned and the total size of the returned data records cannot exceed 10 MB.
> - If multiple EMR compute engine instances are associated with the current workspace, you must select one from the compute engine instances based on your business requirements. If only one EMR compute engine instance is associated with the current workspace, you do not need to select a compute engine instance.

For more information about scheduling parameters, see Overview of scheduling parameters.

If you want to change the scheduling parameter that is assigned to the variable in the code, click **Run with Parameters** in the top toolbar. For more information about the assignment logic of scheduling parameters, see What are the differences in the value assignment logic of scheduling parameters among the Run, Run with Parameters, and Perform Smoke Testing in Development Environment modes?.



For more information about how to configure a Presto SQL job, see Configure a Presto SQL job.

6. In the right-side navigation pane, click **Advanced Settings**. In the Advanced Settings panel, configure the parameters.
   - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters for running Spark jobs. You can configure multiple parameters in the --conf xxx=xxx format.
   - "queue": the scheduling queue to which jobs are committed. Default value: default.
   - "vcores": the number of vCPUs. Default value: 1.
   - "memory": the memory that is allocated to the launcher. Unit: MB. Default value: 2048.
   - "priority": the priority. Default value: 1.
   - "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. The value false indicates that only one SQL statement is executed at a time. The value true indicates that multiple SQL statements are executed at a time.
   - "USE_GATEWAY": specifies whether a gateway cluster is used to commit jobs on the current node. The value **true** indicates that a gateway cluster is used to commit jobs. The value **false** indicates that a gateway cluster is not used to commit jobs and jobs are committed to the header node by default.

     > **Note** If the EMR cluster to which the node belongs is not associated with a gateway cluster but you set the USE_GATEWAY parameter to **true**, jobs may fail to be committed.

7. In the right-side navigation pane, click **Properties**. In the Properties panel, configure properties for the EMR Presto node.
   -
   -

   For more information, see Configure basic properties.
8. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.
9. Test the node. For more information, see View auto triggered nodes.

5.4.4. Create an EMR Hive node

This topic describes how to create an E-MapReduce (EMR) Hive node. EMR Hive nodes allow you to use SQL-like statements to read data from and write data to data warehouses with large volumes of data stored in a distributed storage system and manage these data warehouses. You can use EMR Hive nodes to analyze large amounts of log data in an efficient manner.

Prerequisites

- An Alibaba Cloud EMR cluster is created, and an inbound rule that contains the following content is added to the security group to which the cluster belongs.
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- An EMR compute engine instance is associated with your workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Configure a workspace.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR Hive nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned when you run EMR Hive nodes.

  i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

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

> **Note** In the preceding code, `ALISA.*` and `SKYNET.*` are specific to DataWorks.

ii. After the whitelist configurations are modified, you must restart the Hive service to make the configurations take effect. For more information, see Restart a service.

- An exclusive resource group for scheduling is created, and the resource group is associated with the virtual private cloud (VPC) where the EMR cluster resides. For more information, see Create and use an exclusive resource group for scheduling.

> **Note** You can use only exclusive resource groups for scheduling to run EMR Hive nodes.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the DataStudio page, move the pointer over the **Create** icon and choose **EMR > EMR Hive**.

   Alternatively, find your workflow, right-click the EMR folder in the workflow, and then choose **Create > EMR Hive**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the configuration tab of the EMR Hive node, write code for the node.

   Sample SQL statement for the node:

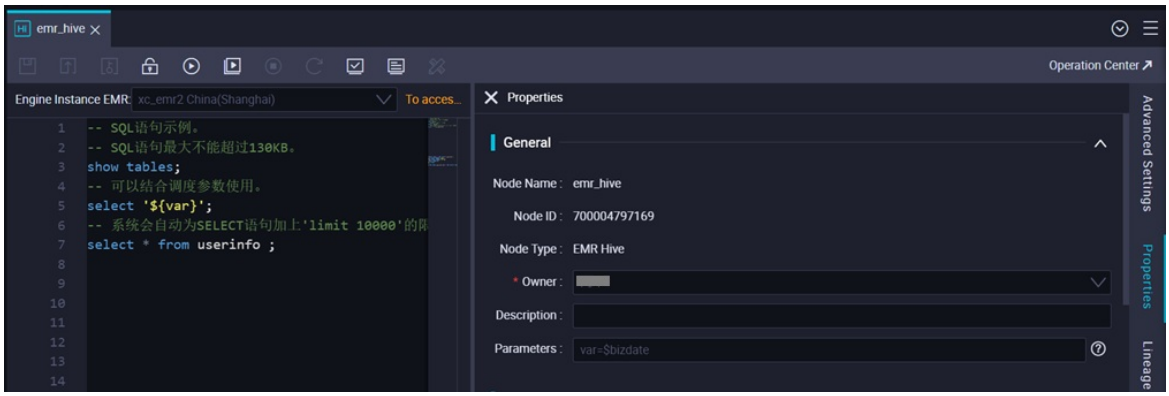
```
show tables;
select '${var}'; // You can assign a specific scheduling parameter to the var variable.

select * from userinfo ;
```

> **Note**
>
> - The size of the SQL statement for the node cannot exceed 130 KB.
> - If you use an EMR Hive node to query data, a maximum of 10,000 data records can be returned and the total size of the returned data records cannot exceed 10 MB.
> - If multiple EMR compute engine instances are associated with the current workspace, you must select one from the compute engine instances based on your business requirements. If only one EMR compute engine instance is associated with the current workspace, you do not need to select a compute engine instance.

For more information about scheduling parameters, see Overview of scheduling parameters.

If you want to change the scheduling parameter that is assigned to the variable in the code, click **Run with Parameters** in the top toolbar. For more information about the assignment logic of scheduling parameters, see What are the differences in the value assignment logic of scheduling parameters among the Run, Run with Parameters, and Perform Smoke Testing in Development Environment modes?.



For more information, see Configure a Hive SQL job.

6. In the right-side navigation pane, click **Advanced Settings**. In the Advanced Settings panel, configure the parameters.
   - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters for running Spark jobs. You can configure multiple parameters in the --conf xxx=xxx format.
   - "queue": the scheduling queue to which jobs are committed. Default value: default.
   - "vcores": the number of vCPUs. Default value: 1.
   - "memory": the memory that is allocated to the launcher. Unit: MB. Default value: 2048.
   - "priority": the priority. Default value: 1.
   - "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. The value false indicates that only one SQL statement is executed at a time. The value true indicates that multiple SQL statements are executed at a time.
   - "USE_GATEWAY": specifies whether a gateway cluster is used to commit jobs on the current node. The value **true** indicates that a gateway cluster is used to commit jobs. The value **false** indicates that a gateway cluster is not used to commit jobs and jobs are committed to the header node by default.

> **Note** If the EMR cluster to which the node belongs is not associated with a gateway cluster but you set the USE_GATEWAY parameter to **true**, jobs may fail to be committed.

7. In the right-side navigation pane, click **Properties**. In the Properties panel, configure properties for the EMR Presto node.
   -
   -

   For more information, see Configure basic properties.
8. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.
9. Test the node. For more information, see View auto triggered nodes.

5.4.5. Create and use an EMR MR node

You can create an E-MapReduce (EMR) MR node to process a large dataset by using multiple parallel map tasks. This way, you can perform parallel computing on large datasets. This topic describes how to create an EMR MR node. It also describes the development process of an EMR MR node. In this process, an Object Storage Service (OSS) bucket is used, and the number of words in an OSS object is calculated by using the EMR MR node.

Prerequisites

- An Alibaba Cloud EMR cluster is created, and an inbound rule that contains the following content is added to the security group to which the cluster belongs.
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- An EMR compute engine instance is associated with your workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Configure a workspace.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR Hive nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned when you run EMR Hive nodes.

i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

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

> **Note** In the preceding code, `ALISA.*` and `SKYNET.*` are specific to DataWorks.

ii. After the whitelist configurations are modified, you must restart the Hive service to make the configurations take effect. For more information, see Restart a service.

- An exclusive resource group for scheduling is created, and the resource group is associated with the virtual private cloud (VPC) where the EMR cluster resides. For more information, see Create and use an exclusive resource group for scheduling.

  > **Note** You can use only exclusive resource groups for scheduling to run EMR Hive nodes.

- Open source code is uploaded as an **EMR JAR resource**. You can reference the open source code in your EMR MR node. .
- User-defined functions (UDFs) are uploaded as **EMR JAR resources** and are registered with EMR. You can reference the UDFs in your EMR MR node. For more information about how to register a UDF function, see Create an EMR function.
- The following operations are performed if you use the node development example in this topic to run the EMR MR node:
  - An OSS bucket is created. For more information, see Create buckets.
  - A Java project is created by using IntelliJ IDEA.

Context

This topic describes the node development process of an EMR MR node. In this process, an OSS bucket is used, and the number of words in an OSS object is calculated by using the EMR MR node. You must replace the information involved in the development process with the actual information. The information includes the object name, bucket name, and directory.

Prepare initial data and a JAR resource package

1. Prepare initial data.

   Create the *input01.txt* file that contains the following initial data:

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

2. Create directories that are used to store initial data and JAR resources.

   i. Log on to the OSS console.

   ii. In the left-side navigation pane, click **Buckets**.

   iii. On the Buckets page, find the desired bucket and click the bucket name to go to the **Files** page.

   In this example, the *onaliyun-bucket-2* bucket is used.

iv. On the Files page, click **Create Folder** to create directories that are used to store initial data and JAR resources.

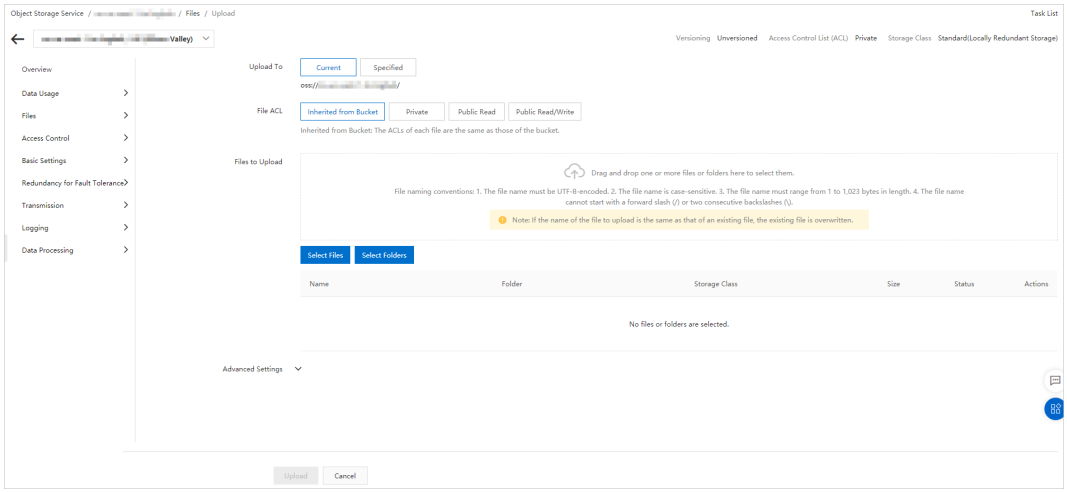
- Set **Folder Name** to *emr/datas/wordcount02/inputs* to create a directory that is used to store initial data.
- Set **Folder Name** to *emr/jars* to create a directory that is used to store JAR resources.

v. Upload the file that stores the initial data to the emr/datas/wordcount02/inputs directory.

a. Go to the *emr/datas/wordcount02/inputs* directory.

b. Click **Upload**.

c. In the **Files to Upload** section, click **Select Files** and upload the *input01.txt* file to the bucket.



d. Click **Upload**.

3. Use the EMR MR node to read the OSS object and generate a JAR package.

i. Open the created IntelliJ IDEA project and add Project Object Model (POM) dependencies.

```
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-mapreduce-client-common</artifactId>
            <version>2.8.5</version> <!--The version of the EMR MR node is V2.8.5.-
->
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>2.8.5</version>
        </dependency>
```

ii. Configure the following parameters to read data from and write data to the OSS object:

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

Parameter description:

- `${accessKeyId}`: the AccessKey ID of your Alibaba Cloud account.

- `${accessKeySecret}`: the AccessKey secret of your Alibaba Cloud account.
- `${endpoint}`: the endpoint of OSS. The endpoint is determined by the region where your EMR cluster resides. You must activate OSS in the region where your EMR cluster resides. For more information, see Regions and endpoints.

In the following code, the WordCount program of Hadoop is modified. The AccessKey ID and AccessKey secret are added to the code. This way, the EMR MR node has permissions to access the OSS object.

```
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. Compress the preceding code into a JAR package after you write the code. In this example, the *onaliyun_mr_wordcount-1.0-SNAPSHOT.jar.* JAR package is generated.

Create and use an EMR MR node

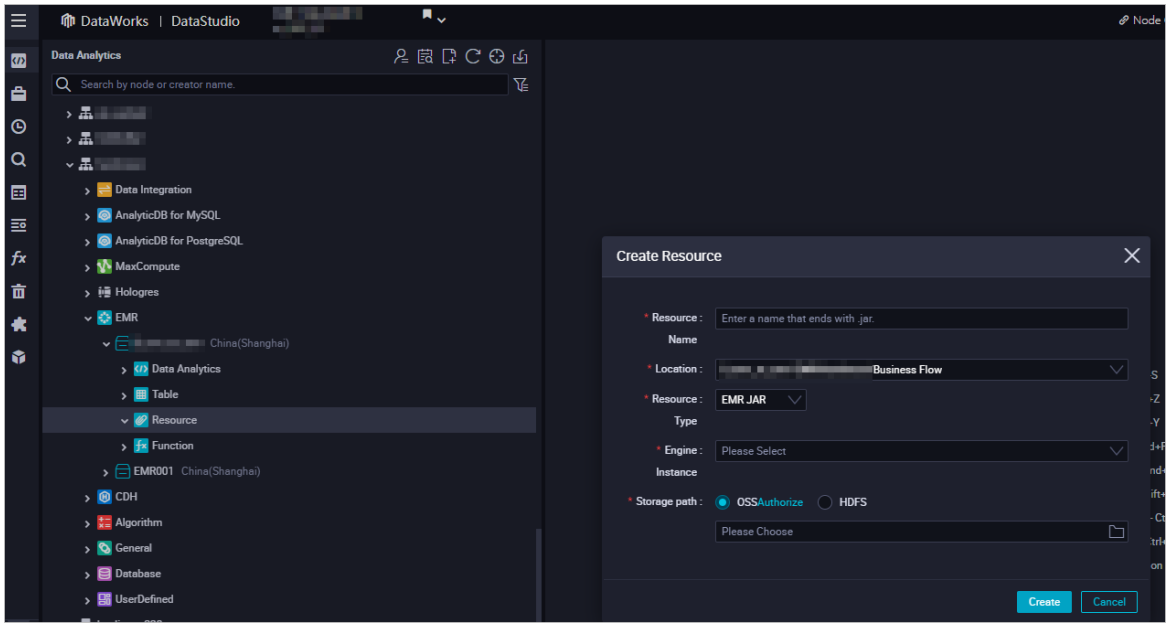
This topic describes the node development process of an EMR MR node. In this process, an OSS bucket is used, and the number of words in an OSS object is calculated by using the EMR MR node.

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the DataStudio page, move the pointer over the **Create** icon and choose **EMR > EMR MR**.

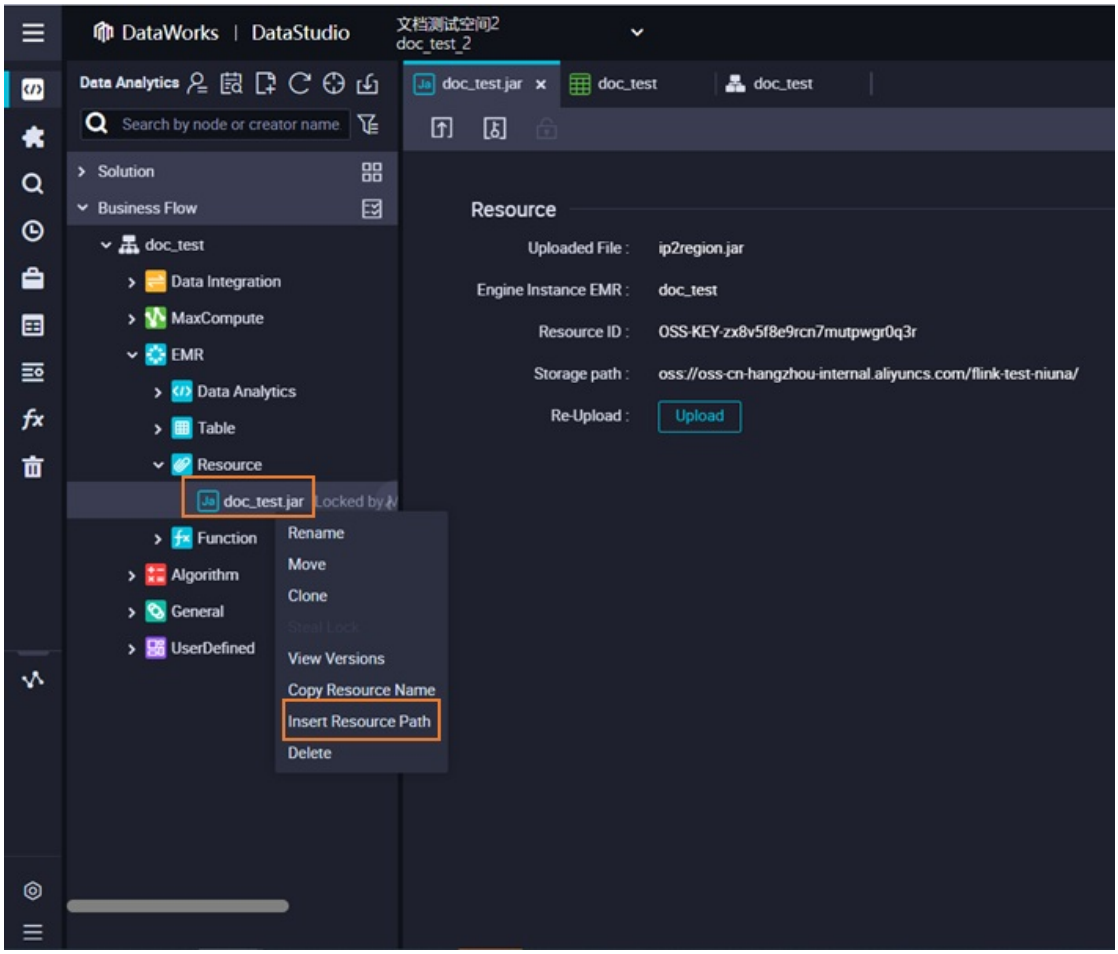
   Alternatively, you can find the required workflow, right-click the workflow name, and then choose **Create > EMR > EMR MR**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. Create an EMR JAR resource.

   In this step, the JAR package is stored in the *emr/jars* directory. If this is the first time you use DataWorks to access OSS, click **Authorize** next to OSS to authorize DataWorks and EMR to access OSS.



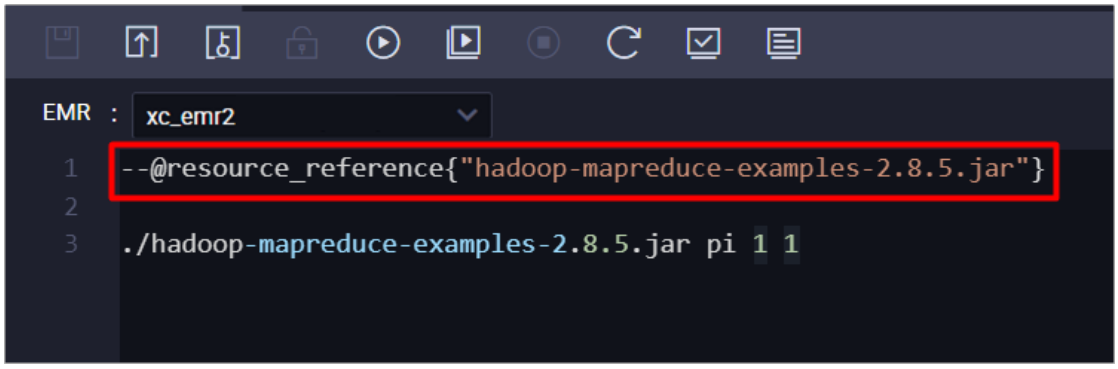
6. Reference the EMR JAR resource.
   i. Open the **EMR MR** node. The configuration tab of the node appears.
   ii. Find the resource that you want to reference under **Resource** in the **EMR** folder, right-click the resource name, and then select **Insert Resource Path**. In this step, the resource that will be referenced is *onaliyun_mr_wordcount-1.0-SNAPSHOT.jar*..



iii. The code resource is referenced if the message shown in the following figure appears on the configuration tab of the **EMR MR** node. Then, run the following code.

You must replace the information in the following code with the actual information. The information includes the resource package name, bucket name, and directory.

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



7. In the right-side navigation pane, click **Advanced Settings**. In the Advanced Settings panel, configure the parameters.
   - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters for running Spark jobs. You can configure multiple parameters in the --conf xxx=xxx format.
   - "queue": the scheduling queue to which jobs are committed. Default value: default.
   - "vcores": the number of vCPUs. Default value: 1.
   - "memory": the memory that is allocated to the launcher. Unit: MB. Default value: 2048.
   - "priority": the priority. Default value: 1.
   - "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. The value false indicates that only one SQL statement is executed at a time. The value true indicates that multiple SQL statements are executed at a time.
   - "USE_GATEWAY": specifies whether a gateway cluster is used to commit jobs on the current node. The value **true** indicates that a gateway cluster is used to commit jobs. The value **false** indicates that a gateway cluster is not used to commit jobs and jobs are committed to the header node by default.

   > **Note** If the EMR cluster to which the node belongs is not associated with a gateway cluster but you set the USE_GATEWAY parameter to **true**, jobs may fail to be committed.

8. In the right-side navigation pane, click **Properties**. In the Properties panel, configure properties for the EMR Presto node.
   -
   -

   For more information, see Configure basic properties.
9. Save and commit the node.

> **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

i. Click the icon in the toolbar to save the node.

ii. Click the icon in the toolbar.

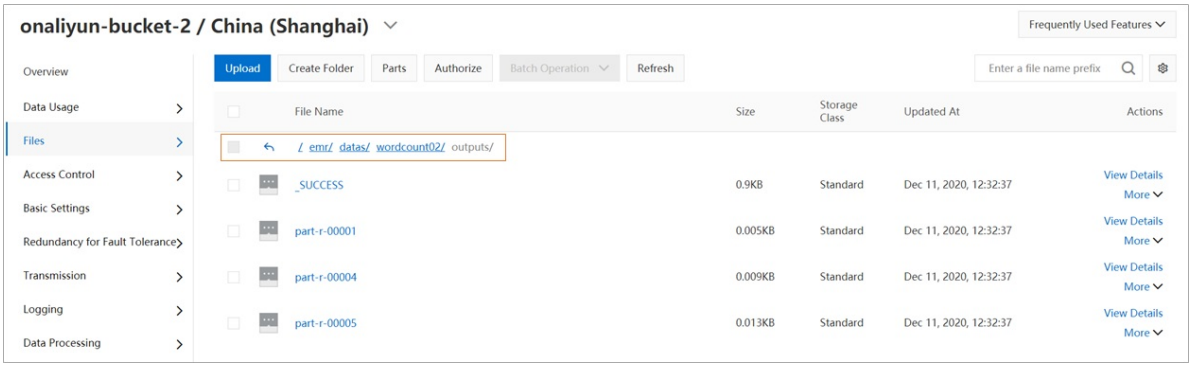
iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iv. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.

View the result

- Log on to the OSS console. Then, you can view the result in the *emr/datas/wordcount02/inputs* directory in which the initial data is stored.



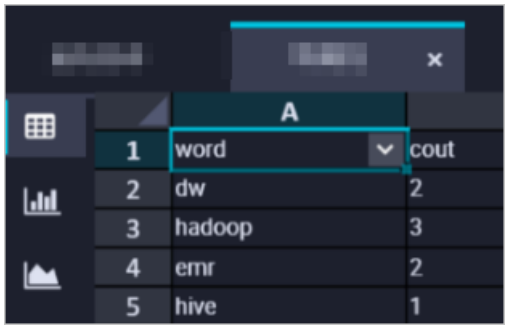
- View the statistical result in the DataWorks console.

i. Create an EMR Hive node. For more information, see Create an EMR Hive node.

ii. Create Hive external tables that are mounted to OSS in the EMR Hive node. Sample code:

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

The following figure shows the result.



5.4.6. Create an EMR Spark SQL node

This topic describes how to create an E-MapReduce (EMR) Spark SQL node. EMR Spark SQL nodes allow you to use the distributed SQL query engine to process structured data. This improves job efficiency.

Prerequisites

- An Alibaba Cloud EMR cluster is created, and an inbound rule that contains the following content is added to the security group to which the cluster belongs.
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- An EMR compute engine instance is associated with your workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Configure a workspace.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR Hive nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned when you run EMR Hive nodes.
  i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

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

  > **Note** In the preceding code, `ALISA.*` and `SKYNET.*` are specific to DataWorks.

  ii. After the whitelist configurations are modified, you must restart the Hive service to make the configurations take effect. For more information, see Restart a service.
- An exclusive resource group for scheduling is created, and the resource group is associated with the virtual private cloud (VPC) where the EMR cluster resides. For more information, see Create and use an exclusive resource group for scheduling.

  > **Note** You can use only exclusive resource groups for scheduling to run EMR Hive nodes.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the DataStudio page, move the pointer over the **+Create** icon and choose **EMR > EMR Spark**

SQL.

Alternatively, find your workflow, right-click the EMR folder in the workflow, and then choose **Create > EMR Spark SQL**.

3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. On the configuration tab of the EMR Spark SQL node, write code for the node.

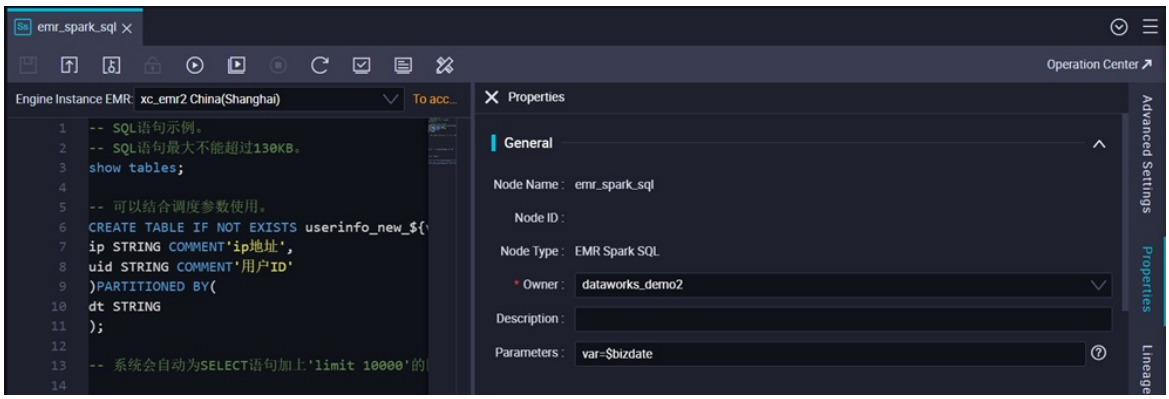
   Sample SQL statement for the node:

```
show tables;
CREATE TABLE IF NOT EXISTS userinfo_new_${var} (
ip STRING COMMENT'IP address',
uid STRING COMMENT'User ID'
)PARTITIONED BY(
dt STRING
); // You can assign a specific scheduling parameter to the var variable.
```

   > **Note**
   > - The size of the SQL statement for the node cannot exceed 130 KB.
   > - If you use an EMR Spark SQL node to query data, a maximum of 10,000 data records can be returned and the total size of the returned data records cannot exceed 10 MB.
   > - If multiple EMR compute engine instances are associated with the current workspace, you must select one from the compute engine instances based on your business requirements. If only one EMR compute engine instance is associated with the current workspace, you do not need to select a compute engine instance.

   For more information about scheduling parameters, see Overview of scheduling parameters.

   If you want to change the scheduling parameter that is assigned to the variable in the code, click **Run with Parameters** in the top toolbar. For more information about the assignment logic of scheduling parameters, see What are the differences in the value assignment logic of scheduling parameters among the Run, Run with Parameters, and Perform Smoke Testing in Development Environment modes?.



   For more information, see Configure a Spark SQL job.

6. In the right-side navigation pane, click **Advanced Settings**. In the Advanced Settings panel, configure the parameters.
   - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters for running Spark jobs. You can configure multiple parameters in the --conf xxx=xxx format.
   - "queue": the scheduling queue to which jobs are committed. Default value: default.
   - "vcores": the number of vCPUs. Default value: 1.
   - "memory": the memory that is allocated to the launcher. Unit: MB. Default value: 2048.
   - "priority": the priority. Default value: 1.
   - "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. The value false indicates that only one SQL statement is executed at a time. The value true indicates that multiple SQL statements are executed at a time.
   - "USE_GATEWAY": specifies whether a gateway cluster is used to commit jobs on the current node. The value **true** indicates that a gateway cluster is used to commit jobs. The value **false** indicates that a gateway cluster is not used to commit jobs and jobs are committed to the header node by default.

     > **Note** If the EMR cluster to which the node belongs is not associated with a gateway cluster but you set the USE_GATEWAY parameter to **true**, jobs may fail to be committed.

7. In the right-side navigation pane, click **Properties**. In the Properties panel, configure properties for the EMR Presto node.
   -
   -

   For more information, see Configure basic properties.
8. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.
9. Test the node. For more information, see View auto triggered nodes.

5.4.7. Create and use an EMR Spark node

E-MapReduce (EMR) Spark nodes allow you to perform complex memory data analysis and help you build large, low-latency data analysis applications. This topic describes how to create an EMR Spark node. It also provides two examples to demonstrate the features of an EMR Spark node.

Prerequisites

- An Alibaba Cloud EMR cluster is created, and an inbound rule that contains the following content is added to the security group to which the cluster belongs.
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- An EMR compute engine instance is associated with your workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Configure a workspace.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR Hive nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned when you run EMR Hive nodes.
  i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

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

  > **Note** In the preceding code, `ALISA.*` and `SKYNET.*` are specific to DataWorks.

  ii. After the whitelist configurations are modified, you must restart the Hive service to make the configurations take effect. For more information, see Restart a service.
- An exclusive resource group for scheduling is created, and the resource group is associated with the virtual private cloud (VPC) where the EMR cluster resides. For more information, see Create and use an exclusive resource group for scheduling.

  > **Note** You can use only exclusive resource groups for scheduling to run EMR Hive nodes.

Create an EMR Spark node

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the DataStudio page, move the pointer over the **+Create** icon and choose **EMR > EMR Spark**.

   Alternatively, you can find the required workflow, right-click the workflow name, and then choose **Create > EMR > EMR Spark**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. In the right-side navigation pane, click **Advanced Settings**. In the Advanced Settings panel, configure the parameters.
   - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters for running Spark jobs. You can configure multiple parameters in the --conf xxx=xxx format.
   - "queue": the scheduling queue to which jobs are committed. Default value: default.
   - "vcores": the number of vCPUs. Default value: 1.
   - "memory": the memory that is allocated to the launcher. Unit: MB. Default value: 2048.
   - "priority": the priority. Default value: 1.
   - "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. The value false indicates that only one SQL statement is executed at a time. The value true indicates that multiple SQL statements are executed at a time.
   - "USE_GATEWAY": specifies whether a gateway cluster is used to commit jobs on the current node. The value **true** indicates that a gateway cluster is used to commit jobs. The value **false** indicates that a gateway cluster is not used to commit jobs and jobs are committed to the header node by default.

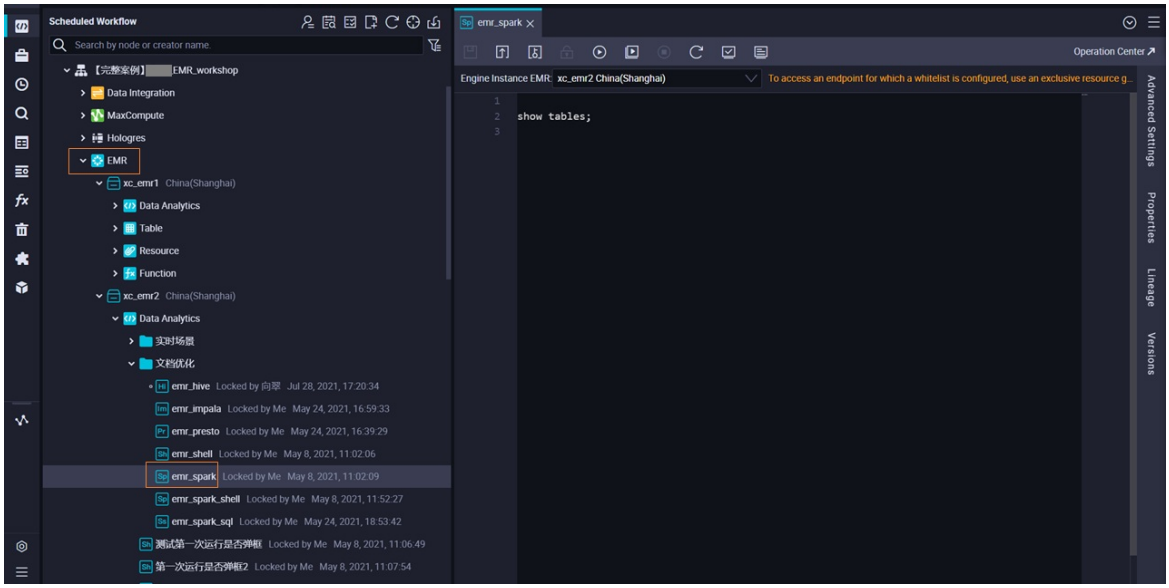
     > **Note** If the EMR cluster to which the node belongs is not associated with a gateway cluster but you set the USE_GATEWAY parameter to **true**, jobs may fail to be committed.

5. In the right-side navigation pane, click **Properties**. In the Properties panel, configure properties for the EMR Presto node.
   -
   -

   For more information, see Configure basic properties.
6. Click **Commit**.

   Then, the configuration tab of the newly created EMR Spark node appears.

   > **Note** If multiple EMR compute engine instances are associated with the current workspace, you must select an EMR compute engine instance. If only one EMR compute engine instance is associated with the current workspace, you do not need to select one.



Save and commit the EMR Spark node

1. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.

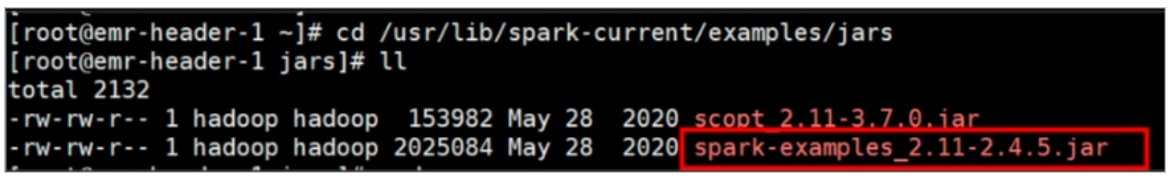
2. Test the node. For more information, see View auto triggered nodes.

Data development example 1: Test whether the EMR Spark node can be successfully run by calculating the Pi value

In this example, the *SparkPi* program of Spark is used to test whether the EMR Spark node can be successfully run. For more information about the sample project of Spark, see Use the sample project.

1. Obtain the storage path of the JAR package *spark-examples_2.11-2.4.5.jar* of the sample project.

   Spark components are installed in the */usr/lib/spark-current* path. You must log on to the EMR console and log on to the master node of the desired EMR cluster to obtain the complete path */usr/lib/spark-current/examples/jars/spark-examples_2.11-2.4.5.jar*. For more information, see Common file paths.



2. On the configuration tab of the EMR Spark node, write code for the node. For more information about how to create an EMR Spark node, see Create an EMR Spark node.

   The following code is used in this example:

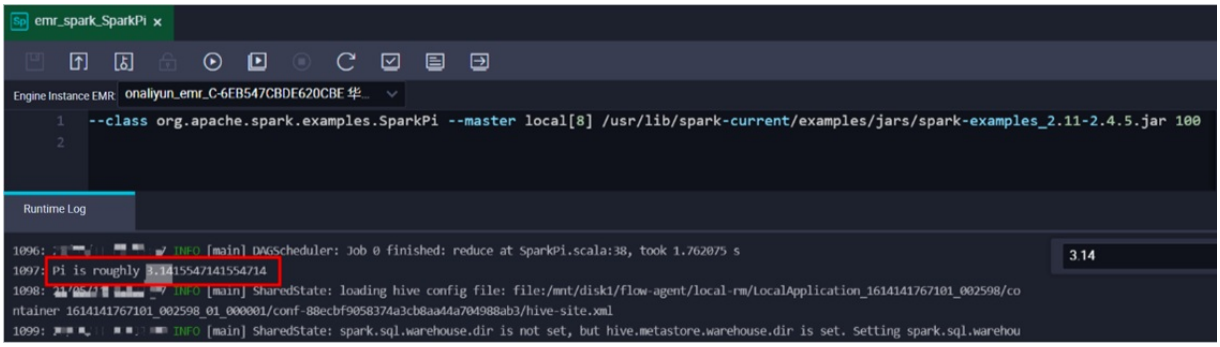
   ```
   --class org.apache.spark.examples.SparkPi --master local[8] /usr/lib/spark-current/examples/jars/spark-examples_2.11-2.4.5.jar 100
   ```

   You need only to complete the content that follows *spark-submit*. *spark-submit* automatically appears when you commit the EMR Spark node. The following code shows the syntax of spark-submit and the code that is actually run:

   ```
   # 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. Save and commit the EMR Spark node. For more information, see *Save and commit the EMR Spark node*.

If the result `1097: Pi is roughly 3.1415547141554714` is returned, the EMR Spark node is successfully run.



Data development example 2: Use Spark to access MaxCompute

This example demonstrates the features of the EMR Spark node by using Spark to obtain the number of rows in a MaxCompute table. For more information, see Use Spark to access MaxCompute.

Before you perform the operation in this example, you must make the following preparations:

- Prepare the environment.
  - Associate an EMR compute engine instance and a MaxCompute compute engine instance with the workspace. For more information, see Configure a workspace.
  - Activate Object Storage Service (OSS) and create a bucket. For more information, see Create buckets.
  - Create an IntelliJ IDEA project in Scala.
- Prepare test data.

  Create an ODPS SQL node on the DataStudio page, and execute SQL statements to create a MaxCompute table and insert data into the table. In this example, the following statements are executed to set the data type of the first column of the MaxCompute table to BIGINT and insert two data records into the table. For more information about how to create an ODPS SQL node, see Create an ODPS SQL node.

```
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. Create a Spark Maven project and add Project Object Model (POM) dependencies. For more information, see Preparations.

   Add POM dependencies. The following code provides an example:

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

   You can refer to the following code in this example. You can change the code based on your business requirements.

   ```
   <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. Use Spark to obtain the number of rows in which data is of the BIGINT type in the first column of the MaxCompute table. For more information, see Use Spark to access MaxCompute.

   The following code provides an example:

```
/*
 * 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)
  }
}
```

Compress the obtained data into a JAR package. In this example, the JAR package *emr_spark_demo-1.0-SNAPSHOT-jar-with-dependencies.jar* is generated.

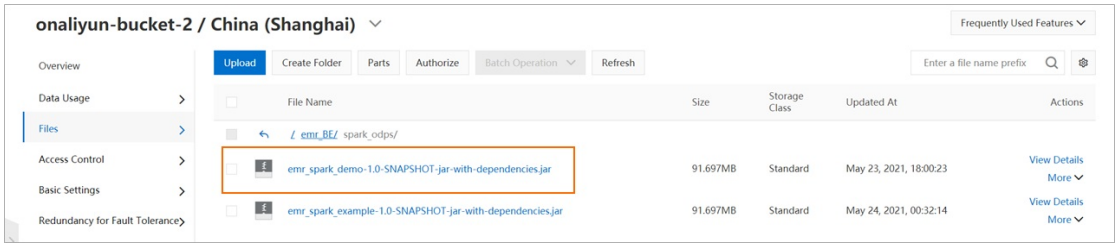
> **Note** Dependencies that are related to MaxCompute are contained in a third-party package. You must compress the third-party package and the obtained data into a JAR package and upload the JAR package to your EMR cluster.

3. Upload the JAR package.
   i. Log on to the OSS console.

ii. Upload the JAR package to the specified path.

In this example, the JAR package *emr_spark_demo-1.0-SNAPSHOT-jar-with-dependencies.jar* is uploaded to the *oss://oss-cn-shanghai-internal.aliyuncs.com/onaliyun-bucket-2/emr_BE/spark_odps/* path. If this is the first time you use OSS to access DataWorks, you must grant access permissions to OSS. For more information, see Create and use an EMR MR node.

> **Note** The upper limit for the size of EMR resources in the DataWorks console is 50 MB. In most cases, the size of a JAR package that is used to add dependencies is greater than 50 MB. Therefore, you must upload the JAR package in the OSS console. If the size of your JAR package is less than 50 MB, you can directly upload the package in the DataWorks console. For more information, see Create and use an EMR JAR resource.



4. Create an EMR Spark node and run the node.

   In this example, the EMR Spark node *emr_spark_odps* is created. For more information about how to create an EMR Spark node, see Create an EMR Spark node.

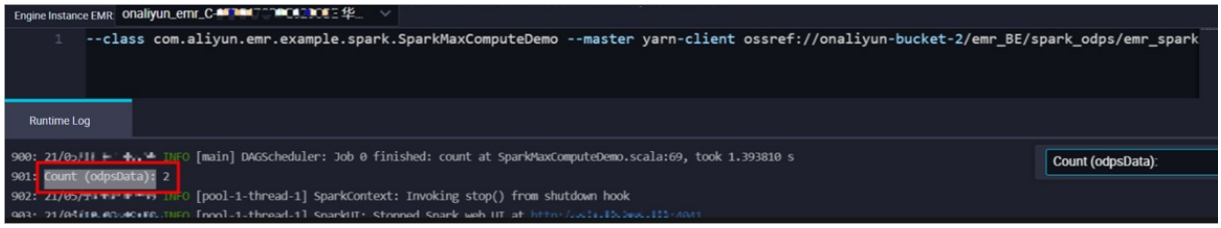
   On the configuration tab of the *emr_spark_odps* node, select the desired EMR compute engine instance and write the following code:

   ```
   --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
   ```

   You must replace the parameters such as <accessKeyId>, <accessKeySecret>, <envType>, <project>, <table>, and <numPartitions> with the actual information that you need to use.

5. Save and commit the EMR Spark node. For more information, see *Save and commit the EMR Spark node*.

You can view the returned result from the operational logs of the EMR Spark node. If the number of data records in the MaxCompute table is *2* in the returned result, the EMR Spark node is successfully run.



5.4.8. Create and use an EMR Shell node

You can create an E-MapReduce (EMR) Shell node and run the node by using the code editor.

Prerequisites

- An Alibaba Cloud EMR cluster is created, and an inbound rule that contains the following content is

added to the security group to which the cluster belongs.

- Action: Allow
- Protocol type: Custom TCP
- Port range: 8898/8898
- Authorization object: 100.104.0.0/16

- An EMR compute engine instance is associated with your workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Configure a workspace.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR Hive nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned when you run EMR Hive nodes.
  i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

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

  > **Note** In the preceding code, `ALISA.*` and `SKYNET.*` are specific to DataWorks.

  ii. After the whitelist configurations are modified, you must restart the Hive service to make the configurations take effect. For more information, see Restart a service.
- An exclusive resource group for scheduling is created, and the resource group is associated with the virtual private cloud (VPC) where the EMR cluster resides. For more information, see Create and use an exclusive resource group for scheduling.

  > **Note** You can use only exclusive resource groups for scheduling to run EMR Hive nodes.

Create an EMR Shell node and use the node to develop data

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2.
3. Create an **EMR Shell** node.
   i. On the DataStudio page, move the pointer over the **Create** icon and choose **EMR > EMR Shell**.

   Alternatively, you can find the desired workflow, right-click the workflow name, and then choose **Create > EMR > EMR Shell**.
   ii.
   iii. Click **Commit**. Then, the configuration tab of the **EMR Shell** node appears.

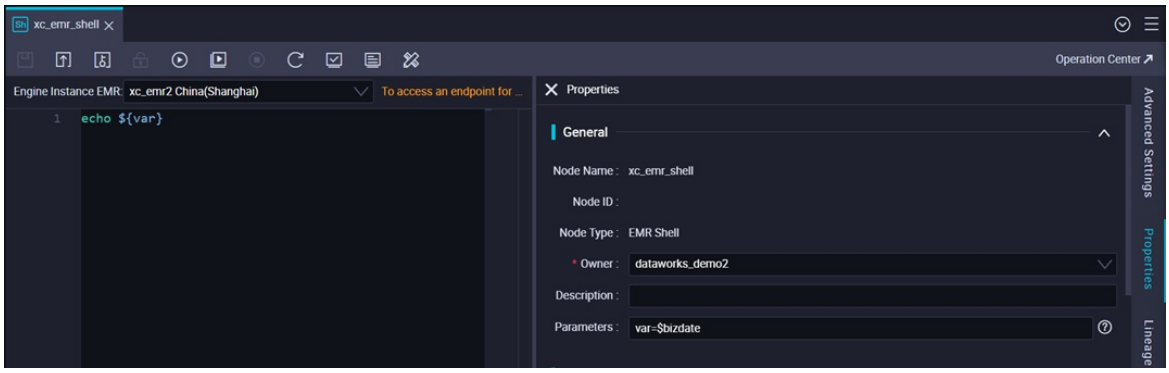
4. Use the **EMR Shell** node to develop data.

   The following code provides an example:

   ```
   DD=`date`;
   echo "hello world, $DD"
   ## Scheduling parameters are supported.
   echo ${var};
   ```

   For more information about scheduling parameters, see Overview of scheduling parameters.

   If you want to change the values that are assigned to the parameters in the code, click **Run with Parameters** in the top toolbar. For more information about value assignment for the scheduling parameters, see Scheduling parameters.

   For more information about how to configure a Shell job, see Configure a Shell job.

5. In the right-side navigation pane, click **Advanced Settings**. In the Advanced Settings panel, configure the parameters.
   - "SPARK_CONF": "--conf spark.driver.memory=2g --conf xxx=xxx": the parameters for running Spark jobs. You can configure multiple parameters in the --conf xxx=xxx format.
   - "queue": the scheduling queue to which jobs are committed. Default value: default.
   - "vcores": the number of vCPUs. Default value: 1.
   - "memory": the memory that is allocated to the launcher. Unit: MB. Default value: 2048.
   - "priority": the priority. Default value: 1.
   - "FLOW_SKIP_SQL_ANALYZE": specifies how SQL statements are executed. The value false indicates that only one SQL statement is executed at a time. The value true indicates that multiple SQL statements are executed at a time.
   - "USE_GATEWAY": specifies whether a gateway cluster is used to commit jobs on the current node. The value **true** indicates that a gateway cluster is used to commit jobs. The value **false** indicates that a gateway cluster is not used to commit jobs and jobs are committed to the header node by default.

     > **Note** If the EMR cluster to which the node belongs is not associated with a gateway cluster but you set the USE_GATEWAY parameter to **true**, jobs may fail to be committed.

6. Configure properties for the EMR Shell node.

   If you want the system to periodically run the EMR Shell node, you can click **Properties** in the right-side navigation pane to configure properties for the node based on your business requirements.

   -

-
- Configure resource properties for the EMR Shell node. For more information, see Configure a resource group. You must select an exclusive resource group for scheduling that is connected to the EMR compute engine instance to run the EMR Shell node. For more information, see Select a network connectivity solution.

7.
8.

5.4.9. Create and use an EMR Spark Streaming node

E-MapReduce (EMR) Spark Streaming nodes can be used to process streaming data with high throughput. This type of node supports fault tolerance, which helps you restore data streams on which errors occur. This topic describes how to create an EMR Spark Streaming node and use the node to develop data.

Prerequisites

- An Alibaba Cloud EMR cluster is created, and an inbound rule that contains the following content is added to the security group to which the cluster belongs.
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- The EMR cluster is associated with your DataWorks workspace as a compute engine instance. The EMR folder is displayed on the DataStudio page only after an EMR cluster is associated with your workspace as a compute engine instance on the Workspace Management page. For more information, see Configure a workspace.
- A resource group is created.

  An exclusive resource group for scheduling is created. For more information, see Create and use an exclusive resource group for scheduling.

Limits

- An EMR Spark Streaming node can run only on an exclusive resource group for scheduling.
- If the exclusive resource group for scheduling and EMR cluster that you use are created before June 10, 2021, you must submit a ticket to upgrade the resource group and the EMR cluster.

Create an EMR Spark Streaming node and use the node to develop data

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.

2.

3. Create an **EMR Spark Streaming** node.

   i. On the DataStudio page, move the pointer over the **Create** icon and choose **EMR > EMR Spark Streaming**.

   Alternatively, you can find the desired workflow, right-click the workflow name, and then choose **Create > EMR > EMR Spark Streaming**.

   ii.

   iii. Click **Commit**. Then, the configuration tab of the **EMR Spark Streaming** node appears.

4. Use the **EMR Spark Streaming** node to develop data.

   i. Select the EMR compute engine instance.

   On the configuration tab of the **EMR Spark Streaming node**, select the EMR compute engine instance.

   ii. Write code for the EMR Spark Streaming node.

   On the configuration tab of the EMR Spark Streaming node, write code for the node.

   Sample code:

```
--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>
```

   The system automatically generates `spark-submit` after the node is created. The following code is the code that is finally sent to the compute engine instance:

```
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` is the name of the JAR package generated by the node code. For more information about the code for the node, see Consume data in real time.

     > **Note** The JAR package can be stored in the following objects:
     > - Master node of the EMR cluster.
     > - Object Storage Service (OSS). We recommend that you store the JAR package in OSS. For more information about how to store the JAR package in OSS, see Operations in the OSS console.

   - You must replace `access-key-id` and `access-key-secret` with the AccessKey ID and AccessKey secret of your Alibaba Cloud account. To obtain the AccessKey ID and AccessKey secret, you can log on to the DataWorks console, move the pointer over the profile picture in the upper-right corner, and then select AccessKey Management.

   For more information about Spark Streaming parameters, see Spark documentation.

iii. Configure a resource group for scheduling.

- Click the icon in the top toolbar. In the Parameters dialog box, select the desired resource group for scheduling.
- Click **OK**.

iv. Save and run the EMR Streaming SQL node.

In the top toolbar, click the icon to save the EMR Streaming SQL node and click the icon to run the EMR Streaming SQL node.

5. Configure properties for the EMR Spark Streaming node.

If you want the system to periodically run the EMR Spark Streaming node, you can click **Properties** in the right-side navigation pane to configure properties for the node based on your business requirements.

- Configure basic properties for the EMR Spark Streaming node. For more information, see Configure basic properties.
- Configure time properties for the EMR Spark Streaming node.

  You can select a mode to start the node and a mode to rerun the node. For more information, see Configure time properties.
- Configure resource properties for the EMR Spark Streaming node. For more information, see Configure a resource group.

6.

7. View the EMR Spark Streaming node.

i. Click **Operation Center** in the upper-right corner of the DataStudio page to go to Operation Center.

ii. View the EMR Spark Streaming node that is running. For more information, see Manage real-time computing nodes.

5.5. Create a Hologres SQL node

This topic describes how to create a Hologres SQL node. Hologres seamlessly integrates with MaxCompute at the underlying layer. This integration allows you to use standard PostgreSQL statements to query and analyze large volumes of data stored in MaxCompute. In this case, you can quickly obtain query results without the need to transfer data.

Prerequisites

A **Hologres** compute engine instance is added on the **Workspace Management** page. This ensures that the **Hologres** folder is displayed on the page on which you want to create a Hologres SQL node. For more information, see Associate a Hologres compute engine instance with a workspace.

Procedure

1. Go to the **DataStudio** page.

i. Log on to the DataWorks console.

ii. In the left-side navigation pane, click **Workspaces**.

iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.

2. Create a **workflow**.

   If you have a **workflow**, skip this step.

   i. Move the pointer over the **+Create** icon and select **Workflow**.

   ii. In the **Create Workflow** dialog box, specify the **Workflow Name** parameter.

   iii. Click **Create**.

3. Create a **Hologres SQL** node.

   i. Move the pointer over the **+Create** icon and choose **Hologres > Hologres SQL**.

   You can also find the workflow in which you want to create a Hologres SQL node, right-click **Hologres**, and choose **Create > Hologres SQL**.

   ii. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length, and can contain letters, digits, underscores (_), and periods (.).

   iii. Click **Commit**.

4. On the tab that appears, edit and run the code in the code editor.

   After the Hologres SQL node is created, edit the code in compliance with the Hologres SQL syntax. For more information about the Hologres SQL syntax, see SELECT.

   > **Note** If you query data by executing a `SELECT` statement that does not contain the `LIMIT` clause, a maximum of 200 records are returned by default. If you want to view more query results, add the `LIMIT` clause at the end of the `SELECT` statement. A maximum of 10,000 records can be returned.
   >
   > For example, if you execute the `select col_1,col_2 from your_table_name where pt>0 limit 500;` statement, a maximum of 500 records are returned.

5. On the node configuration tab, click the **Properties** tab in the right-side navigation pane and configure scheduling properties for the node. For more information, see Configure basic properties.

6. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.

7. Test the node. For more information, see View auto triggered nodes.

5.6. Create an ADB for PostgreSQL node

DataWorks supports ADB for PostgreSQL nodes. You can create an ADB for PostgreSQL node in the DataWorks console to build an online extract, transform, load (ETL) process.

Prerequisites

- DataWorks Standard Edition or higher is activated.
- An AnalyticDB for PostgreSQL instance is bound to the workspace where you want to create an ADB for PostgreSQL node. The AnalyticDB for PostgreSQL service is available in a workspace only after you bind an AnalyticDB for PostgreSQL instance to the workspace on the **Workspace Management** page. For more information, see Configure a workspace.
- An exclusive resource group for scheduling is added. For more information, see Purchase an exclusive resource group for scheduling.

> **Note** ADB for PostgreSQL nodes can run only on exclusive resource groups for scheduling.

Context

ADB for PostgreSQL nodes are used to connect to AnalyticDB for PostgreSQL of Alibaba Cloud. For more information, see AnalyticDB for PostgreSQL.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the **DataStudio** page, move the pointer over the **Create** icon and choose **AnalyticDB > ADB for PostgreSQL**.

   Alternatively, you can click a workflow in the Business process section, right-click **AnalyticDB for PostgreSQL**, and then choose **New > ADB for PostgreSQL**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. Configure the ADB for PostgreSQL node.

i. Select a connection from the **Select data source** drop-down list.

> **Notice**
> - When you bind an AnalyticDB for PostgreSQL instance to the workspace, DataWorks automatically creates a connection to the instance.
> - You can select only a connection that is configured by using a connection string.

ii. Write the SQL statements of the node.

After you select a connection, write SQL statements based on the syntax that is supported by AnalyticDB for PostgreSQL.

iii. Click the icon in the toolbar to save the SQL statements to the server.

iv. Click the icon in the toolbar to execute the SQL statements you have saved.

When you run the node for the first time, the **Parameters** dialog box appears. You must select a resource group for running the node from the **Scheduling Resource Group** drop-down list, set other parameters as required, and then click **Confirm**.

When you run the node later, the system uses the resource group and parameter settings that you specify for the first running of the node. If you need to change the resource group or modify the parameter settings, click the icon in the toolbar.

> **Note** To access a data store in a virtual private cloud (VPC), a node must be run on an exclusive resource group for scheduling. In this example, you must select an exclusive resource group for scheduling that is connected to the AnalyticDB for PostgreSQL instance.

6. On the node configuration tab, click the **Scheduling configuration** tab in the right-side navigation pane. On the Scheduling configuration tab, set the scheduling properties for the node. For more information, see Configure basic properties.

   You must select an exclusive resource group for scheduling that is connected to the AnalyticDB for PostgreSQL instance to periodically run the node.

7. Save and commit the node.

> **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

i. Click the icon in the toolbar to save the node.

ii. Click the icon in the toolbar.

iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iv. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.

8. Test the node. For more information, see View auto triggered nodes.

5.7. Create and use an AnalyticDB for MySQL node

You can create an AnalyticDB for MySQL node and use SQL statements to develop data for the AnalyticDB for MySQL data source. This topic describes how to create and use an AnalyticDB for MySQL node.

Prerequisites

- DataWorks of the required edition is activated, the required data source is purchased, and an AnalyticDB for MySQL compute engine instance is associated with the workspace.
  - DataWorks Standard Edition or a more advanced edition is activated. For more information, see Billing of DataWorks advanced editions.
  - An AnalyticDB for MySQL cluster is created. For more information, see Create a cluster.
  - An AnalyticDB for MySQL compute engine instance is associated with the workspace on the **Workspace Management** page. For more information, see Configure a workspace.
- A resource group is created.

  An exclusive resource group for scheduling is created. For more information, see Purchase an exclusive resource group for scheduling.
- A data source is prepared.

  An AnalyticDB for MySQL data source is added. For more information, see Configure an AnalyticDB for MySQL 3.0 connection.

Context

AnalyticDB for MySQL is an analytical database of Alibaba Cloud. For more information, see What is AnalyticDB for MySQL?

Limits

- An AnalyticDB for MySQL compute engine instance can be associated only with a DataWorks workspace of the Standard Edition or a more advanced edition.
- AnalyticDB for MySQL nodes can run only on exclusive resource groups for scheduling.
- An AnalyticDB for MySQL node can be used to develop data for an AnalyticDB for MySQL data source that is added by using the connection string mode or by associating an AnalyticDB for MySQL compute engine instance with a workspace. You can go to the **Data Source** page, find the desired data source, and then click **Edit** in the **Operation** column to view the mode that was used to add the data source. For more information, see Configure an AnalyticDB for MySQL 3.0 connection.

Create an AnalyticDB for MySQL node and use the node to develop data

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.

2.

3. Create an **AnalyticDB for MySQL** node.

   i. On the DataStudio page, move the pointer over the  icon and choose **AnalyticDB for MySQL > ADB for MySQL**.

   You can also find the newly created workflow, right-click the workflow name, and then choose **Create > AnalyticDB for MySQL > ADB for MySQL**.

   ii.

   iii. Click **Commit**. Then, the configuration tab of the AnalyticDB for MySQL node appears.

4. Use the **AnalyticDB for MySQL** node to develop data.

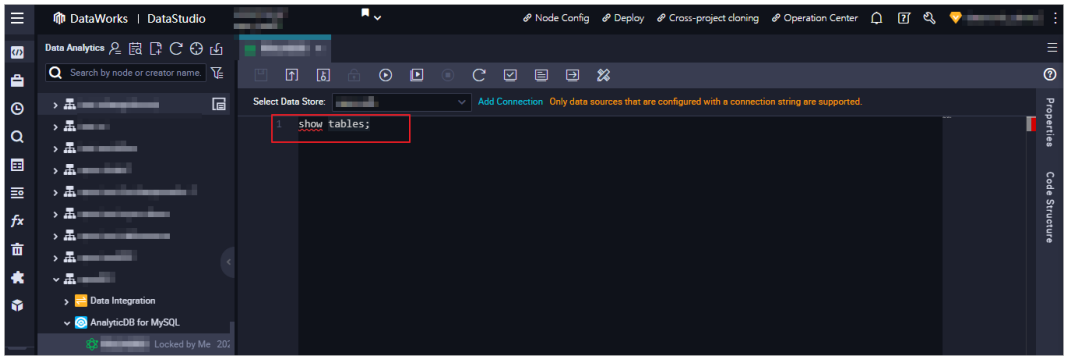
   i. Select a data source.

   Select the desired AnalyticDB for MySQL data source from the **Select Data Source** drop-down list. If no AnalyticDB for MySQL data source is available, click **Add Data Source** and add an AnalyticDB for MySQL data source on the **Data Source** page. For more information, see Configure an AnalyticDB for MySQL 3.0 connection.

   > **Note**
   >
   > - If you associate an AnalyticDB for MySQL compute engine instance with a workspace, the system automatically creates an AnalyticDB for MySQL data source. The system uses the data source to develop data by default. You can also select the desired data source.
   > - An AnalyticDB for MySQL node can be used to develop data for an AnalyticDB for MySQL data source that is added by using the connection string mode or by associating an AnalyticDB for MySQL compute engine instance with a workspace. You can go to the **Data Source** page, find the desired data source, and then click **Edit** in the **Operation** column to view the mode that was used to add the data source. For more information, see Configure an AnalyticDB for MySQL 3.0 connection.

ii. Use SQL statements to create a task.

a. Enter SQL statements in the SQL editor to create a task.



In this example, the following statement is used to query tables in the data source. You can enter SQL statements that you want to execute based on the syntax supported by AnalyticDB for MySQL and your business requirements.

```
show tables;
```

b. Select a resource group.

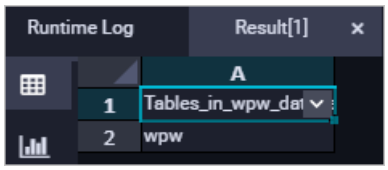
In the top toolbar, click the icon. In the **Parameters** dialog box, select the created exclusive resource group for scheduling.

> **Note**
>
> - You must use an exclusive resource group for scheduling that is connected to the data source. For more information, see Select a network connectivity solution.
> - If you want to change the resource group in subsequent operations, you can also perform the change in the Parameters dialog box.

c. Save and execute the SQL statement.

In the top toolbar, click the icon to save the compiled SQL statement. Then, click the icon to execute the SQL statement.

The following figure shows the results.



5. Configure properties for the AnalyticDB for MySQL node.

If you want the system to periodically run the AnalyticDB for MySQL node, you can click **Properties** in the right-side navigation pane to configure properties for the node based on your business requirements.

-
-
- Configure resource properties for the AnalyticDB for MySQL node. For more information, see

Configure a resource group. You must select the exclusive resource group for scheduling that is connected to the AnalyticDB for MySQL data source. For more information, see Select a network connectivity solution.

6.

7.

5.8. Create a MySQL node

You can create a MySQL node and use SQL statements to develop data for a MySQL data source. This topic describes how to create and use a MySQL node.

Prerequisites

- MySQL nodes can use only exclusive resource groups for scheduling. For more information about how to create and use an exclusive resource group for scheduling, see Create and use an exclusive resource group for scheduling.
- A MySQL data source is added by using the connection string mode. For more information, see Add a MySQL data source. Make sure that the data source that you created is connected to the exclusive resource group for scheduling.

> **Note** MySQL nodes support only data sources in the production environment.

Limits

- MySQL nodes can be used to develop data only for MySQL data sources that are in the production environment and that are added by using the connection string mode. You can go to the **Data Source** page by performing the operations in Add a MySQL data source. On the Data Source page, find your desired data source, and click **Edit** in the **Operation** column to view the mode that is used to add the data source.
- If you want to access the MySQL data source over the Internet, you must configure a whitelist for the data source. To ensure the connectivity between the resource group and the data source, we recommend that you use an exclusive resource group for scheduling.
- If you want to access the MySQL data source over a virtual private cloud (VPC), you can use only an exclusive resource group for scheduling to develop data.

> **Note** The current node type does not support MySQL 8.0 or later.

Proposals and suggestions for network connections

This section describes the proposals and suggestions for network connections.

- Resources in exclusive resource groups for scheduling can be scheduled on demand to ensure the outputs of the nodes. We recommend that you use exclusive resource groups for scheduling to run nodes.
- If you want to access the MySQL data source over a VPC, use an exclusive resource group for scheduling.
- If you want to access the MySQL data source over the Internet, we recommend that you use an exclusive resource group for scheduling.
- If you want to use the shared resource group for scheduling to access the MySQL data source over

the Internet, you must configure a whitelist for the data source.

Create and use a MySQL node to develop data

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Create a **workflow**.

   If you have a **workflow**, skip this step.
   i. Move the pointer over the **Create** icon and select **Workflow**.
   ii. In the **Create Workflow** dialog box, set the **Workflow Name** parameter.
   iii. Click **Create**.
3. Create a **MySQL** node.
   i. Move the pointer over the **Create** icon and choose **Database > MySQL**.

   You can also right-click the name of the workflow that you created and then choose **Create > Database > MySQL**.
   ii. In the **Create Node** dialog box, set the **Node Name**, **Node Type**, and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

   iii. Click **Commit** to go to the configuration tab of the MySQL node.
4. Use the **MySQL** node to develop data.
   i. Select a data source.

   Select the data source for which you want to develop data from the **Select Data Source** drop-down list. If you cannot find the required data source in the drop-down list, click **Add Data Source** and add a data source on the **Data Source** page. For more information, see Add a MySQL data source.

   > **Note**
   >
   > MySQL nodes can be used to develop data only for MySQL data sources that are in the production environment and that are added by using the connection string mode. You can go to the **Data Source** page by performing the operations in Add a MySQL data source. On the Data Source page, find your desired data source, and click **Edit** in the **Operation** column to view the mode that is used to add the data source.

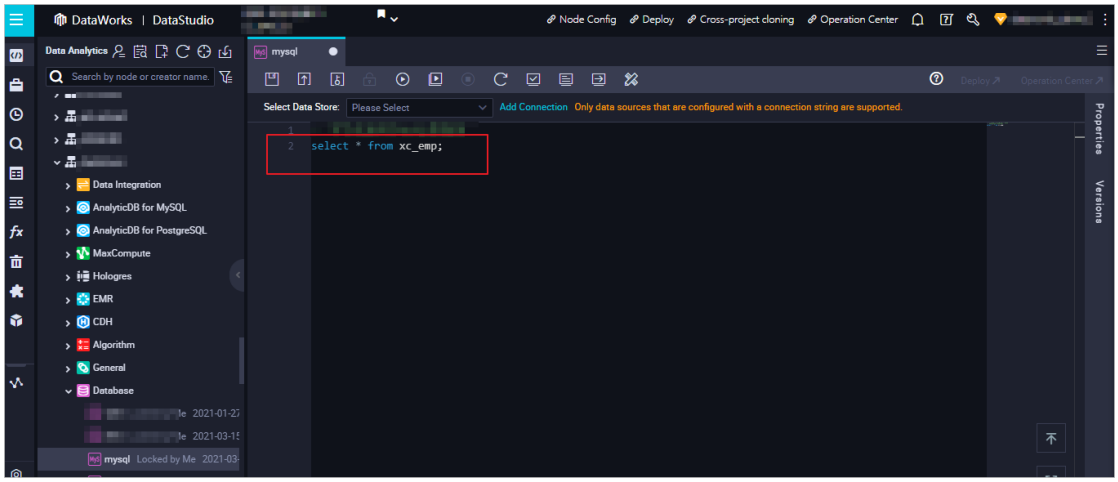
ii. Select a resource group.

In the top toolbar, click the icon. In the **Parameters** dialog box, select the created resource group for scheduling.

> **Note**
>
> - If you want to access a data source over the Internet or a VPC, you must use the resource group for scheduling that is connected to the data source. For more information, see Select a network connectivity solution.
> - If you want to change the resource group in subsequent operations, you can also perform the change in the Parameters dialog box.

iii. Use SQL statements to create a task.

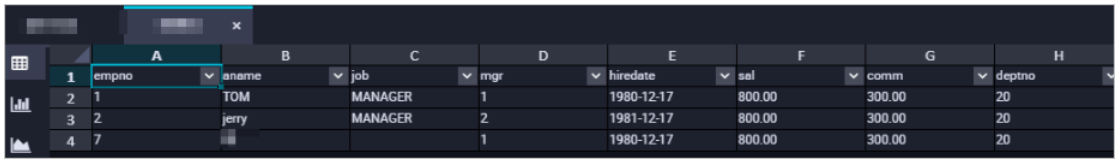
Write SQL statements in the SQL editor to create a task.



In this example, the following statement is used to query data in the *xc_emp* table. You can write SQL statements that you want to execute based on your business requirements and the syntax that is supported by MySQL.

```
select * from xc_emp;
```

The following figure shows the results.



	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

If the task execution fails, you can view the error message and troubleshoot the issues based on the instructions in What do I do if the node fails to run and the system displays the error message indicating that SQL execution failed and the JDBC driver is not supported?.

iv. Save and execute the SQL statements.

In the top toolbar, click the icon to save the SQL statements. Then, click the icon to execute the SQL statements.

5. Configure scheduling properties for the MySQL node.

If you want the system to periodically run the MySQL node, you can click **Properties** in the right-side navigation pane to configure scheduling properties for the node based on your business requirements.

- Configure basic properties for the MySQL node. For more information, see Configure basic properties.
- Configure the scheduling cycle, rerun properties, and scheduling dependencies of the MySQL node. For more information, see Configure time properties and Configure same-cycle scheduling dependencies.

  > **Note** You must set the **Rerun** and **Parent Nodes** parameters on the Properties tab before you commit the node.

- Configure resource properties for the MySQL node. For more information, see Configure a resource group. If you want to access the MySQL data source over the Internet or a VPC, you must use the exclusive resource group for scheduling that is connected to the MySQL data source to run the MySQL node. For more information, see Select a network connectivity solution.

6. Commit and deploy the MySQL node.
   i. Click the icon in the top toolbar to save the node.
   ii. Click the icon in the top toolbar to commit the node.
   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iv. Click **OK**.

   If you use a workspace in standard mode, you must deploy the node in the production environment after you commit the node. Click **Deploy** in the upper-right corner. For more information, see Deploy nodes.
7. View the MySQL node.
   i. Click **Operation Center** in the upper-right corner of the DataStudio page to go to Operation Center.
   ii. View the scheduled MySQL node. For more information, see View auto triggered nodes.

What do I do if the node fails to run and the system displays the error message indicating that SQL execution failed and the JDBC driver is not supported?

- Problem description

  When I configure a MySQL data source that is not added by using the connection string mode, the node fails to run, and the system displays the error message indicating that SQL execution failed and `the JDBC driver is not supported`.

- Cause

  A MySQL data source that is not added by using the connection string mode is used.

- Solution

Select a data source that is added by using the connection string mode. You can go to the **Data Source** page by performing the operations in Add a MySQL data source. On the Data Source page, find your desired data source, and click **Edit** in the **Operation** column to view the mode that is used to add the data source.

5.9. Create a Machine Learning (PAI) node

Machine Learning (PAI) nodes are used to call nodes that are created in Machine Learning Platform for AI (PAI) and schedule production activities based on the node configuration.

Prerequisites

Before you create a Machine Learning (PAI) node in DataWorks, you must create a PAI experiment in PAI.

In this topic, a heart disease prediction experiment is used to describe how to load a PAI experiment to a Machine Learning node in DataWorks.

> **Note** Before you perform the operations described in this topic, you must go to the PAI console from the DataStudio page, create a PAI experiment based on the heart disease prediction case described in Predict heart disease, and then upload the required data to DataStudio.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the **DataStudio** page, move the pointer over the **+Create** icon and choose **Machine Learning > Machine Learning Platform for AI** to create a Machine Learning (PAI) node.

   Alternatively, you can open the desired workflow, right-click **Algorithm**, and then choose **Create > Machine Learning Platform for AI** to create a Machine Learning (PAI) node.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. On the configuration tab of the node, select a PAI experiment that you have created from the **Experiment** drop-down list.

   If you want to edit the loaded PAI experiment, click **Edit in PAI Console** to go to the experiment editing page in the PAI console.

> **Note** You can select a PAI experiment from the Experiment drop-down list and edit the experiment on the experiment editing page in the PAI console only after you have created a PAI experiment in the PAI console.

6. On the configuration tab of the node, click **Properties** in the right-side navigation pane. On the Properties tab, configure properties for the node. For more information, see Configure basic properties.
7. Save and commit the node.

> **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.
   ii. Click the icon in the toolbar.
   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.
8. Test the node. For more information, see View auto triggered nodes.

5.10. Create and use a ClickHouse SQL node

ClickHouse SQL nodes allow you to use a distributed SQL query engine to process structured data. This improves the task efficiency. This topic describes how to create and use a ClickHouse SQL node to develop data.

Prerequisites

- An EMR ClickHouse cluster or an ApsaraDB for ClickHouse cluster is created. The security group to which the cluster belongs has the following inbound rule:
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- The ClickHouse compute engine is associated with your DataWorks workspace. For more information, see Configure a workspace.

> **Note** After you associate the ClickHouse compute engine with your workspace, you can view the directory of the ClickHouse compute engine on the **DataStudio** page.

- The exclusive resource group for scheduling is created and is associated with the VPC to which the ClickHouse cluster belongs. For more information, see Create and use an exclusive resource group for scheduling.

Limits

You can use only exclusive resource groups for scheduling to run ClickHouse SQL nodes.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Move the pointer over the **+Create** icon and choose **ClickHouse > Click SQL**.

   Alternatively, right-click a workflow and select **ClickHouse**, and choose **Create > Click SQL**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. Develop data in the code editor.

   You can run SQL nodes based on your business requirements. Sample code:

```
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. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.
   ii. Click the icon in the toolbar.
   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.
7. Test the node. For more information, see View auto triggered nodes.

5.11. Create a common node

5.11.1. OSS Object Inspection node

You can use the Object Storage Service (OSS) object inspection feature to monitor OSS objects if descendant nodes can run only after OSS objects are generated in OSS. For example, you can run a node for synchronizing OSS data files to DataWorks only after the OSS data files are generated. In this case, you can use the OSS object inspection feature to monitor the OSS data files.

The OSS object inspection feature can monitor OSS objects of all tenants. To create an OSS Object Inspection node, follow these steps:

1. Log on to the DataWorks console. In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
2. Move the pointer over the +Create icon and choose **General > OSS Object Inspection**.

   You can also find the target workflow, right-click **General**, and choose **Create > OSS Object Inspection**.
3. In the **Create Node** dialog box that appears, enter the node name, select the target folder, and click **Commit**.

   > **Note** A node name can be up to 128 characters in length.
4. On the **OSS Object Inspection** page that appears, set the parameters.

No.	Parameter	Description
1	**OSS Object**	The storage path of the OSS object. You can add a scheduling parameter to the storage path. For more information, see Overview of scheduling parameters.
2	**Timeout**	The timeout period during which DataWorks checks whether the OSS object exists in OSS every five seconds. If the OSS object is not detected before the timeout period ends, the OSS Object Inspection node fails.
3	**Storage Address**	The storage space of the OSS object. Valid values: ○ **Myself**: detects the OSS object in the storage space of the current tenant. ○ **Other**: detects the OSS object in the storage space of another tenant.

> **Note**
>
> - When an OSS Object Inspection node is running, it monitors the OSS object through MaxCompute. Make sure that MaxCompute has the required permissions on the OSS bucket. For more information, see STS authorization.
> - In the development or production environment, the node monitors the OSS object through the access identity of the development or production environment. Make sure that the access identity has the required permissions on the OSS bucket.
> - You cannot specify an OSS object by using the wildcard (*), nor can you use the system parameters cyctime and bizdate. However, you can use custom parameters.

5. Grant MaxCompute the permission to access OSS in the Resource Access Management (RAM) console.

   MaxCompute uses RAM and Security Token Service (STS) of Alibaba Cloud to resolve security issues of accounts.

   - If the owners of MaxCompute and OSS are using the same Alibaba Cloud account, you can go to the RAM console and click Confirm Authorization Policy to authorize MaxCompute to access OSS.
   - If the owners of MaxCompute and OSS are using different Alibaba Cloud accounts, you can follow these steps to authorize MaxCompute to access OSS:
     a. Create a role in the RAM console.

        Create a role, such as AliyunODPSDefaultRole or AliyunODPSRoleForOtherUser, and set the following policy:

```
--The owners of MaxCompute and OSS are using different Alibaba Cloud accounts.
{
"Statement": [
{
"Action": "sts:AssumeRole",
"Effect": "Allow",
"Principal": {
"Service": [
"The ID of the Alibaba Cloud account used by the owner of MaxCompute@odps.aliyuncs.com"
]
}
}
],
"Version": "1"
}
```

b. Create the AliyunODPSRolePolicy permission policy that contains the permissions required for accessing OSS.

```
{
"Version": "1",
"Statement": [
{
"Action": [
 "oss:ListBuckets",
 "oss:GetObject",
 "oss:ListObjects",
 "oss:PutObject",
 "oss:DeleteObject",
 "oss:AbortMultipartUpload",
 "oss:ListParts"
],
"Resource": "*",
"Effect": "Allow"
}
]
}
--You can also add other permissions as required.
```

c. Grant the AliyunODPSRolePolicy permission policy to the role.

6. Go to **Operation Center** to view the runtime logs.

   If the following error information appears, the OSS object is not detected:

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

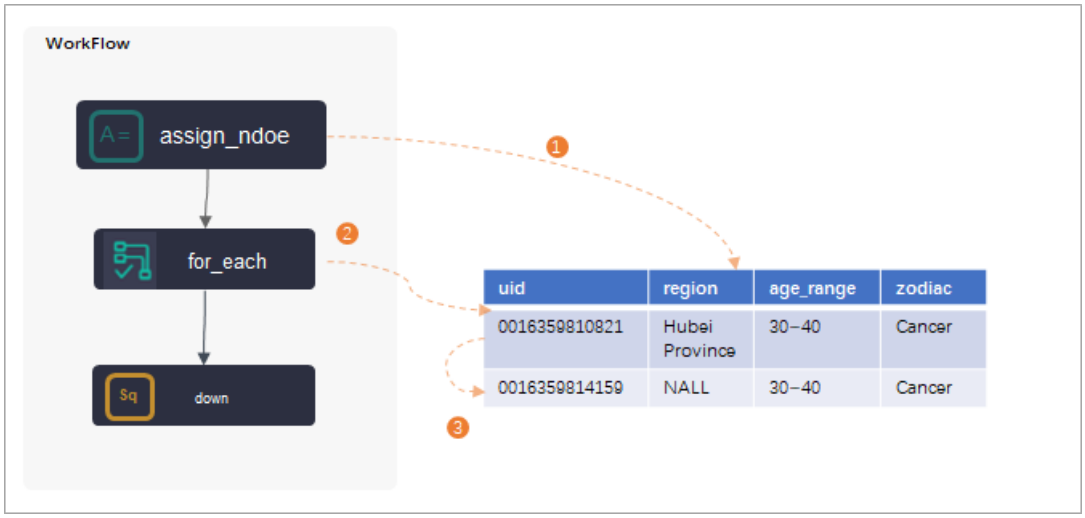
5.11.2. Configure a for-each node

5.11.2.1. Composition and application logic

DataWorks provides for-each nodes. You can use a for-each node to traverse the output of an assignment node in loops. You can also customize the workflow in a for-each node. This topic describes the composition and application logic of a for-each node.

Scenarios

In DataWorks, a for-each node is used to traverse the output of an assignment node in loops. Before you use a for-each node, you must configure the for-each node as a descendant node of an assignment node. After the assignment node passes its output to the for-each node, the for-each node traverses the output in loops.



For more information about the limits of a for-each node, see Limits.

A for-each node contains inner nodes. The inner nodes are used to compile task code for traversal in loops. For more information, see Node composition.

Limits

- Dependencies

  A for-each node is used to traverse the output of an assignment node in loops. You must configure the assignment node as an ancestor node of the for-each node.

- for-each nodes
  - You can use for-each nodes only in DataWorks of the Standard Edition or a more advanced edition.

- A for-each node can traverse the output of an assignment node in a maximum of 128 loops. If the number of loops exceeds 128, an error is reported. The number of loops that are actually run by a for-each node varies based on the output of the assignment node that is configured as an ancestor node of the for-each node.
  - If the output of an assignment node is a one-dimensional array, the number of loops is the number of elements in the array.

    For example, a for-each node is configured as a descendant node of an assignment node in the Shell or Python 2 language, and the output of the assignment node is the one-dimensional array `2021-03-28,2021-03-29,2021-03-30,2021-03-31,2021-04-01`. In this case, the for-each node traverses the output in five loops.
  - If the output of an assignment node is a two-dimensional array, the number of loops is the number of rows in the array.

    For example, a for-each node is configured as a descendant node of an assignment node in the ODPS SQL language, and the output of the assignment node is a two-dimensional array. The following code shows the two-dimensional array.

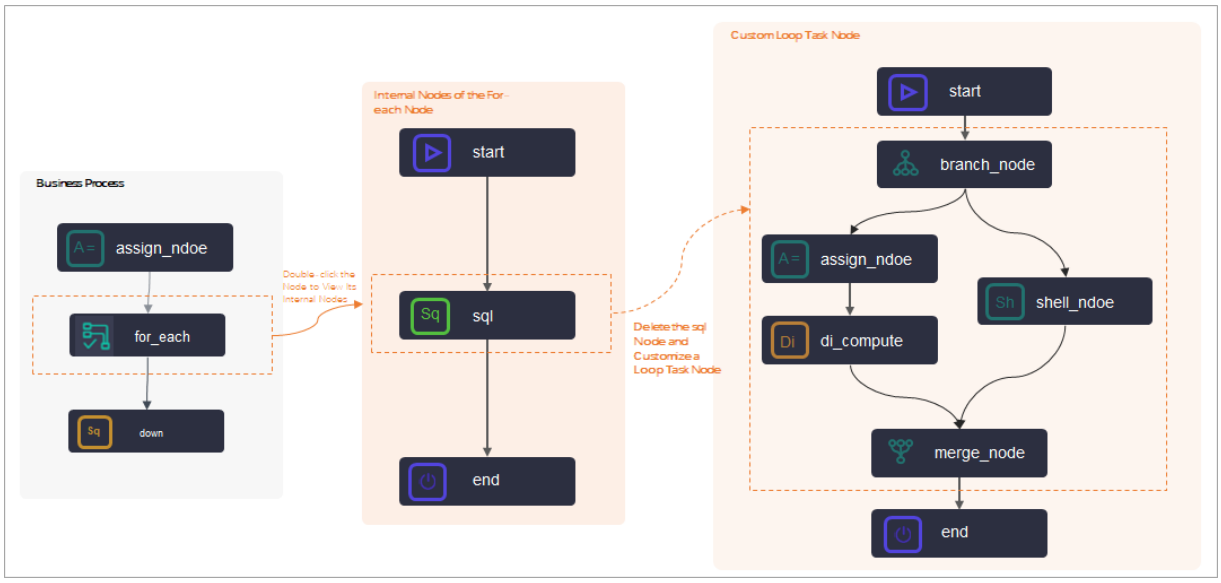
    ```
    +------------------------------------------------------------------------------------+
    | uid            | region         | age_range            | zodiac                 |
    +------------------------------------------------------------------------------------+
    | 0016359810821  | Hubei Province | 30 to 40 years old   | Cancer (constellation) |
    | 0016359814159  | Unknown        | 30 to 40 years old   | Cancer (constellation) |
    +------------------------------------------------------------------------------------+
    ```

    In this case, the for-each node traverses the output in two loops.
- Inner nodes
  - You can delete the existing dependencies between the inner nodes of a for-each node and customize the workflow in the for-each node. However, you must make sure that the workflow of the for-each node starts with the **start** node and ends with the **end** node.
  - If you use a branch node as an inner node of a for-each node to perform logical judgments or traverse the output of an assignment node, you must also use a merge node.
- Testing
  - If the workspace that you use is in standard mode, you cannot test a for-each node on the DataStudio page.

    To test a for-each node and view the result, you must commit and deploy the for-each node to the production environment. Then, you can view the operational logs of the for-each node on the Operation Center page.
  - To view the operational logs of a for-each node on the Operation Center page, right-click the for-each node in a directed acyclic graph (DAG) and select **View Internal Nodes**.

Node composition

In DataWorks, for-each nodes are a special type of node that contains inner nodes. After you create a for-each node, the following three inner nodes are created: **start**, **sql**, and **end**. The start node marks the start of a loop. The sql node runs a loop. The end node marks the end of a loop and determines whether to start the next loop. The three inner nodes are organized as a workflow to traverse the output of an assignment node.



The preceding figure shows the inner nodes of a for-each node.

- **sql** node

  By default, DataWorks creates an SQL node named **sql** in a for-each node. You can delete the sql node and customize inner nodes to run the loop task of the for-each node.

  - If you want to use the **sql** node to run the loop task of your for-each node, double-click the node and write code on the configuration tab of the node.
  - If the loop task of your for-each node is complex, you can create more inner nodes to run the task and connect the nodes based on your business requirements.

    > **Note** Before you customize inner nodes that are used to run the loop task of your for-each node, you can delete the dependencies between the original inner nodes and customize new inner nodes and a workflow in the for-each node. However, you must make sure that the workflow starts with the **start** node and ends with the **end** node.

- **start** and **end** nodes

  The start node marks the start of a loop. The end node marks the end of a loop. The two nodes are not used to run loops.

  > **Note** The end node does not determine the number of loops. The number of loops that are actually run by a for-each node varies based on the output of the assignment node that is configured as an ancestor node of the for-each node.

Built-in variables

Each time you run a for-each node to traverse the output of an assignment node in loops, you can configure some built-in variables to obtain the number of loops that are run by the for-each node and the offset.

Built-in variable	Description	Comparison between a loop run by a for-each node and a for loop
`${dag.loopDataArray}`	The output of an assignment node.	The value of this variable is equivalent to the code result in the for loop. Example: `data=[]`

`${dag.foreach.current}`	The current data entry.	Example of code in the for loop:
		`for(int i=0;i<data.length;i++) {`
		`print(data[i]);`
		`}`
		• `data[i]` is equivalent to `${dag.foreach.current}`.
		• `i` is equivalent to `${dag.offset}`.

`${dag.offset}`	The offset of the current number of loops to 1.	
`${dag.loopTimes}`	The current number of loops.	-

If you understand the schema of your output table, you can configure the variables that are described in the following table to obtain the values of other variables.

Variable	Description
`${dag.dag.foreach.current[n]}`	The data in a column of the current row in each loop if the output of an assignment node is a two-dimensional array.
`${dag.loopDataArray[i][j]}`	The data in Column j of Row i in the output if the output of an assignment node is a two-dimensional array.
`${dag.foreach.current[n]}`	The data in a specified column if the output of an assignment node is a one-dimensional array.

Examples of variable values

- Example 1

  An assignment node in the Shell language is configured as an ancestor node of a for-each node, and the last output of the assignment node is `2021-03-28,2021-03-29,2021-03-30,2021-03-31,2021-04-01`. The following table lists the values of the built-in variables in this example.

  > **Note** The output of the assignment node is a one-dimensional array. The array contains five elements. The elements are separated by commas (,). Therefore, the total number of loops that are run by the for-each node is 5.

Built-in variable	Value obtained in the first loop	Value obtained in the second loop
`${dag.loopDataArray}`	`2021-03-28,2021-03-29,2021-03-30,2021-03-31,2021-04-01`	
`${dag.foreach.current}`	`2021-03-28`	`2021-03-29`
`${dag.offset}`	0	1
`${dag.loopTimes}`	1	2
`${dag.foreach.current[3]}`	`2021-03-30`	

- Example 2

  An assignment node in the ODPS SQL language is configured as an ancestor node of a for-each node. The following pieces of data are obtained after the last SELECT statement is executed:

```
+------------------------------------------------------------------------------+
| uid            | region         | age_range            | zodiac             |
+------------------------------------------------------------------------------+
| 0016359810821  | Hubei Province | 30 to 40 years old   | Cancer (constellation) |
| 0016359814159  | Unknown        | 30 to 40 years old   | Cancer (constellation) |
+------------------------------------------------------------------------------+
```

  The following table lists the values of the built-in variables in this example.

  > **Note** The output of the assignment node is a two-dimensional array. The array contains two rows. Therefore, the total number of loops that are run by the for-each node is 2.

Built-in variable	Value obtained in the first loop	Value obtained in the second loop

`${dag.loopDataArray}`	`+------------------------------------------------------------------------------+`
	`| uid | region | age_range | zodiac |`
	`+------------------------------------------------------------------------------+`
	`| 0016359810821 | Hubei Province | 30 to 40 years old | Cancer (constellation) |`
	`| 0016359814159 | Unknown | 30 to 40 years old | Cancer (constellation) |`
	`+------------------------------------------------------------------------------+`	

`${dag.foreach.current}`	`0016359810821,Hubei Province,30 to 40 years old,Cancer (constellation)`	`0016359814159,Unknown,30 to 40 years old,Cancer (constellation)`

Built-in variable	Value obtained in the first loop	Value obtained in the second loop
`${dag.offset}`	0	1
`${dag.loopTimes}`	1	2
`${dag.dag.foreach.current[1]}`	`0016359810821`	`0016359814159`
`${dag.loopDataArray[1][0]}`	`0016359814159`	

5.11.2.2. Configure a for-each node

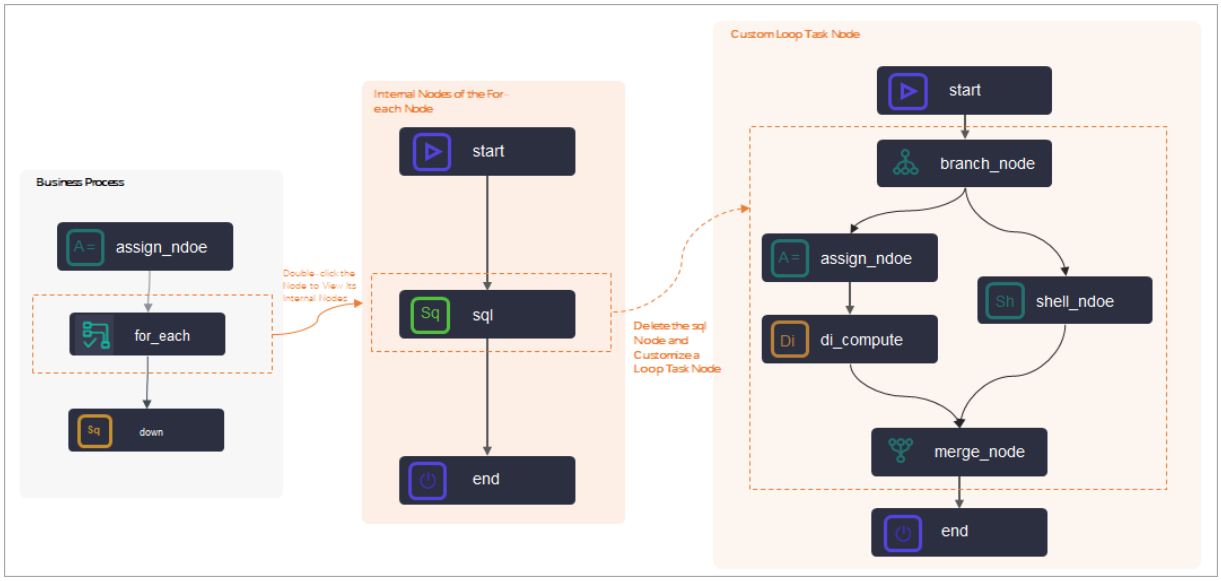
DataWorks provides for-each nodes. You can use a for-each node to traverse the output of an assignment node in loops. You can also customize the workflow in a for-each node. This topic provides an example on how to configure and use a for-each node. In this example, the for-each node is used to traverse the output of an assignment node in two loops, and the system displays the current number of loops in the Operation Center for each loop.

Prerequisites

DataWorks of the Standard Edition or a more advanced edition is activated.

Context

In DataWorks, a for-each node is used to traverse the output of an assignment node in loops. Before you use a for-each node, you must configure the for-each node as a descendant node of an assignment node. After the assignment node passes its output to the for-each node, the for-each node traverses the output in loops. After you create a for-each node, the following three inner nodes are created: **start**, **sql**, and **end**.



You can customize the workflow in a for-each node and configure built-in variables to obtain the output of an assignment node. For more information, see Composition and application logic.

Limits

- Dependencies

  A for-each node is used to traverse the output of an assignment node in loops. You must configure the assignment node as an ancestor node of the for-each node.

- for-each nodes
  - You can use for-each nodes only in DataWorks of the Standard Edition or a more advanced edition.
  - A for-each node can traverse the output of an assignment node in a maximum of 128 loops. If the number of loops exceeds 128, an error is reported. The number of loops that are actually run by a for-each node varies based on the output of the assignment node that is configured as an ancestor node of the for-each node.
    - If the output of an assignment node is a one-dimensional array, the number of loops is the number of elements in the array.

      For example, a for-each node is configured as a descendant node of an assignment node in the Shell or Python 2 language, and the output of the assignment node is the one-dimensional array `2021-03-28,2021-03-29,2021-03-30,2021-03-31,2021-04-01`. In this case, the for-each node traverses the output in five loops.

    - If the output of an assignment node is a two-dimensional array, the number of loops is the number of rows in the array.

      For example, a for-each node is configured as a descendant node of an assignment node in the ODPS SQL language, and the output of the assignment node is a two-dimensional array. The following code shows the two-dimensional array.

      ```
      +------------------------------------------------------------------------------+
      | uid            | region         | age_range            | zodiac               |
      +------------------------------------------------------------------------------+
      | 0016359810821  | Hubei Province | 30 to 40 years old   | Cancer (constellation) |
      | 0016359814159  | Unknown        | 30 to 40 years old   | Cancer (constellation) |
      +------------------------------------------------------------------------------+
      ```

      In this case, the for-each node traverses the output in two loops.

- Inner nodes
  - You can delete the existing dependencies between the inner nodes of a for-each node and customize the workflow in the for-each node. However, you must make sure that the workflow of the for-each node starts with the **start** node and ends with the **end** node.
  - If you use a branch node as an inner node of a for-each node to perform logical judgments or traverse the output of an assignment node, you must also use a merge node.

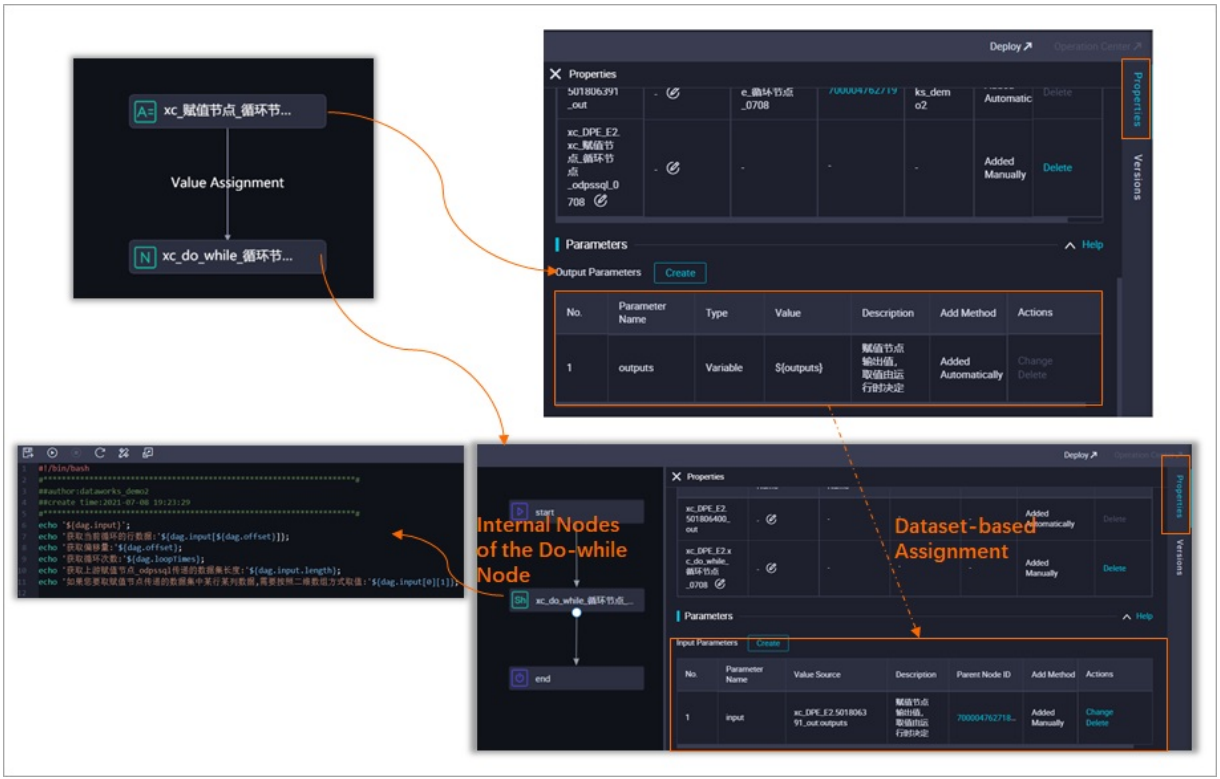
- Testing
  - If the workspace that you use is in standard mode, you cannot test a for-each node on the DataStudio page.

    To test a for-each node and view the result, you must commit and deploy the for-each node to the production environment. Then, you can view the operational logs of the for-each node on the Operation Center page.

  - To view the operational logs of a for-each node on the Operation Center page, right-click the for-each node in a directed acyclic graph (DAG) and select **View Internal Nodes**.

Procedure

Before you use a for-each node, you must configure the for-each node as a descendant node of an assignment node. The following figure shows the configuration procedure.



1. Configure node dependencies.

   Configure an assignment node as an ancestor node of a for-each node. For more information, see Create and configure a workflow.

2. Configure inputs for the for-each node.

   In the **Parameters** section of the Properties tab for the for-each node, add the **outputs** parameter of the assignment node to **Input Parameters**. For more information, see Configure an assignment node.

3. Configure the inner nodes of the for-each node.

   Customize the workflow in the for-each node based on your business requirements. Then, configure built-in variables for the inner nodes to enable the inner nodes to obtain and traverse the output of the assignment node in loops. For more information about the built-in variables, see Built-in variables. For more information about how to configure a for-each node, see Configure a for-each node.

Create and configure a workflow

To create a workflow that contains an assignment node as the ancestor node and a for-each node as the descendant node, perform the following steps:

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.

iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.

2. Create a workflow.

i. Move the pointer over the Create icon and select **Workflow**.

ii. In the **Create Workflow** dialog box, specify **Workflow Name** and **Description**.

> **Notice** The workflow name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

iii. Click **Create**.

3. Create a for-each node.

i. Move the pointer over the Create icon and choose **General > for-each**.

Alternatively, find the required workflow in the Scheduled Workflow pane, click the workflow name, right-click **General**, and then choose **Create > for-each**.

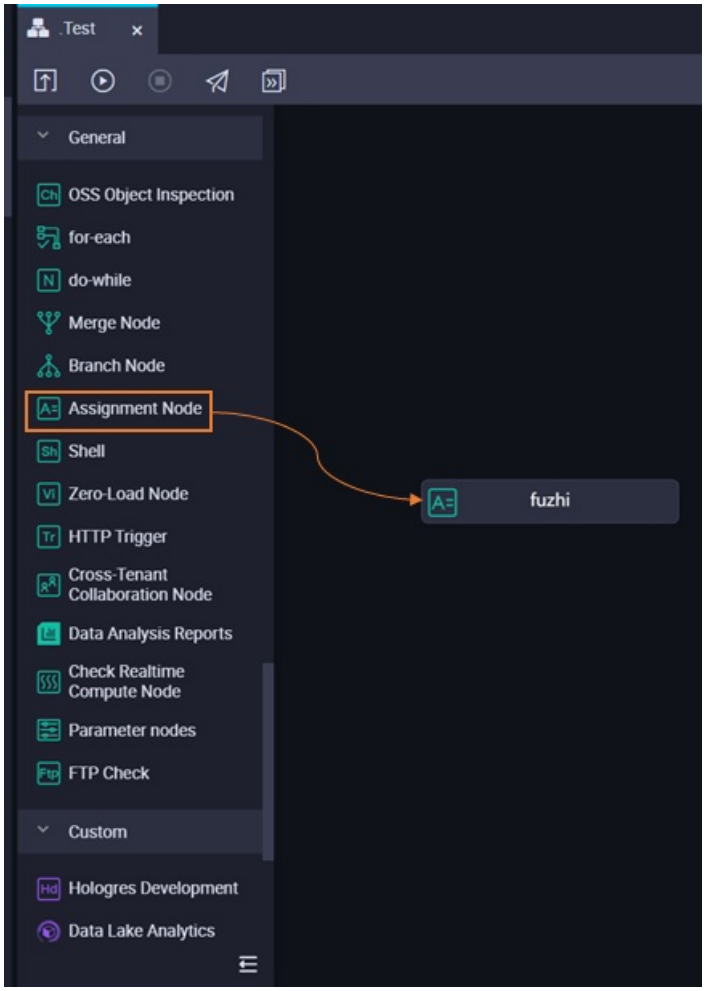
ii. In the **Create Node** dialog box, specify **Node Name** and **Location**.

> **Notice** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

iii. Click **Commit**.

4. Create an assignment node. For more information, see Configure an assignment node.

i. On the configuration tab of the created workflow, drag **Assignment Node** in the **General** section to the canvas on the right side.

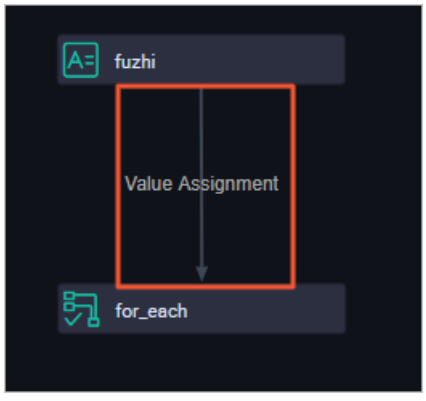


ii. In the **Create Node** dialog box, specify **Node Name** and **Location**. The default value of the Location parameter is the path of the current workflow.

> **Notice** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

iii. Click **Commit**.

5. Drag a directed line to configure the assignment node as an ancestor node of the for-each node.

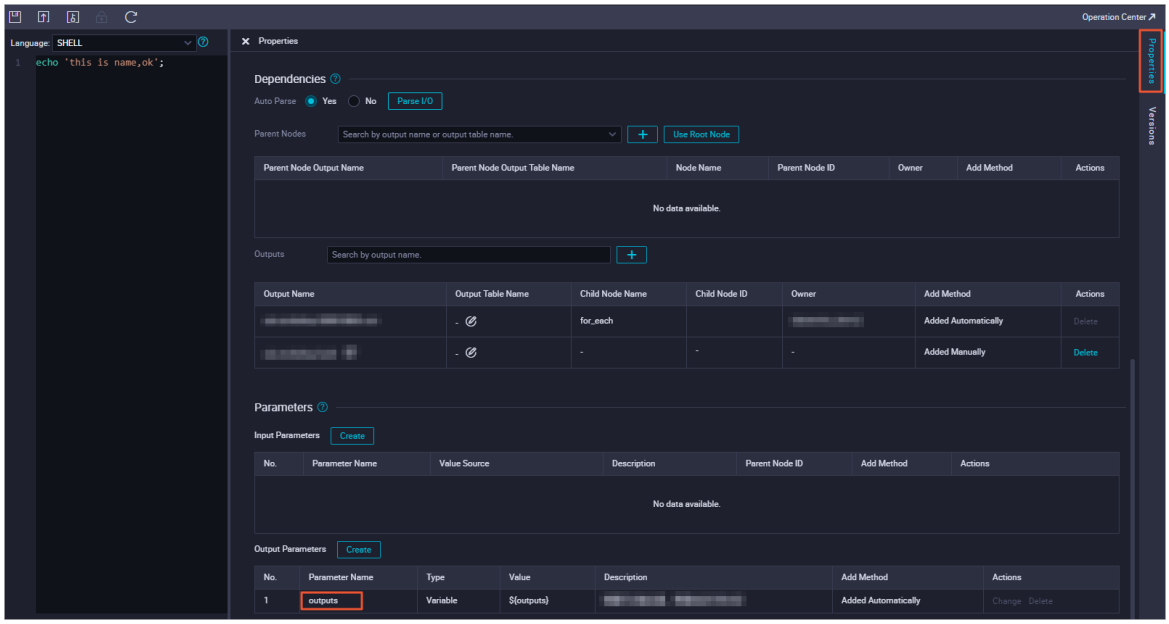


Configure an assignment node

1. On the configuration tab of the created workflow, double-click the name of the assignment node that you created. The configuration tab of the assignment node appears.
2. Select **SHELL** from the **Language** drop-down list.
3. Enter the following statement in the code editor:

```
echo 'this is name,ok';
```

4. In the right-side navigation pane, click the **Properties** tab. In the **Parameters** section, view the information about the outputs parameter below **Output Parameters**. The outputs parameter is the default output parameter of the assignment node.



5. Click the icon in the top toolbar to save the assignment node.
6. Commit the assignment node.

> **Notice** You must specify **Rerun** and **Parent Nodes** on the Properties tab before you commit the assignment node.

   i. Click the icon in the top toolbar.
   ii. In the **Commit Node** dialog box, specify **Change description**.
   iii. Click **OK**.

If the workspace that you use is in standard mode, you must click **Deploy** in the upper-right corner to deploy the node after you commit the node. For more information, see Deploy nodes.

Configure a for-each node

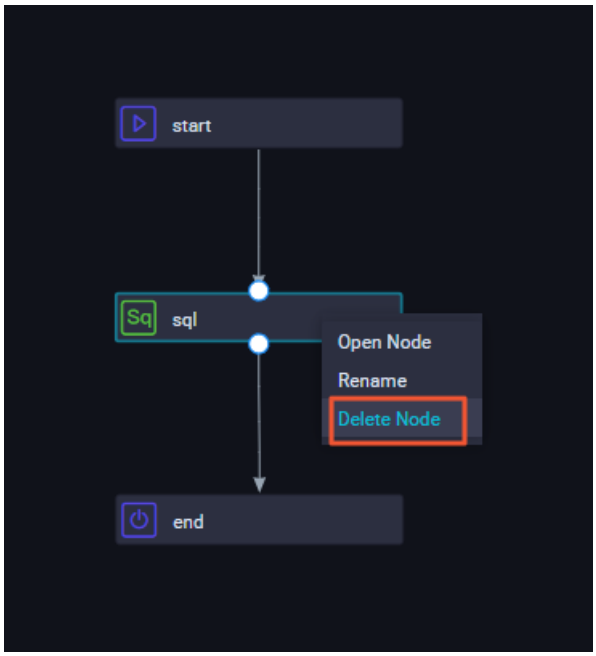
1. Double-click the for-each node that you created. By default, the start, sql, and end nodes are displayed on the configuration tab of the for-each node.

2. Delete the sql node.

   You can use a node other than an SQL node in the workflow of the for-each node.

   - If you want to use an ODPS SQL node, skip this step.
   - If you want to use a node other than an SQL node, delete the sql node first. In this example, a Shell node is used.

     i. Right-click the sql node and select **Delete Node**.

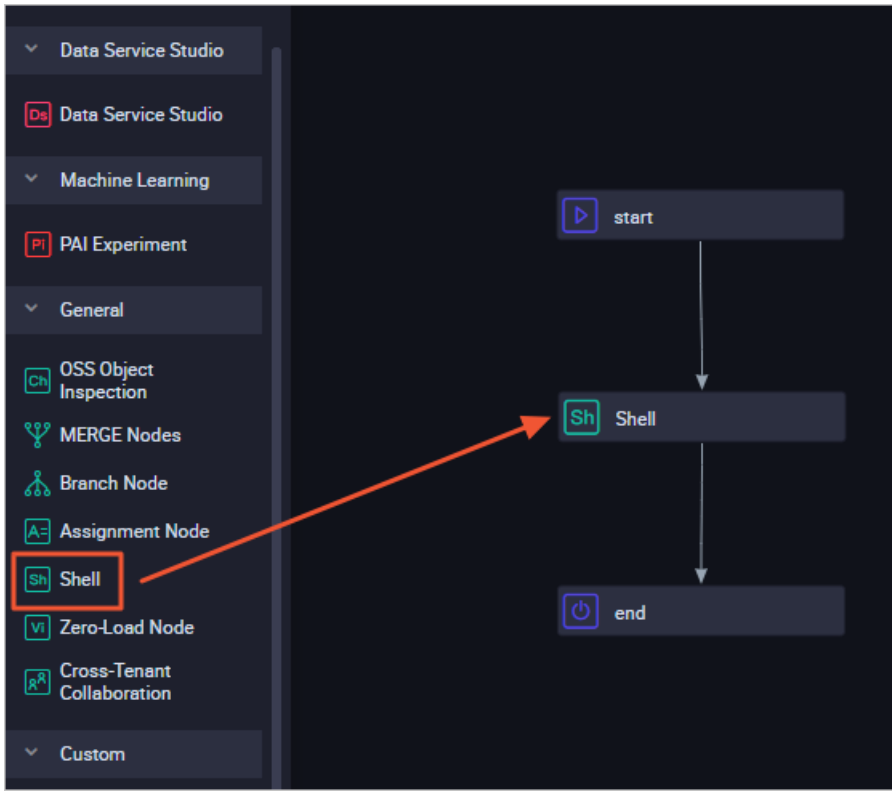
     

     ii. In the **Delete** message, click **OK**.

3. Create and configure a Shell node.

   You can create other types of nodes by using the same method. If you want to use the default sql node, skip this step.

i. Drag **Shell** in the **General** section to the configuration tab of the for-each node.



ii. In the **Create Node** dialog box, specify **Node Name**.

> **Notice** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

iii. Click **Create**.

iv. Drag directed lines to configure the start node as an ancestor node of the Shell node and the end node as a descendant node of the Shell node.

v. Double-click the Shell node. The configuration tab of this node appears.

vi. Enter the following code:

```
echo ${dag.loopTimes} ----Display the current number of loops.
```

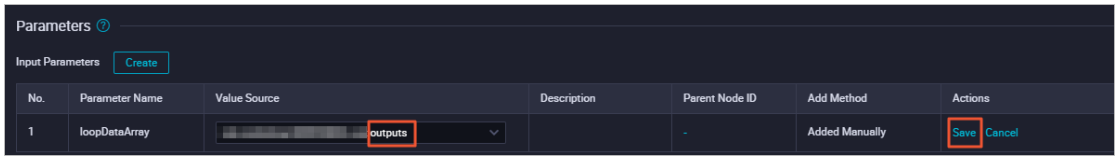
> **Note**
> - The start and end nodes in the workflow of the for-each node have fixed logic and cannot be modified.
> - After you modify the code of the Shell node, save the modification. No message that reminds you to save the modification will appear when you commit the node. If you do not save the modification, the code cannot be immediately updated to the latest version.

A for-each node supports the following environment variables:

- ${dag.foreach.current}: the current data entry.
- ${dag.loopDataArray}: the output of an assignment node.
- ${dag.offset}: the offset of the current number of loops to 1.
- ${dag.loopTimes}: the current number of loops. The value is equivalent to the value of ${dag.offset} plus 1.

For more information about the variables, see Built-in variables and Examples of variable values.

4. Configure the scheduling properties of the for-each node.
   i. On the configuration tab of the for-each node, click the **Properties** tab in the right-side navigation pane.
   ii. Find the loopDataArray parameter below **Input Parameters** in the **Parameters** section and click **Change** in the Actions column. The loopDataArray parameter is the default input parameter of the for-each node.
   iii. Select the outputs parameter of the assignment node from the drop-down list in the **Value Source** column.



> **Note** After you configure the assignment node as an ancestor node of the for-each node, you must specify the input parameter for the for-each node on the Properties tab. If you do not specify the input parameter, an error occurs when you commit the for-each node.

   iv. Click **Save**.

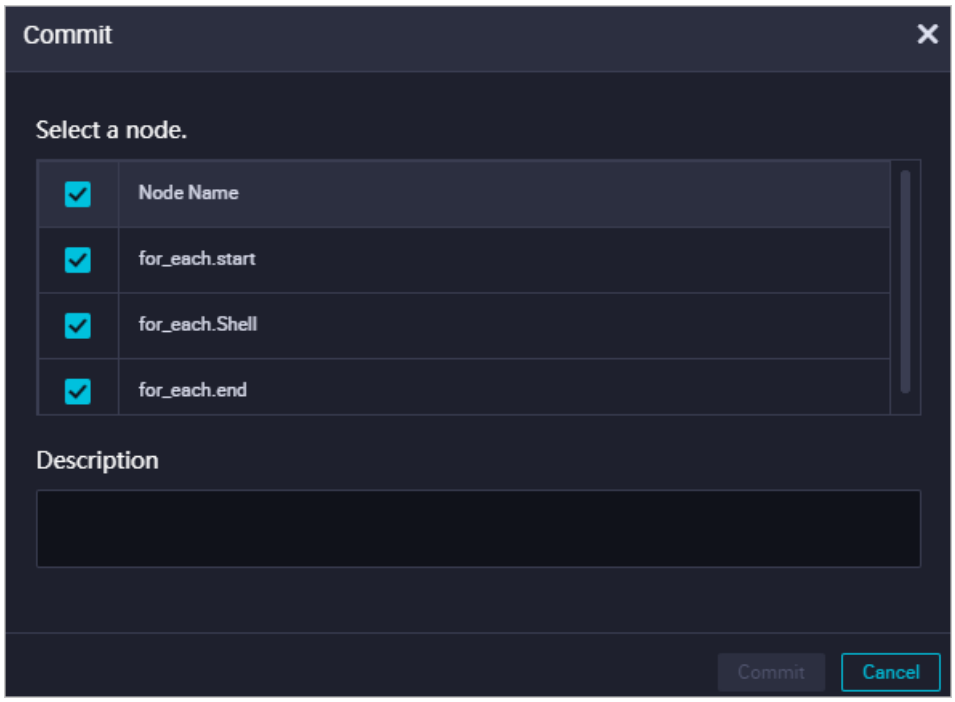
5. Click the icon in the top toolbar to save the for-each node.

6. Commit the for-each node.

> **Notice** You must specify **Rerun** and **Parent Nodes** on the Properties tab before you commit the node.

i. Click the icon in the top toolbar.

ii. In the **Commit** dialog box, select the inner nodes that you want to commit and enter your comments in the **Description** field.



iii. Click **Commit**.

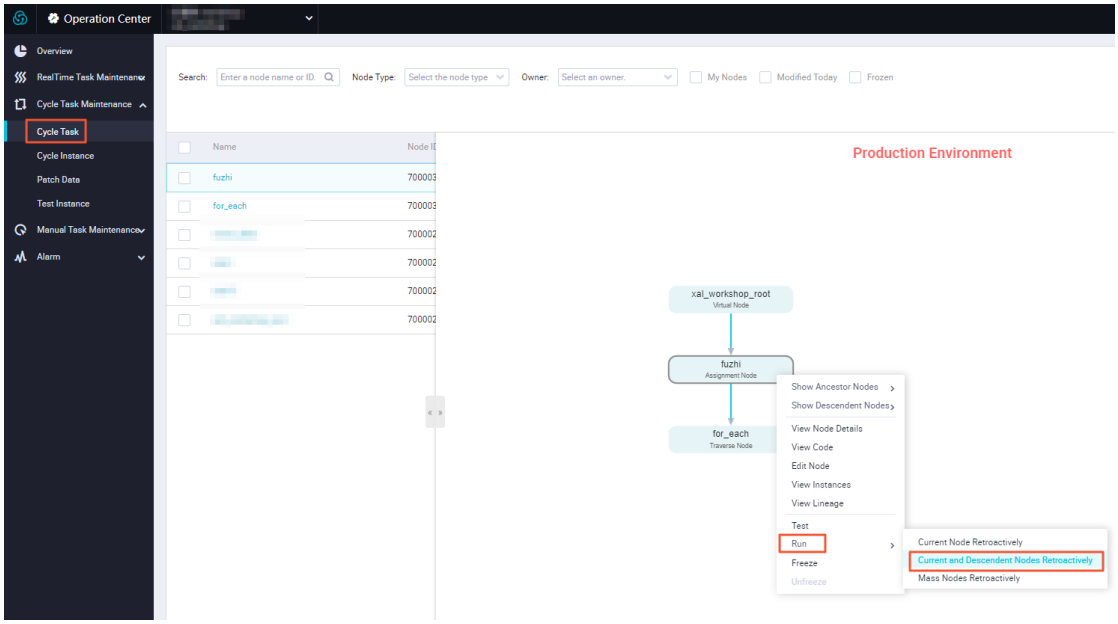
If the workspace that you use is in standard mode, you must click **Deploy** in the upper-right corner to deploy the for-each node after you commit the for-each node. For more information, see Deploy nodes.

7. Test the for-each node and view the result.

i. On the configuration tab of the for-each node, click **Operation Center** in the upper-right corner.

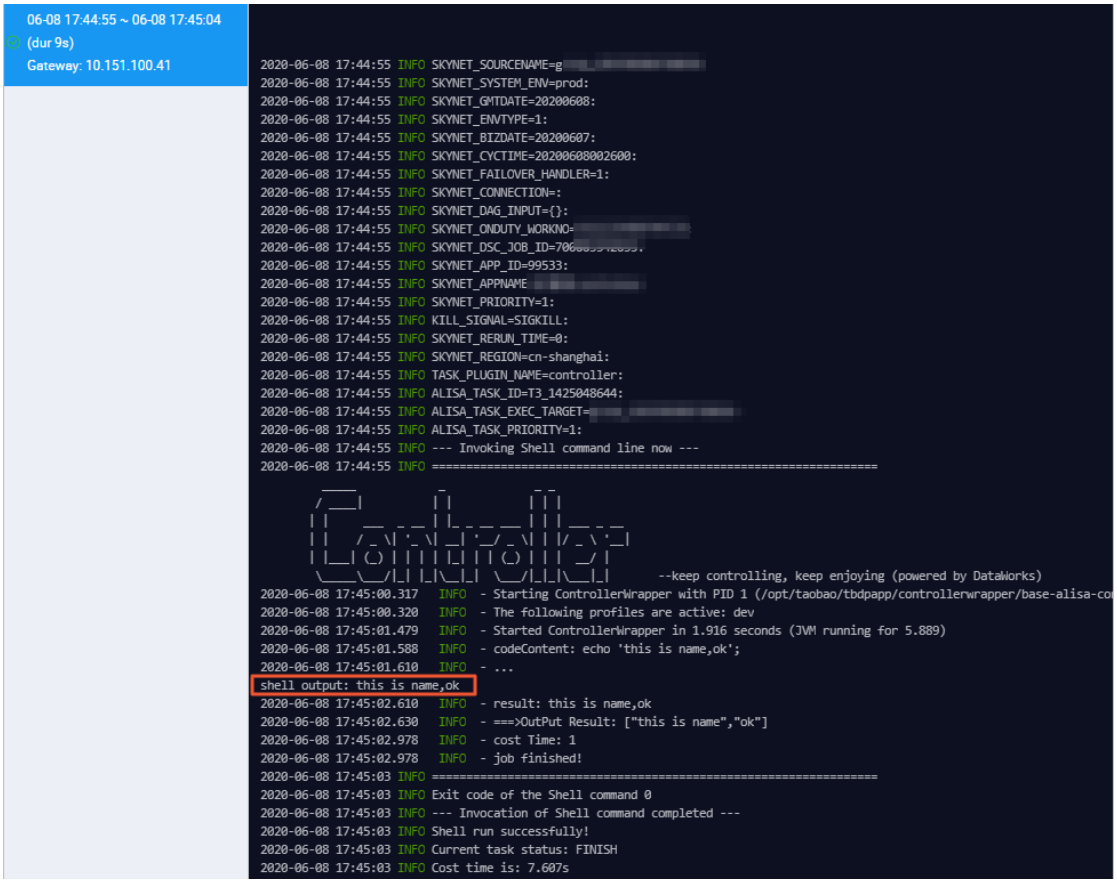
ii. In the left-side navigation pane of the Operation Center page, choose **Cycle Task Maintenance > Cycle Task**.

iii. On the page that appears, click the name of the for-each node in the node list. In the directed acyclic graph (DAG) on the right side, right-click the assignment node and choose **Run > Current and Descendant Nodes Retroactively**. In the Patch Data dialog box, select the assignment node and for-each node and click OK.

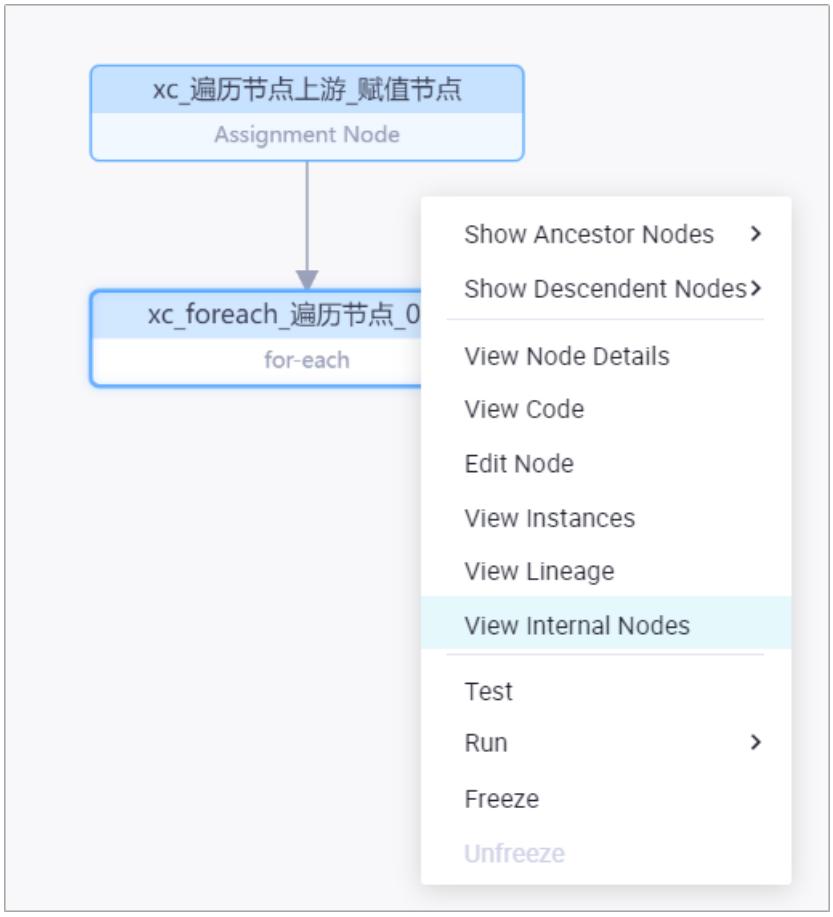


iv. On the **Patch Data** page, wait until the data backfill node instance is run. Then, click **DAG** in the Actions column of the instance.

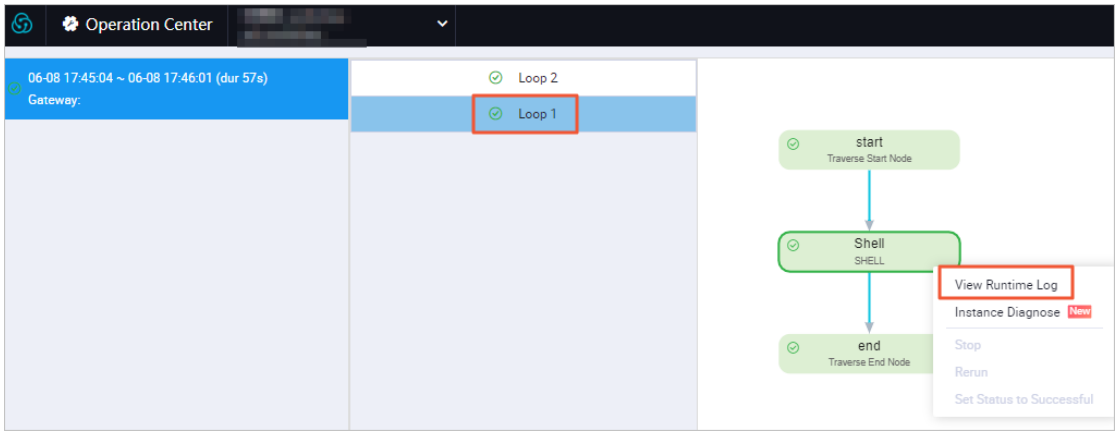
v. In the DAG that appears, right-click the assignment node and select **View Runtime Log** to view the operational logs of the node.


06-08 17:44:55 ~ 06-08 17:45:04
(dur 9s)
Gateway: 10.151.100.41
2020-06-08 17:44:55 INFO SKYNET_SOURCENAME=g
2020-06-08 17:44:55 INFO SKYNET_SYSTEM_ENV=prod:
2020-06-08 17:44:55 INFO SKYNET_GMTDATE=20200608:
2020-06-08 17:44:55 INFO SKYNET_ENVTYPE=1:
2020-06-08 17:44:55 INFO SKYNET_BIZDATE=20200607:
2020-06-08 17:44:55 INFO SKYNET_CYCTIME=20200608002600:
2020-06-08 17:44:55 INFO SKYNET_FAILOVER_HANDLER=1:
2020-06-08 17:44:55 INFO SKYNET_CONNECTION=:
2020-06-08 17:44:55 INFO SKYNET_DAG_INPUT={}:
2020-06-08 17:44:55 INFO SKYNET_ONDUTY_WORKNO=
2020-06-08 17:44:55 INFO SKYNET_DSC_JOB_ID=70
2020-06-08 17:44:55 INFO SKYNET_APP_ID=99533:
2020-06-08 17:44:55 INFO SKYNET_APPNAME
2020-06-08 17:44:55 INFO SKYNET_PRIORITY=1:
2020-06-08 17:44:55 INFO KILL_SIGNAL=SIGKILL:
2020-06-08 17:44:55 INFO SKYNET_RERUN_TIME=0:
2020-06-08 17:44:55 INFO SKYNET_REGION=cn-shanghai:
2020-06-08 17:44:55 INFO TASK_PLUGIN_NAME=controller:
2020-06-08 17:44:55 INFO ALISA_TASK_ID=T3_1425048644:
2020-06-08 17:44:55 INFO ALISA_TASK_EXEC_TARGET=
2020-06-08 17:44:55 INFO ALISA_TASK_PRIORITY=1:
2020-06-08 17:44:55 INFO --- Invoking Shell command line now ---
2020-06-08 17:44:55 INFO ==
--keep controlling, keep enjoying (powered by DataWorks)
2020-06-08 17:45:00.317 INFO - Starting ControllerWrapper with PID 1 (/opt/taobao/tbdpapp/controllerwrapper/base-alisa-co
2020-06-08 17:45:00.320 INFO - The following profiles are active: dev
2020-06-08 17:45:01.479 INFO - Started ControllerWrapper in 1.916 seconds (JVM running for 5.889)
2020-06-08 17:45:01.588 INFO - codeContent: echo 'this is name,ok';
2020-06-08 17:45:01.610 INFO - ...
shell output: this is name,ok
2020-06-08 17:45:02.610 INFO - result: this is name,ok
2020-06-08 17:45:02.630 INFO - ===>OutPut Result: ["this is name","ok"]
2020-06-08 17:45:02.978 INFO - cost Time: 1
2020-06-08 17:45:02.978 INFO - job finished!
2020-06-08 17:45:03 INFO ==
2020-06-08 17:45:03 INFO Exit code of the Shell command 0
2020-06-08 17:45:03 INFO --- Invocation of Shell command completed ---
2020-06-08 17:45:03 INFO Shell run successfully!
2020-06-08 17:45:03 INFO Current task status: FINISH
2020-06-08 17:45:03 INFO Cost time is: 7.607s

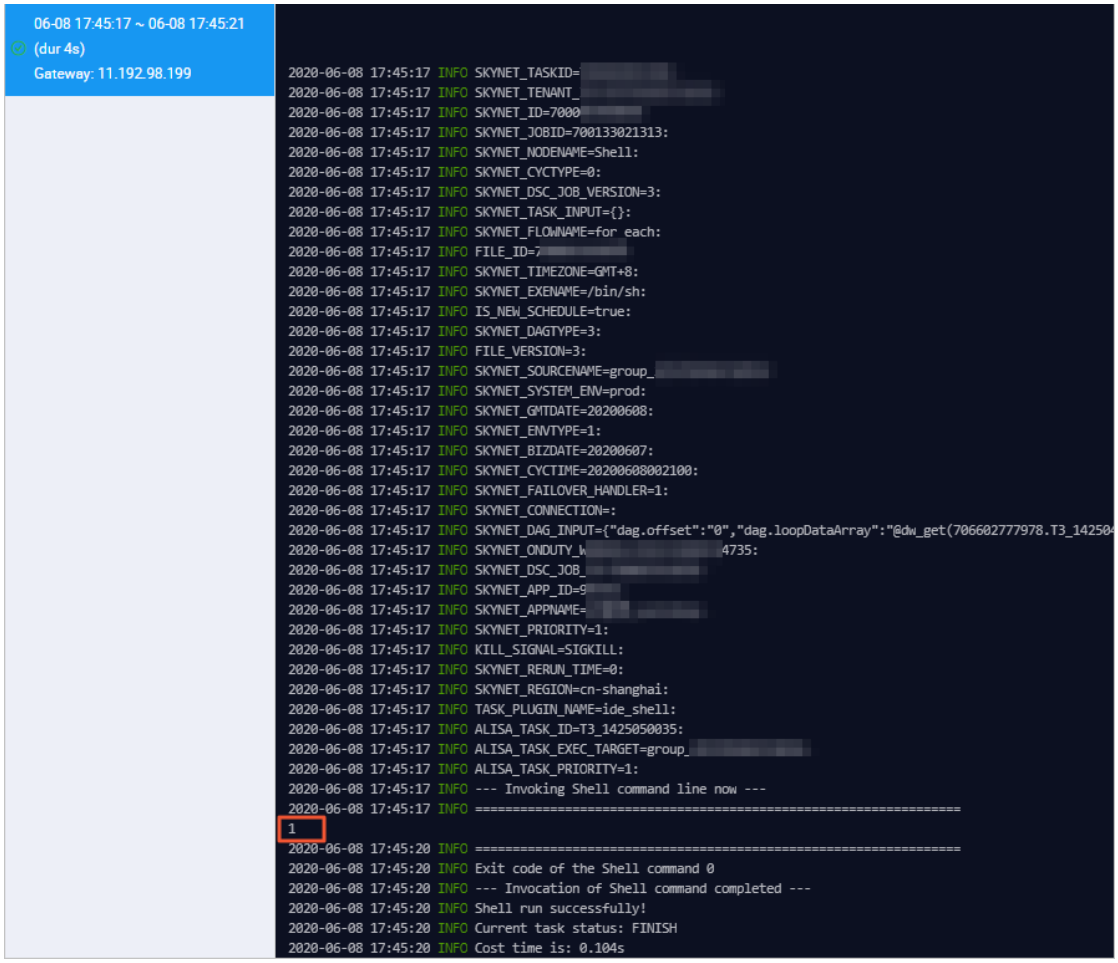
vi. On the **Patch Data** page, right-click the for-each node in the DAG and select **View Internal Nodes**.



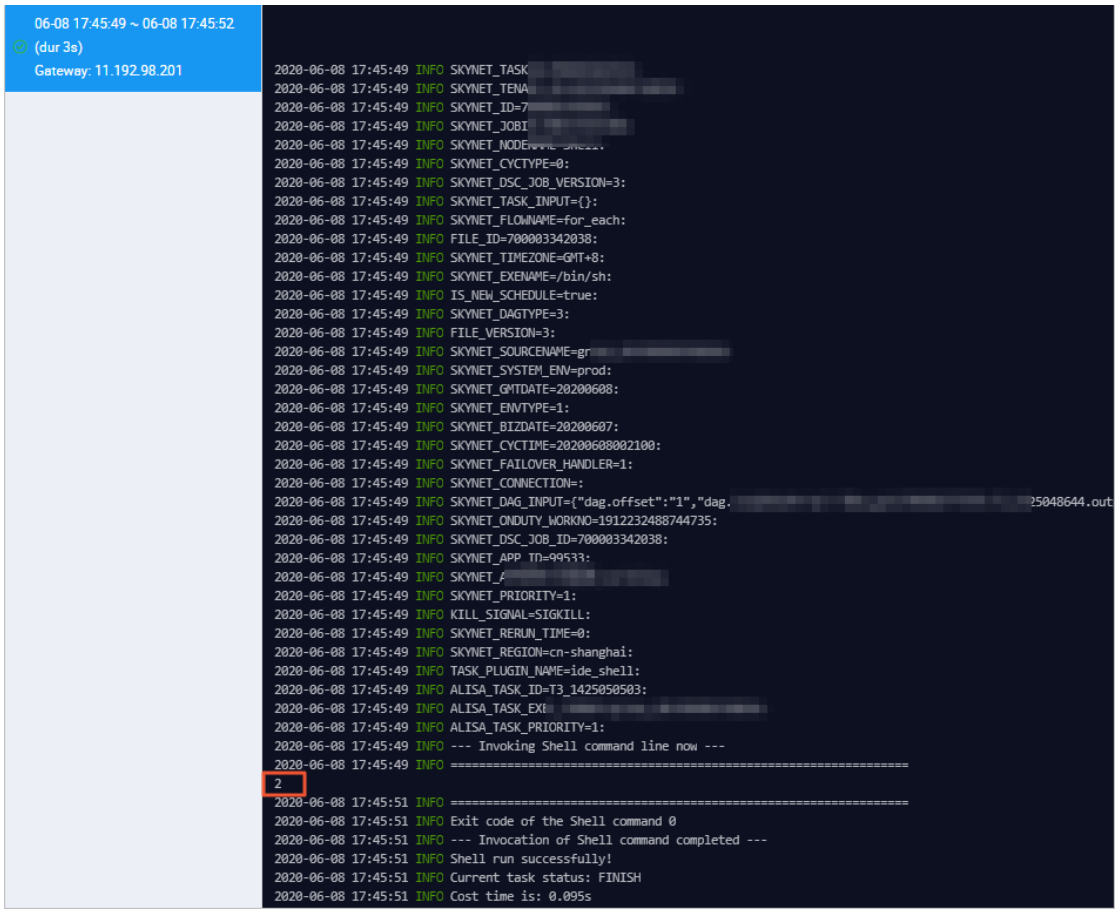
vii. On the page that appears, click **Loop 1** in the middle pane, right-click the Shell node in the DAG, and then select **View Runtime Log**.



On the page that appears, view the operational logs of the Shell node in the first loop.



viii. Use the same method to view the operational logs of the Shell node in the second loop.



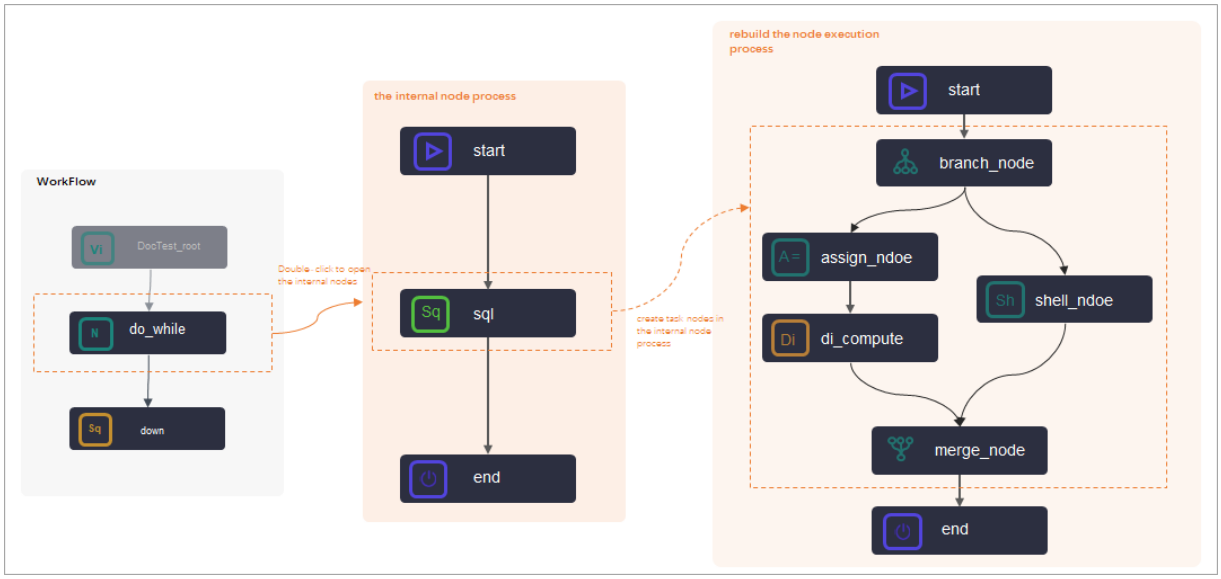
5.11.3. Configure a do-while node

5.11.3.1. Logic principles

DataWorks provides do-while nodes. You can rearrange the workflow inside a do-while node, write the logic to be executed in a loop in the node, and then configure an end node to determine whether to exit the loop. You can also use a do-while node together with an assignment node to loop through the result set that is passed by the assignment node. This topic describes the composition and application logic of do-while nodes.

Node composition

A do-while node in DataWorks is a special node that contains internal nodes. When you create a do-while node, the following three internal nodes are automatically created: the **start** node (loop start node), the **sql** node (loop task node), and the **end** node (loop end node). The internal nodes are organized into an internal node process to perform the task in a loop.



The preceding figure shows the following information:

- **start node**

  The start node in the internal nodes does not carry specific task code.

- **sql node**

  By default, DataWorks creates an internal SQL task node. You can delete the default **sql** node and customize internal loop task nodes.

  - If your loop task is an SQL task, you can double-click the default **sql** node to go to the node configuration tab to develop loop task code.
  - If your loop task is complex, you can create task nodes in the internal node process and rebuild the node execution process based on actual situations.

    Generally, a do-while node is used together with an assignment node, a branch node, and a merge node. For more information about typical scenarios, see Typical scenario: Use a do-while node together with an assignment node.

    > **Note** When you customize a do-while node, you can delete the dependencies between the internal nodes and rearrange the internal workflow of the do-while node. However, you must use the **start** node and the **end** node as the start and end nodes of the internal workflow of the do-while node.

- **end node**

  - The **end** node is used to determine the number of times the loop is run for the do-while node. The end node is essentially an assignment node. The end node returns `true` or `false`. true indicates to run the loop again, and false indicates to exit the loop.

- You can use MaxCompute SQL, Shell, or Python 2 to develop the code of the **end** node. The do-while node provides convenient built-in variables for you to develop the end code. For more information about built-in variables, see Built-in variables and Examples of variable values. For more information about sample codes developed in different languages, see Example 1: Sample code of the end node.

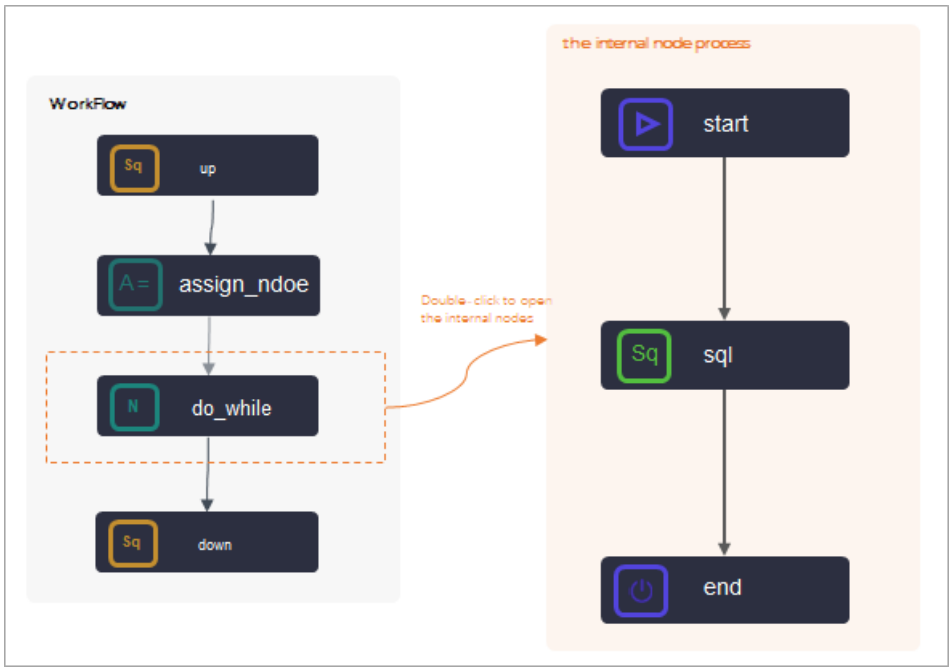
Limits and usage nodes

- Support for do-while nodes
  - You can use do-while nodes only in DataWorks Standard Edition or a more advanced edition.
  - A do-while node supports a maximum of 128 times the loop is run. If the number of times the loop is run determined by the **end** node exceeds 128, an error is returned.
- Internal nodes
  - When you customize a do-while node, you can delete the dependencies between the internal nodes and rearrange the internal workflow of the do-while node. However, you must use the **start** node and the **end** node as the start and end nodes of the internal workflow of the do-while node.
  - When the internal nodes of a do-while node use a branch node to perform logical judgments or traverse results, a merge node also needs to be used.
  - You cannot add comments when you develop the code of the **end** node of a do-while node.
- Test and running
  - If the workspace is in standard mode, you cannot directly test and run a do-while node in DataStudio.

    To test the do-while node and view the result, you must commit the do-while node to Operation Center and run the do-while node in Operation Center. If you use the value passed by an assignment node in the do-while node, run both the assignment node and do-while node during the test in Operation Center.
  - When you view the operational logs of a do-while node in Operation Center, right-click the do-while node and select **View Internal Nodes** to view the operational logs of the internal nodes.

Typical scenario: Use a do-while node together with an assignment node

A do-while node is often used together with an assignment node, as shown in the following figure.



When you use a do-while node together with an assignment node:

- You must use the output of the assignment node as the input of the do-while node, and configure the dependencies of the do-while node on the assignment node. For more information about usage notes, see Example 2: Use a do-while node together with an assignment node.
- You can use some built-in variables to obtain the current number of times the loop is run and the values of the assignment parameters. For more information, see Built-in variables.

Built-in variables

A do-while node in DataWorks uses internal nodes to run a task in a loop. Each time the task is run in a loop, you can use some built-in variables to obtain the current number of times the loop is run and the offset.

Built-in variable	Description	Value
`${dag.loopTimes}`	The current number of times the loop is run.	1 when the loop is run for the first time, 2 when the loop is run for the second time, 3 when the loop is run for the third time, ..., and n when the loop is run for the nth time.
`${dag.offset}`	The offset.	0 when the loop is run for the first time, 1 when the loop is run for the second time, 2 when the loop is run for the third time, ..., and n-1 when the loop is run for the nth time.

If you use a do-while node together with an assignment node, you can also obtain the values of the assignment parameters and loop variables in the following way:

> **Note** In the following variable examples, *input* specifies the name of the input parameter defined in the do-while node. You must replace input with the actual name of the input parameter.

Built-in variable	Description
`${dag.input}`	The dataset passed by the parent assignment node.
`${dag.input[${dag.offset}]}`	The data row obtained by the do-while node in the current loop.
`${dag.input.length}`	The length of the dataset obtained inside the do-while node.

Examples of variable values

- Example 1

  The parent assignment node is a Shell node, and the last output is `2021-03-28,2021-03-29,2021-03-30,2021-03-31,2021-04-01`. The following table describes the values of the variables in Example 1.

Built-in variable	Value when the loop is run for the first time	Value when the loop is run for the second time
`${dag.input}`	`2021-03-28,2021-03-29,2021-03-30,2021-03-31,2021-04-01`	
`${dag.input[${dag.offset}]}`	`2021-03-28`	`2021-03-29`
`${dag.input.length}`	5	
`${dag.loopTimes}`	1	2
`${dag.offset}`	0	1

- Example 2

  The parent assignment node is an ODPS SQL node, and the last SELECT statement queries the following two pieces of data:

```
+------------------------------------------------+
| uid            | region | age_range | zodiac |
+------------------------------------------------+
| 0016359810821  | Hubei Province | 30-40 years old   | Cancer (constellation) |
| 0016359814159  | Unknown   | 30-40 years old   | Cancer (constellation) |
+------------------------------------------------+
```

  The following table describes the values of the variables in Example 2.

Built-in variable	Value when the loop is run for the first time	Value when the loop is run for the second time

Built-in variable	Value when the loop is run for the first time	Value when the loop is run for the second time

`${dag.input}`	`+------------------------------------------+`
	`| uid | region | age_range | zodiac |`
	`+------------------------------------------+`
	`| 0016359810821 | Hubei Province | 30-40 years old | Cancer (constellation) |`
	`| 0016359814159 | Unknown | 30-40 years old | Cancer (constellation) |`
	`+------------------------------------------+`	

`${dag.input[${dag.offset}]}`	`0016359810821, Hubei Province, 30-40 years old, Cancer (constellation)`	`0016359814159, unknown, 30-40 years old, Cancer (constellation)`
`${dag.input.length}`	2 **Note** The number of rows in a two-dimensional array is the length of the dataset. The number of rows in a two-dimensional array in the output of the assignment node is 2.	
`${dag.input[0][1]}` **Note** The value in the first row and first column of the two-dimensional array.	`0016359810821`	
`${dag.loopTimes}`	1	2
`${dag.offset}`	0	1

Example 1: Sample code of the end node

You can use MaxCompute SQL, Shell, or Python 2 to develop the code of the **end** node. The following part shows typical sample code in these three different languages.

- MaxCompute SQL:

```
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';
```

In the preceding sample code of the end node, the number of rows and the offset are compared with fixed values to limit the number of times the loop is run for the do-while node.

- Shell:

```
if [ ${dag.loopTimes} -lt 5 ];
then
     echo "True"
else
     echo "False"
fi
```

In the preceding code, the number of times the loop is run is compared with 5 to limit the number of times the loop is run for the do-while node. The `${dag.loopTimes}` variable specifies the number of times the loop is run.

The value of the `${dag.loopTimes}` variable is 1 when the loop is run for the first time and is incremented by 1 each time, for example, 2 for the second time, and 5 for the fifth time. At this point, the output of the end node is false, and the do-while node exits the loop.

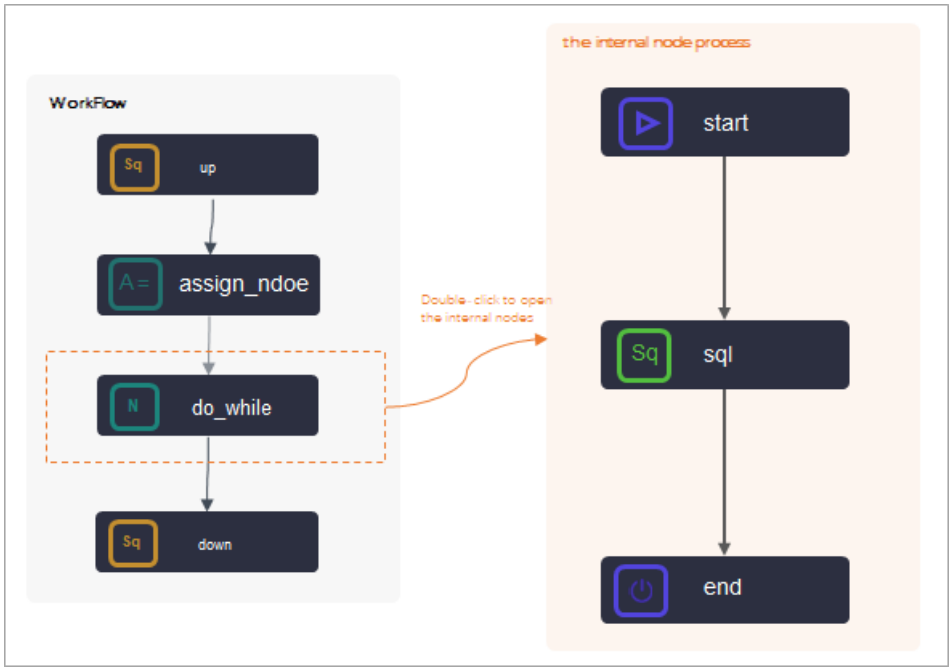
- Python 2:

```
if ${dag.loopTimes}<${dag.input.length}:
   print True;
else
   print False;
# Run the loop again if the end node returns True.
# Exit the loop if the end node returns False.
```

In the preceding code, the number of times the loop is run is compared with the number of rows in the dataset passed by the assignment node to limit the number of times the loop is run for the do-while node. The `${dag.loopTimes}` variable specifies the number of times the loop is run.

Example 2: Use a do-while node together with an assignment node

The following table describes a typical scenario and usage notes for using a do-while node together with an assignment node.

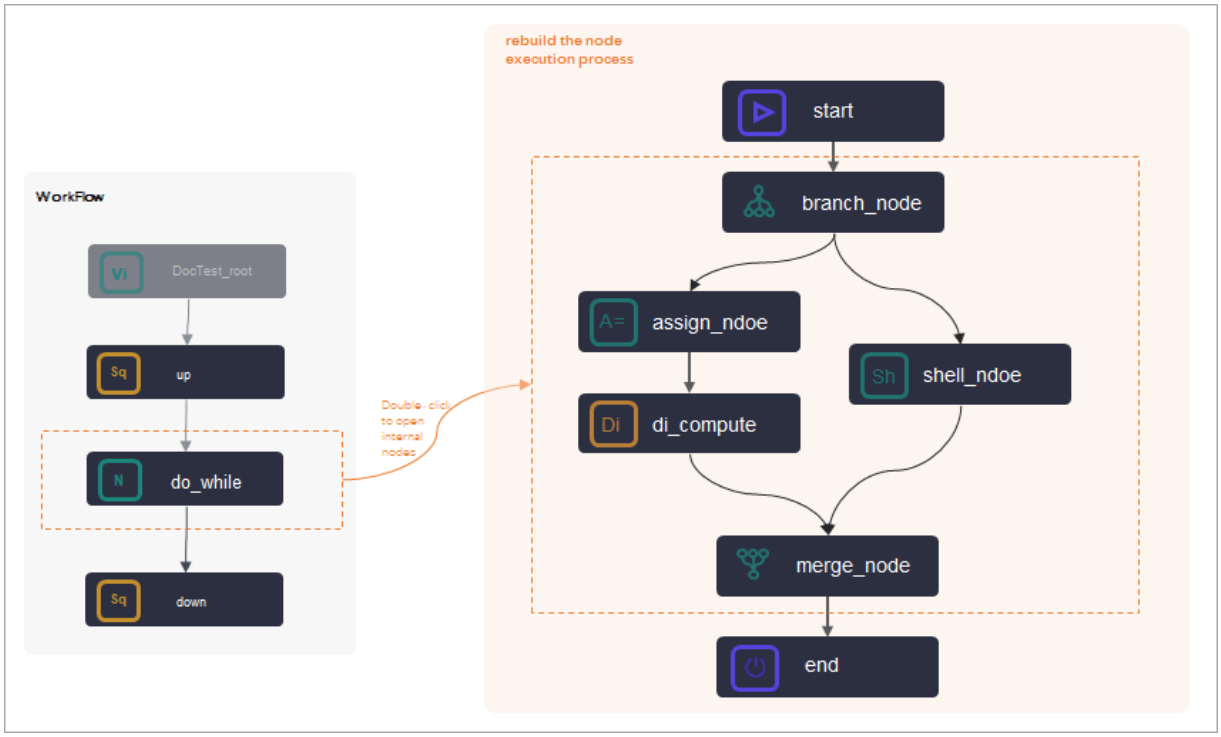


Scenario	Usage note	Configuration example

Scenario	Usage note	Configuration example
When a `do-while` node is used to perform a loop task, each time the loop is run, the internal nodes need to obtain and use the output parameters of the parent node (`up node`). In this case, you can use an assignment node (`assign_node`).	• Dependencies The do-while node depends on the parent assignment node (assign_node). **Note** The do-while node rather than the `sql` node of the do-while node depends on the assignment node, as shown in the preceding figure. • Context parameters ○ The assignment node must use the output parameters as the output parameters of the assignment node (`assign_node`). ○ The output parameters of the assignment node must be added as the input parameters of the `sql` node of the do-while node. **Note** You must set context relationships for the internal loop task node rather than the do-while node.	

Example 3: Use a do-while node together with a branch node and a merge node

The following table describes a typical scenario and usage notes for using a do-while node together with a branch node and a merge node.



Scenario	Usage note
The do-while node needs to perform logical judgment or result traversal. In this case, you can customize the loop task node in the internal node of the do-while node and use a branch node (`branch_node`) and a merge node (`merge_node`).	In the do-while node, the branch node (`branch_node`) and the merge node (`merge_node`) need to be used at the same time.

5.11.3.2. Configure a do-while node

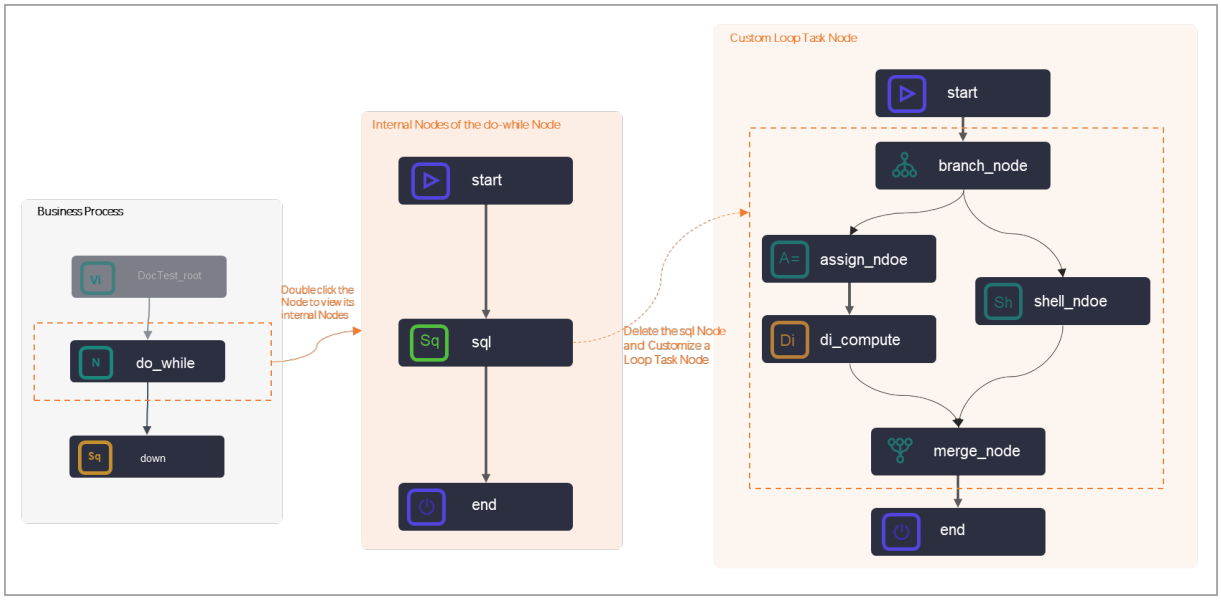
DataWorks provides do-while nodes. You can rearrange the workflow inside a do-while node, write the logic that you want to execute in a loop in the node, and then configure an end node to determine whether to exit from looping. You can also use a do-while node together with an assignment node to traverse the output of the assignment node in loops. This topic provides examples on how to configure a do-while node in simple and complex scenarios.

Prerequisites

DataWorks Standard Edition or a more advanced edition is activated.

Context

In DataWorks, do-while nodes are a special type of node that contains inner nodes. After you create a do-while node, the following three inner nodes are created: **start**, **sql**, and **end**. The start node marks the start of a loop. The sql node runs a loop. The end node marks the end of a loop and controls the number of loops to run. The three inner nodes are organized as a workflow to traverse data in loops.



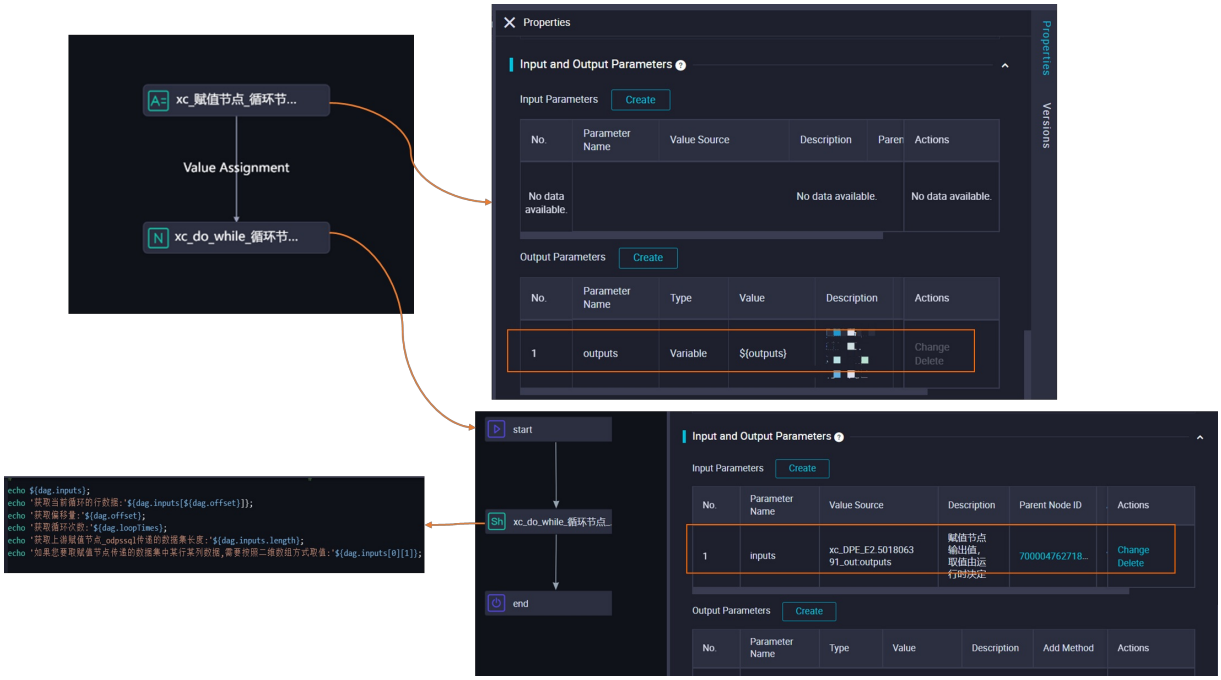
You can customize the sql node and use the built-in variables provided by the do-while node to write the code of the end node. For more information about logic principles, see Logic principles. You can plan the workflow inside your do-while node based on your business requirements. For more information about how to configure a do-while node, see the procedures described in the following sections.

Limits and usage nodes

- Support for do-while nodes
  - You can use do-while nodes only in DataWorks Standard Edition or a more advanced edition.
  - A do-while node supports a maximum of 128 times the loop is run. If the number of times the loop is run determined by the **end** node exceeds 128, an error is returned.
- Internal nodes
  - When you customize a do-while node, you can delete the dependencies between the internal nodes and rearrange the internal workflow of the do-while node. However, you must use the **start** node and the **end** node as the start and end nodes of the internal workflow of the do-while node.
  - When the internal nodes of a do-while node use a branch node to perform logical judgments or traverse results, a merge node also needs to be used.
  - You cannot add comments when you develop the code of the **end** node of a do-while node.
- Test and running
  - If the workspace is in standard mode, you cannot directly test and run a do-while node in DataStudio.

    To test the do-while node and view the result, you must commit the do-while node to Operation Center and run the do-while node in Operation Center. If you use the value passed by an assignment node in the do-while node, run both the assignment node and do-while node during the test in Operation Center.
  - When you view the operational logs of a do-while node in Operation Center, right-click the do-while node and select **View Internal Nodes** to view the operational logs of the internal nodes.

Procedure



1. Configure node dependencies.

   Configure an assignment node as an ancestor node of a do-while node.

2. Configure inputs for the do-while node.

   In the **Input and Output Parameters** section of the Properties tab for the do-while node, add the **outputs** parameter of the assignment node to **Input Parameters**.

3. Configure the inner nodes of the do-while node.

   Customize the workflow inside the do-while node based on your business requirements. Then, configure built-in variables for the inner nodes of the do-while node to obtain and traverse the output of the assignment node in loops.

Create a do-while node

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. In the **Scheduled Workflow** pane of the DataStudio page, move the pointer over the **Create** icon and choose **General > do-while**.

   Alternatively, you can click the name of the workflow in which you want to create a do-while node, right-click **General**, and then choose **Create > do-while**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.

Simple example of using a do-while node

This section describes how to use a do-while node to traverse the output of an assignment node in five loops and display the current number of loops each time a loop is run.

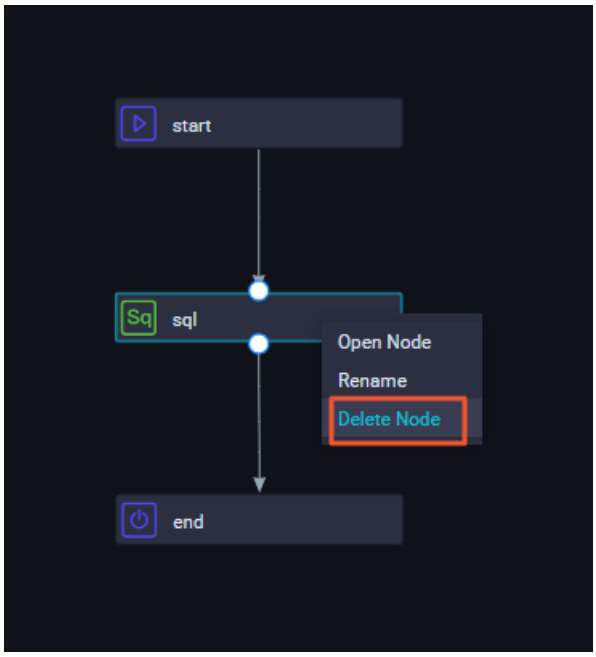
1. Double-click the name of the do-while node. The configuration tab of the node appears.

   By default, the do-while node consists of the start, sql, and end nodes.

   - The start node marks the start of a loop and does not run business code.
   - The sql node is a sample business processing node provided by DataWorks. You can replace the sql node based on your business requirements. For example, you can replace this node with a Shell node named Display loop count.
   - The end node marks the end of a loop and determines whether to start the next loop. The end node defines the condition for exiting from looping for the do-while node.

2. Delete the sql node.

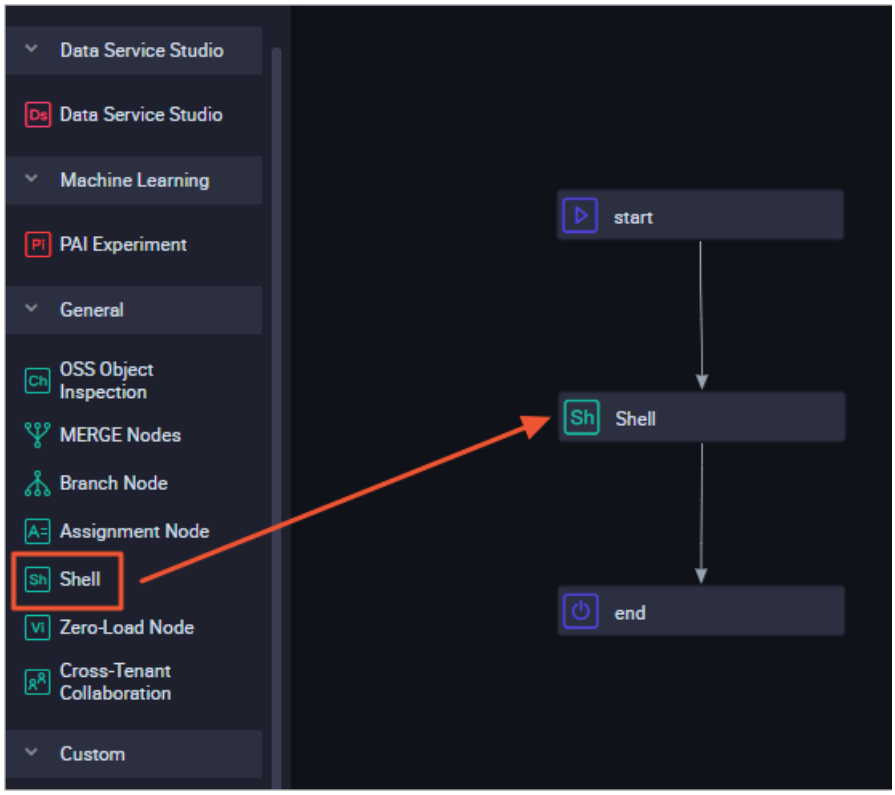
   i. Right-click the sql node and select **Delete Node**.

   ii. In the **Delete** message, click **OK**.

3. Create and configure a loop task node. In this example, a Shell node is used.

i. Choose **General > Shell** and drag Shell to the canvas on the right.



ii. In the **Create Node** dialog box, enter a name in the **Node Name** field.

> **Notice** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

iii. Click **Commit**.

iv. On the canvas of the do-while node, drag lines to configure the Shell node as the descendant node of the start node and the ancestor node of the end node.

v. Double-click the Shell node. The configuration tab of the Shell node appears.

vi. Enter the following code in the code editor:

```
echo ${dag.loopTimes} ----Display the current number of loops.
```

- The ${dag.loopTimes} variable is a reserved variable of the system. This variable specifies the current number of loops, and the value of this variable starts from 1. All inner nodes of the do-while node can reference this variable. For more information about built-in variables, see Built-in variables and Examples of variable values.
- After you modify the code of the Shell node, save the modification. No message that reminds you to save the modification will appear when you commit the node. If you do not save the modification, the code cannot be updated to the latest version in time.

4. Configure the end node to control the number of loops that can be run.

i. Double-click the end node. The configuration tab of the node appears.

ii. Select **Python** from the **Language** drop-down list.

iii. Enter the following code to define the condition for exiting from looping for the do-while node:

```
if ${dag.loopTimes}<5:
 print True;
else:
 print False;
```

- The ${dag.loopTimes} variable is a reserved variable of the system. This variable specifies the current number of loops, and the value of this variable starts from 1. All inner nodes of the do-while node can reference this variable. For more information about built-in variables, see Built-in variables and Examples of variable values.
- In the code, the value of the `dag.loopTimes` variable is compared with 5 to limit the number of loops that can be run. The value of the ${dag.loopTimes} variable is 1 for the first loop and increases by 1 each time. In this case, the value of the ${dag.loopTimes} variable is 2 for the second loop and 5 for the fifth loop. The do-while node exits from looping when the result of ${dag.loopTimes}<5 is False.

5. On the configuration tab of the do-while node, click the **Properties** tab on the right-side navigation pane to configure scheduling properties for the node. For more information, see Configure basic properties.
6. Click the icon in the top toolbar.
7. Commit the do-while node.

> **Notice** You can commit the do-while node only after you configure the **Rerun** and **Parent Nodes** parameters.

i. Click the icon in the top toolbar.

ii. In the **Commit** dialog box, select the nodes that you want to commit and enter your comments in the **Description** field.

iii. Click **Commit**.

If the workspace that you use is in standard mode, you must click **Deploy** in the upper-right corner to deploy the do-while node after you commit it. For more information, see Deploy nodes.

8. Test the node and view the result.

> **Note** If the workspace that you use is in standard mode, you cannot directly perform a test to run a do-while node in DataStudio.
>
> To perform a test to run the do-while node and view the result, you must commit the do-while node to Operation Center and run the do-while node in Operation Center. If you use the value passed by an assignment node in the do-while node, run both the assignment node and do-while node during the test in Operation Center.

i. On the node configuration tab, click **Operation Center** in the upper-right corner to go to **Operation Center**.

ii. In the left-side navigation pane of the Operation Center page, choose **Cycle Task Maintenance > Cycle Task**.

iii. On the Cycle Task page, find the do-while node and click DAG in the Actions column to open the directed acyclic graph (DAG) of the do-while node. In the DAG of the do-while node, right-click the assignment node and choose **Run > Current and Descendent Nodes Retroactively**. In the Patch Data dialog box, configure the parameters and click OK.

iv. Refresh the **Patch Data** page. After the data backfill instance is run, click **DAG** in the Actions column of the instance.

v. Right-click the do-while node and select **View Internal Nodes**.

The internal workflow of the do-while node is divided into three parts:

- The left pane of the view displays the rerun history of the do-while node. A record is generated each time a do-while node instance is run.
- The middle pane of the view displays a loop record list that shows all existing loops of the do-while node and the status of each loop.
- The right pane of the view displays the details about each loop. You can click a record in the loop record list to view the details of each instance in the loop.

vi. On the inner node page, click **Loop 3** on the left, right-click the Shell node, and then select **View Runtime Log**.

The preceding example shows that a do-while node works based on the following application logic:

i. The system starts a loop from the start node.

ii. Other nodes inside the do-while node run in sequence based on the dependencies configured for them.

iii. The system executes the conditional statement defined in the code of the end node for exiting from looping.

iv. The system records the number of loops that are run, and the next loop starts if the conditional statement returns True in the logs of the end node.

v. The entire looping process ends if the conditional statement returns False in the logs of the end node.

Complex example of using a do-while node

In addition to the preceding simple scenario, you may encounter complex scenarios in which each data entry is processed in sequence by using a loop. You can use a do-while node to process data in these scenarios. Before you use a do-while node to process data in these scenarios, make sure that the following conditions are met:

- Another node is deployed and configured as the ancestor node of the do-while node. The node can pass its output to the do-while node. You can use an assignment node as the ancestor node.
- The output of the assignment node is configured as the input of the do-while node. This way, the do-while node can obtain the output of the assignment node.
- The inner nodes of the do-while node can reference each data entry. The built-in variable ${dag.offset} is used to reference the input parameters configured for the do-while node.

The following example shows how to configure a do-while node in a complex scenario. The preceding figure shows the following information:

- The output of the assignment node is a two-dimensional array. The two-dimensional array is passed to the do-while node.

Sample values of the two-dimensional array:

```
+-----------------------------------------------+
| uid             | region | age_range | zodiac |
+-----------------------------------------------+
| 0016359810821   | Hubei Province | 30 to 40 years old   | Cancer |
| 0016359814159   | Unknown    | 30 to 40 years old   | Cancer |
+-----------------------------------------------+
```

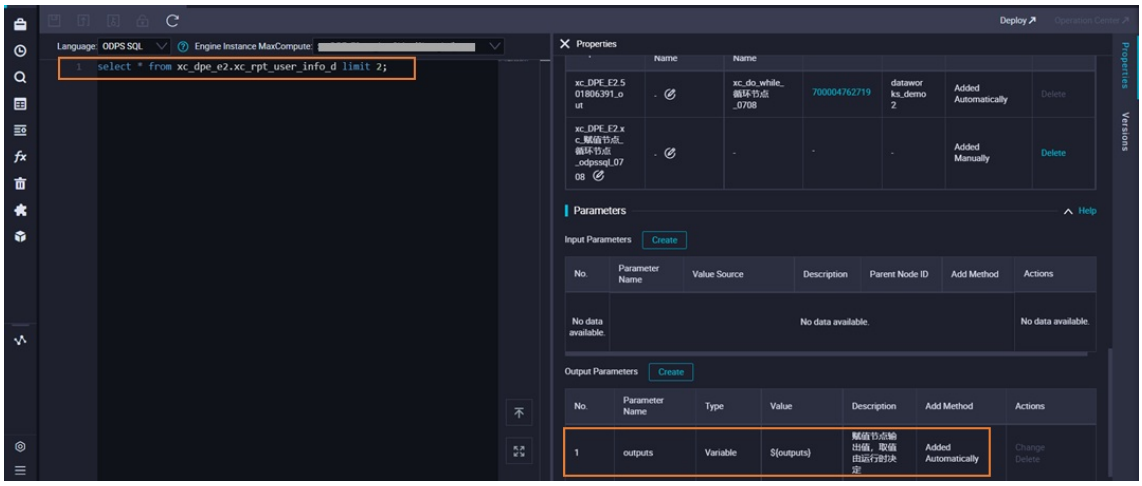
- The inner nodes of the do-while node use variables to obtain and print the loop parameters, offsets, and parameter values of the input from the ancestor assignment node.

1. Create and configure an assignment node.

   Key points:

   - Value assignment code and input and output parameters: Select the language of the assignment node and write the code of the assignment parameter. The system generates output parameters for the output of the assignment node based on specific rules.

     > **Note** The output of the assignment node is used as the input of the do-while node.

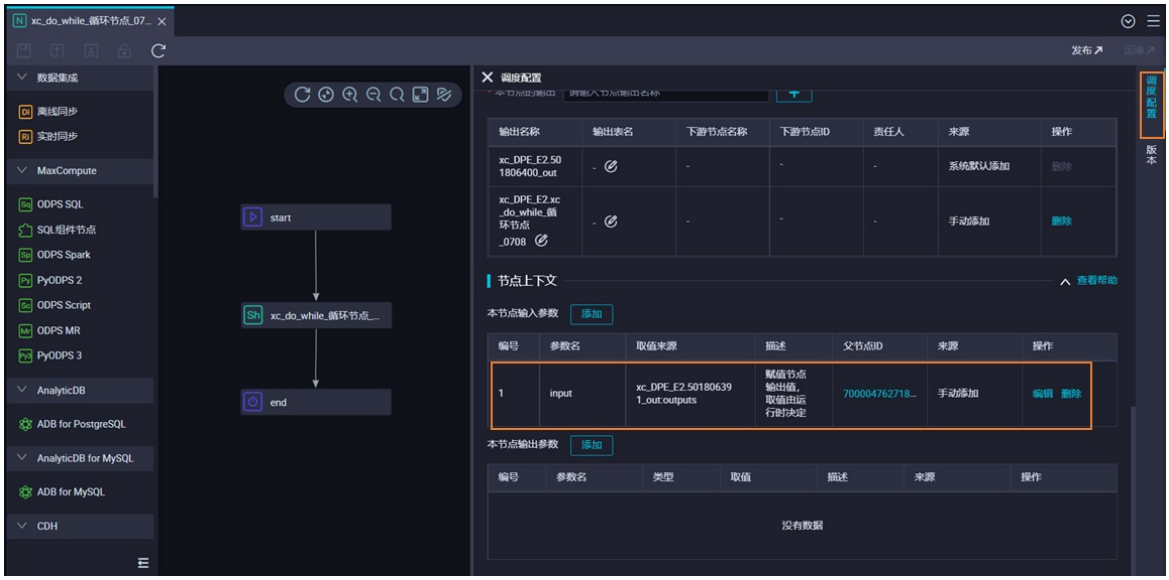
   - Node dependencies: You can create an assignment node in the workflow and drag a line to configure the assignment node as the ancestor node of the do-while node.

   For more information, see Configure an assignment node.

2. Configure the output of the assignment node as the input of the do-while node.

   On the configuration tab of the do-while node, click the **Properties** tab on the right-side navigation pane. In the **Input and Output Parameters** section, click **Create**. Set the **Parameter Name** parameter to input and the **Value Source** parameter to the output parameter of the ancestor assignment node.

   > **Note** The input and output parameters are configured for the assignment node and the do-while node, not for the inner nodes of the do-while node.



3. Configure the inner loop task node of the do-while node.

   Double-click the name of the do-while node. The node configuration tab appears. Then, define the workflow inside the do-while node.

   By default, the do-while node consists of three nodes: start, sql, and end. In this example, you must delete the sql node, create a Shell node, and then write code for the Shell node to print the loop parameters. Take note of the following key points:

   - Node dependencies: After you delete the sql node and create a Shell node, you must drag lines to establish dependencies between the inner nodes.

   - Loop task code: When you write code for the Shell node, you can use built-in variables to print various loop parameters. For more information about the built-in variables available for a do-while node, see Built-in variables. You can refer to the following sample code to write code for the Shell node:

```
echo '${dag.input}';
echo 'Obtain the row data of the current loop:'${dag.input[${dag.offset}]};
echo 'Obtain the offset:'${dag.offset};
echo 'Obtain the number of loops that are run:'${dag.loopTimes};
echo 'Obtain the length of the dataset passed by the ancestor assignment node _odpssq
l:'${dag.input.length};
echo 'If you want to select data in a specific row and a specific column in the outpu
t of the assignment node, select the value based on a two-dimensional array:'${dag.in
put[0][1]};
```

4. Define the loop exit condition for the end node.

   You can use the built-in variables supported by the do-while node for loop control. In this example, the values of the dag.loopTimes and dag.input.length variables are compared. The dag.loopTimes variable specifies the number of loops that are run, and the dag.input.length variable specifies the length of the dataset passed by the ancestor assignment node. If the value of the dag.loopTimes variable is less than the value of the dag.input.length variable, the end node returns True and the next loop starts. Otherwise, the end node returns False, and the entire looping process ends. In this example, the following code is used:

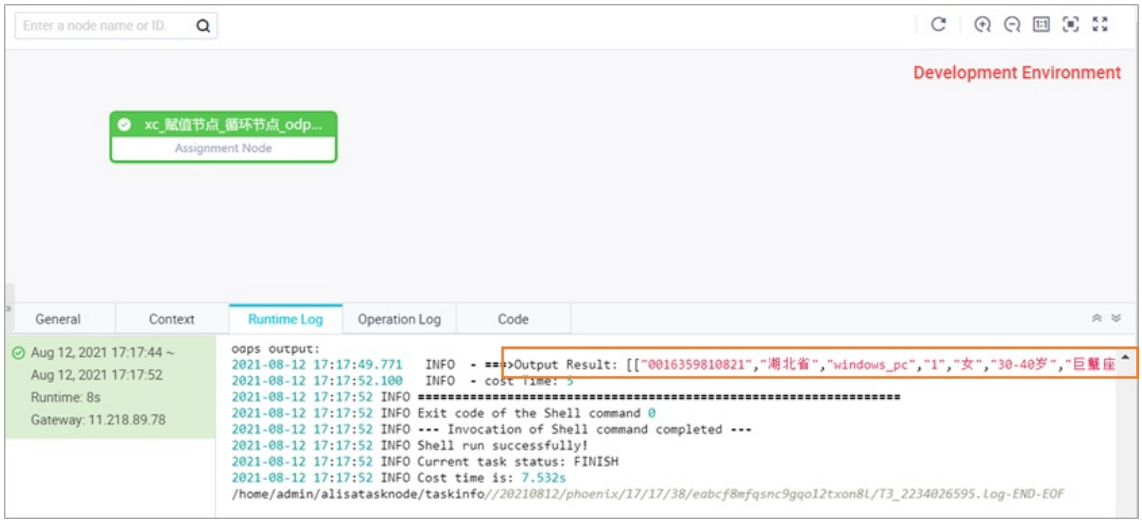
   ```
   if ${dag.loopTimes}<${dag.input.length}:
       print True;
   else:
       print False;
   ```

5. Run the do-while node and view the result.

   Go to Operation Center, find the do-while node, and then open the DAG of the node. In the DAG, right-click the node name and choose **Run > Current and Descendent Nodes Retroactively**. In the Nodes section of the Patch Data dialog box, select the assignment node and the do-while node. After the data backfill instances are run, you can view the result in the logs.

   > **Note**
   > - If you use the value passed by an assignment node in the do-while node, run both the assignment node and do-while node during the test in Operation Center.
   > - To view the operational logs of a do-while node in Operation Center, perform the following steps: find the do-while node and open the DAG of the node. In the DAG, right-click the node name and select **View Internal Nodes** to view the operational logs of the inner nodes.

   - View the output of the assignment node.



- View the result that is returned after the end node is run for the first time
- View the result that is returned after the end node is run for the second time

Summary

- Comparison between a do-while node and the while, For Each, and do-while loop statements:
  - A do-while node runs based on a workflow that starts a loop before evaluation. This node functions the same way as the do-while statement. A do-while node can use the built-in variable ${dag.offset} and input and output parameters to achieve the feature of the For Each statement.
  - A do-while node cannot achieve the feature of the while statement because a do-while node runs a loop before evaluation.
- Work procedure of a do-while node:
  i. The system runs a loop from the start node and runs other nodes based on the dependencies configured for them.
  ii. After the system runs the code that is defined for the end node in a loop, one of the following situations occur:
    - The next loop starts if the end node returns True.
    - The entire looping process ends if the end node returns False.
- Input and output parameters: The inner nodes of the do-while node use a variable ${dag.Input and output parameter names} to reference the input and output parameters configured for the do-while node.
- Built-in variables: DataWorks provides the following built-in variables for the inner nodes of the do-while node:
  - dag.loopTimes: the number of loops that are run. The value of this variable starts from 1.
  - dag.offset: the offset of the number of loops that are run to 1. The value of this variable starts from 0.

5.11.4. Configure a merge node

This topic describes the definition of merge nodes and how to create a merge node and define the merging logic. This topic also provides a sample merge node to show how to configure and run a merge node.

Context

A merge node is a logical control node in DataStudio. A merge node can merge the results of its ancestor nodes. A merge node aims to facilitate the scheduling of nodes that depend on the output of the child nodes of a branch node.

You cannot change the status of a merge node. A merge node merges the results of multiple child nodes of a branch node and sets the status to Successful. To ensure the proper scheduling of a node that depends on the output of the child nodes of a branch node, you can configure the node to depend on a merge node.

For example, Branch Node C has two logically exclusive branches C1 and C2. These two branches use different logic to write data to the same MaxCompute table. Assume that Node B depends on the output of this MaxCompute table. To ensure that Node B can be run as expected, you must use Merge Node J to merge the results of branches C1 and C2, and then configure Merge Node J as the parent node of Node B. If Node B directly depends on branches C1 and C2, one of the branches will fail to be run because only one branch meets the branch condition each time Branch Node C is run. In this case, Node B and its descendant nodes cannot be triggered as scheduled.

Limits

DataWorks Standard Edition or a more advanced edition is activated so that you can use merge nodes.

Create a merge node

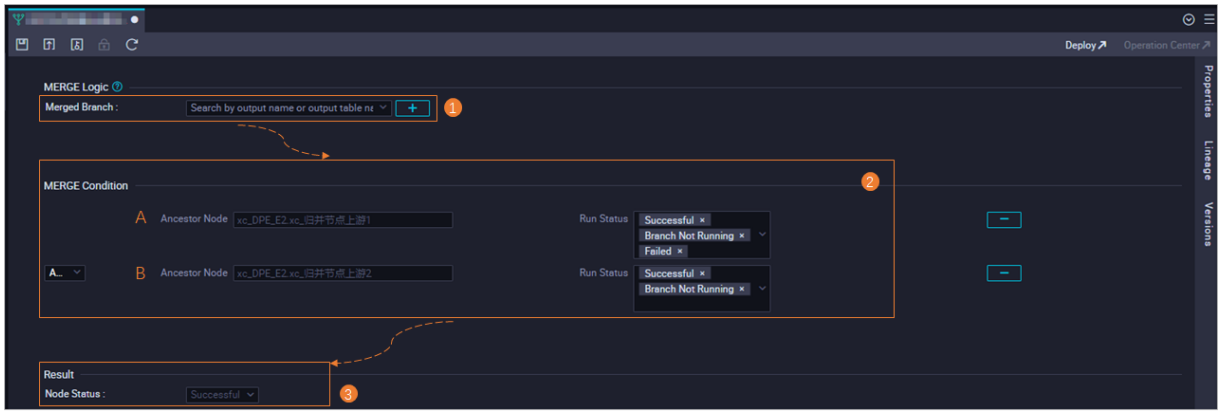
1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Move the pointer over the **+Create** icon and choose **General > Merge Node**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.

Define the merging logic

After the merge node is created, the node configuration tab appears. Define the merging logic for the node.



1. Add the branch node whose branches need to be merged. The branch node becomes an ancestor node of the merge node.

   In the **Merged Branch** drop-down list, enter the output name or output table name of the branch node, select the branch node, and then click the + icon.

   > **Note** If you need to merge branches of multiple branch nodes, you must repeat this step to add the branch nodes one by one.

2. In the **MERGE Condition** section, configure merge conditions for the branch nodes.

   You need to configure merge logic conditions and states for the branch nodes.

   - The following merge logic conditions are supported:
     - **AND**: The node status specified in the **Result** section takes effect only if all the ancestor branch nodes are run and in the specified state.
     - **OR**: The node status specified in the **Result** section takes effect if an ancestor branch node is run and in the specified state.
   - You can specify the following states for the branch nodes:
     - **Successful**
     - **Failed**
     - **Branch Not Running**

3. In the **Result** section, specify the status of the merge node.

   > **Note** You can set the status of the merge node only to **Successful**.

The preceding figure shows a merge node with the following configurations:

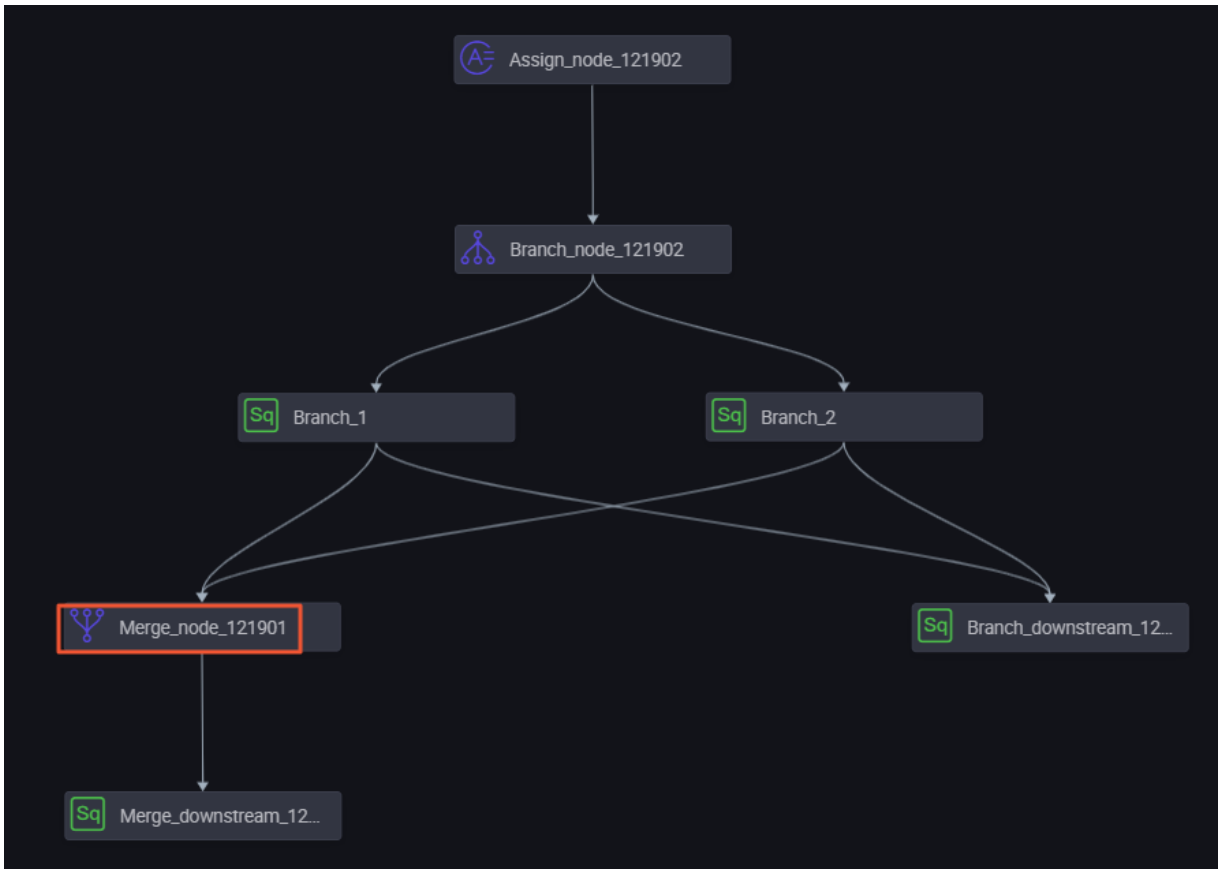
- Branch Nodes A and B are added as the ancestor nodes of the merge node.
- The **Successful**, **Branch Not Running**, and **Failed** states are specified for Node A. In this case, Node A needs only to be run, regardless of the result.
- The **Successful** and **Branch Not Running** states are specified for Node B. In this case, Node B needs to be run and the result must not be Failed.
- The merge logic condition is set to **AND**.

Therefore, the **Successful** status of the merged node takes effect if Nodes A and B are run and the result of Node B is not Failed.

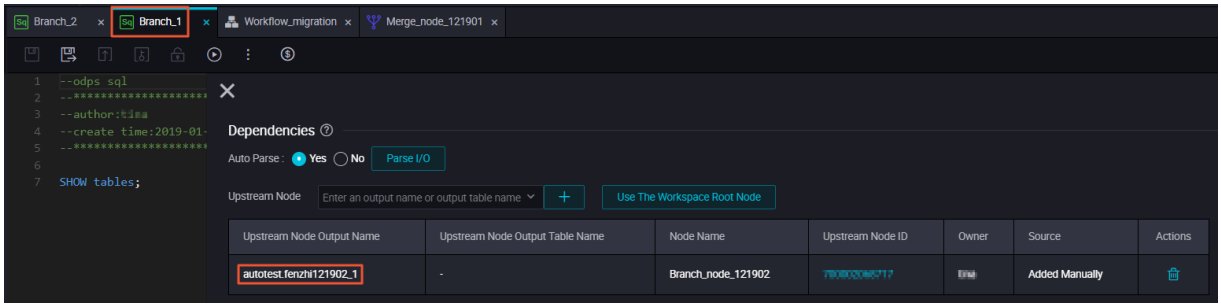
On the node configuration tab, click the **Properties** tab in the right-side navigation pane. Configure the scheduling properties of the merge node. For more information, see Configure basic properties.

Sample merge node

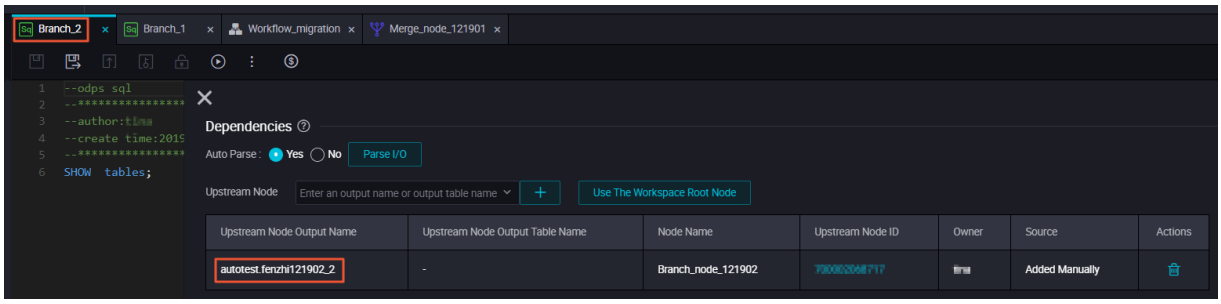
You can associate child nodes with different outputs of a branch node to define the branches under different conditions. For example, in the workflow that is shown in the following figure, the branches **Branch_1** and **Branch_2** are defined as the child nodes of the branch node.



Branch_1 depends on the output that is named autotest.fenzhi121902_1.



Branch_2 depends on the output that is named autotest.fenzhi121902_2.



Run nodes

The condition of Branch_1 is met. The child node of this branch is run. You can select the branch and view the running details of the child node on the **Runtime Log** tab.

The condition of Branch_2 is not met. The child node of this branch is skipped. You can select the branch and view relevant information on the **Runtime Log** tab.

The child node of the merge node is run as expected.

5.11.5. Configure a branch node

A branch node is a logical control node in DataStudio. It can define the branch logic and the direction of branches under different logical conditions.

Prerequisites

- DataWorks Standard Edition or a more advanced edition is activated. Then, you can use branch nodes.
- Generally, branch nodes need to be used together with assignment nodes. For more information, see Configure an assignment node.

Create a branch node

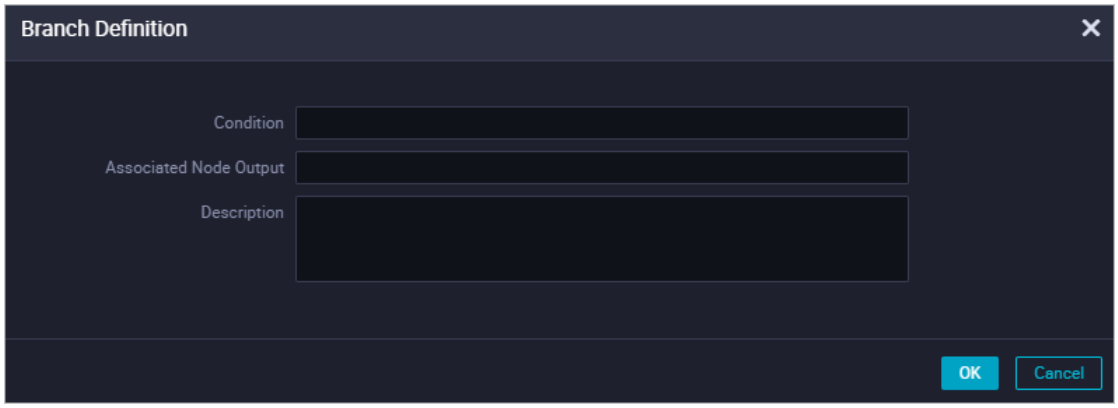
1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Move the pointer over the **+Create** icon and choose **General > Branch Node**.

   Alternatively, find the required workflow, right-click **General**, and then choose **Create > Branch Node**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. Define the branch logic.

i. In the **Definition** section, click **Add Branch**.

ii. In the **Branch Definition** dialog box, set the parameters.



Parameter	Description
Condition	The following information describes the condition of the branch: ■ You can use only Python comparison operators to define logical conditions for the branch node. ■ If the result of the expression is *true* when the node is running, the corresponding branch condition is met. ■ If the expression fails to be parsed when the node is running, the whole branch node fails. ■ To define branch conditions, you can use global variables and the parameters that are defined in the node context. For example, the ${input} variable can be used as an input parameter of the branch node.
Associated Node Output	The following information describes the associated node output: ■ The node output is used to configure dependencies between the child nodes and the branch node. ■ If the branch condition is met, the child node that depends on the node output is run. If the child node also depends on the output of other nodes, the status of these nodes also matters. ■ If the branch condition is not met, the child node that depends on the node output is not run. The child node is set to the `Not Running` state because the branch condition is not met.
Description	The description of the branch. For example, the branches ${input}==1 and ${input}>2 are defined.

The following example shows associated node outputs.

Assume that a branch node is associated with two child nodes. The names of the child nodes are qqqq and wwww. If the condition of Branch 1 is met, the qqqq node is run. If the condition of Branch 2 is met, the wwww node is run. When you configure the branch node, you can configure associated node outputs as needed. Assume that the associated node output of Branch 1 is 1234, and the associated node output of Branch 2 is 2324. Both 1234 and 2324 are used as the output names of the branch node. The qqqq node must be associated with the branch node based on the output name 1234. The wwww node must be associated with the branch node based on the output name 2324.

iii. Click **OK**.

After you add a branch, you can click **Change** or **Delete**.

- Click **Change** to modify the branch and related dependencies.
- Click **Delete** to delete the branch and related dependencies.

6. Click the **Properties** tab in the right-side navigation pane and set the scheduling properties for the node.

After the branch conditions are defined, the output names are automatically added to the output name list in the **Dependencies** section of the **Properties** tab. Then, you can associate child nodes with the branch node based on the output names.

> **Note**
> - Child nodes inherit dry-run properties of the parent node. Therefore, we recommend that you do not create a node that depends on its last-cycle instance as the branch.
> - The dependencies that are established by drawing lines between nodes are not recorded on the Properties tab. You must manually specify these dependencies.

7. Commit the node.

> **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

i. Click the icon in the toolbar.

ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

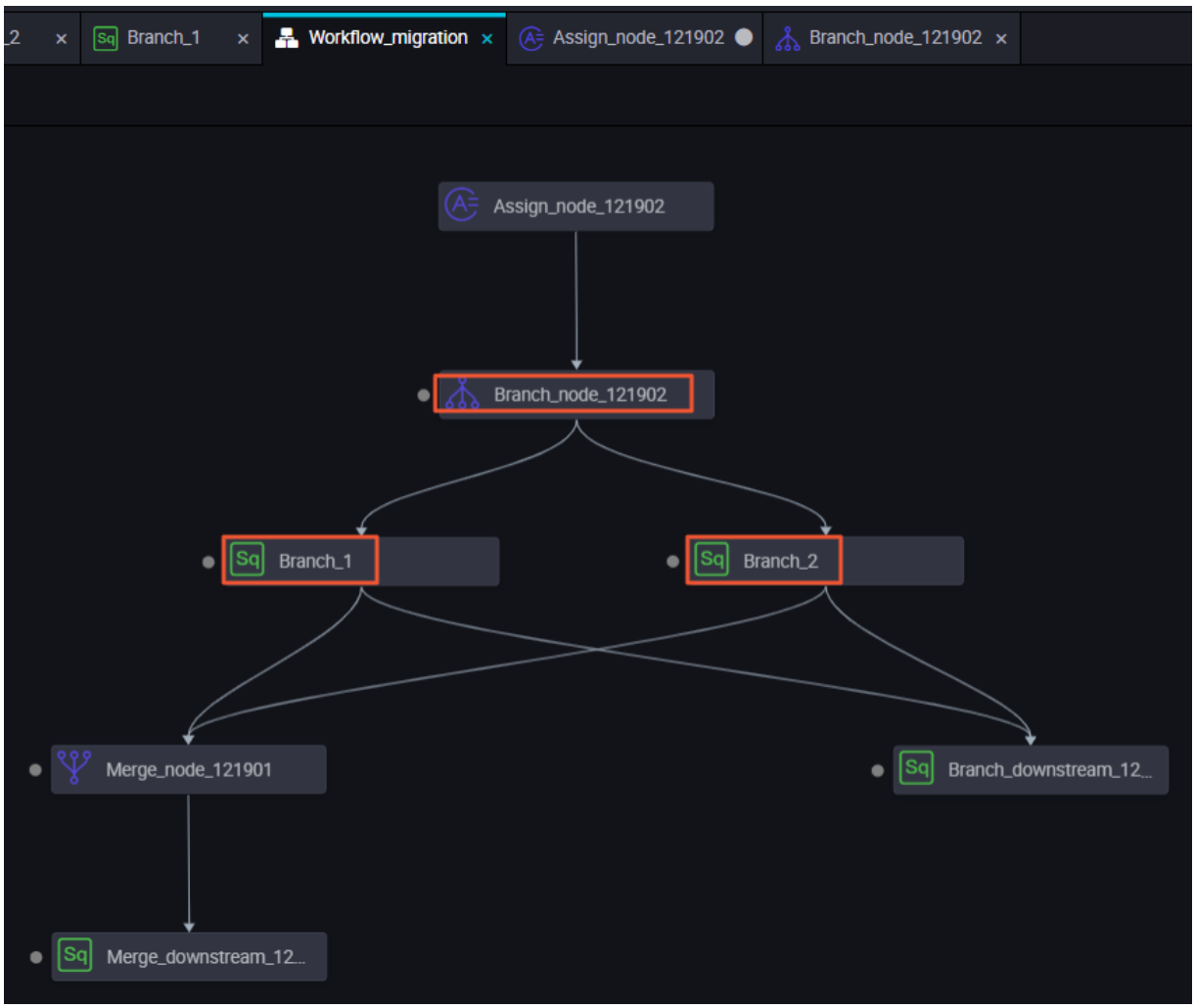
iii. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the do-while node.

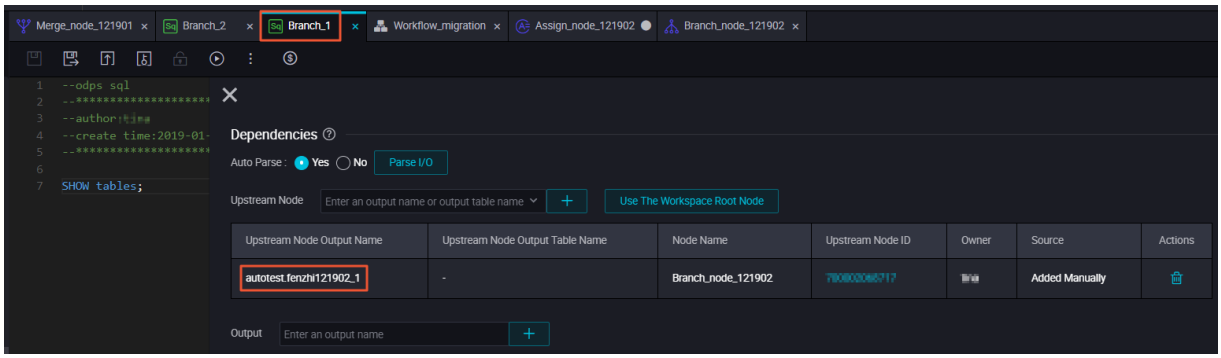
8. Test the node. For more information, see View auto triggered nodes.

Example: Configure child nodes for a branch node

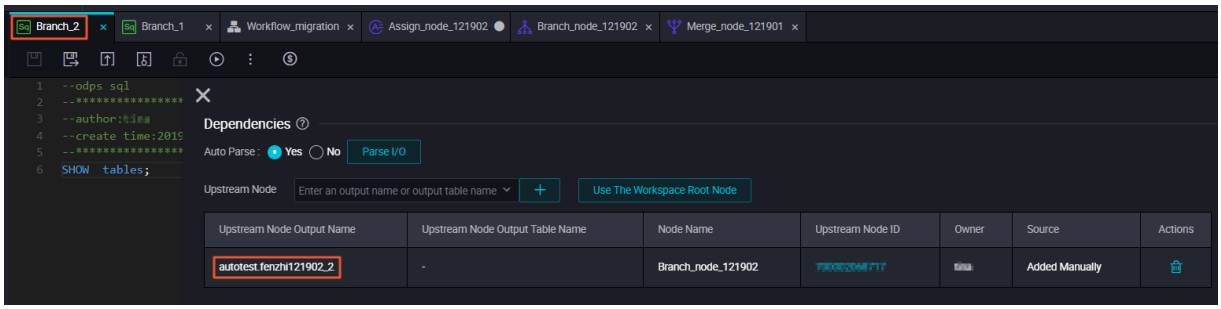
You can associate child nodes with different outputs of a branch node to define the branches under different conditions. For example, in the workflow that is shown in the following figure, the branches **Branch_1** and **Branch_2** are defined as the child nodes of the branch node.



Branch_1 depends on the output that is named autotest.fenzhi121902_1.



Branch_2 depends on the output that is named autotest.fenzhi121902_2.



Commit the branch node and run it in Operation Center. In this example, the condition of Branch_1 is met. Branch_1 depends on the autotest.fenzhi121902_1 output of the branch node. The following figures show the operational logs.

- The condition of Branch_1 is met. The child node of this branch is run. You can select the branch and view the running details of the child node on the **Runtime Log** tab.
- The condition of Branch_2 is not met. The child node of this branch is skipped. You can select the branch and view relevant information on the **Runtime Log** tab.

Supported Python comparison operators

Assume that the value of Variable a is 10 and that of Variable b is 20 in the following table.

Operator	Description	Example
==	Equal: checks whether two objects are equal.	(a==b) returns false.
!=	Not equal: checks whether two objects are not equal.	(a!=b) returns true.
<>	Not equal: checks whether two objects are not equal.	(a<>b) returns true. This operator is similar to !=.
>	Greater than: checks whether x is greater than y.	(a>b) returns false.
<	Less than: checks whether x is less than y. If the returned result is 1 or 0, 1 indicates true and 0 indicates false. The results 1 and 0 are equivalent to the special variables true and false.	(a<b) returns true.
>=	Greater than or equal to: checks whether x is greater than or equal to y.	(a>=b) returns false.
<=	Less than or equal to: checks whether x is less than or equal to y.	(a<=b) returns true.

5.11.6. Configure an assignment node

If you want a node to use the data of its ancestor node, you can use an assignment node to pass the data. Assignment nodes support the Shell, ODPS SQL, and Python 2 languages and automatically add the outputs parameter based on value assignment rules. This helps nodes reference the data of their ancestor nodes. This topic describes how to use an assignment node.

Prerequisites

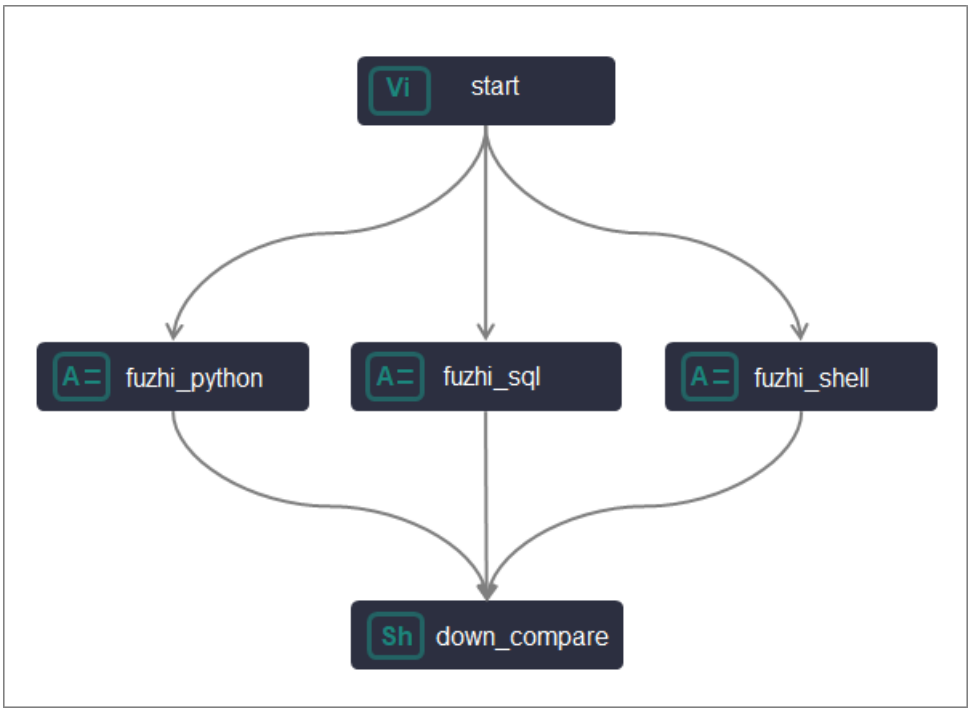
DataWorks of the Standard Edition or a more advanced edition is activated.

Only DataWorks of the Standard Edition or a more advanced edition supports assignment nodes.

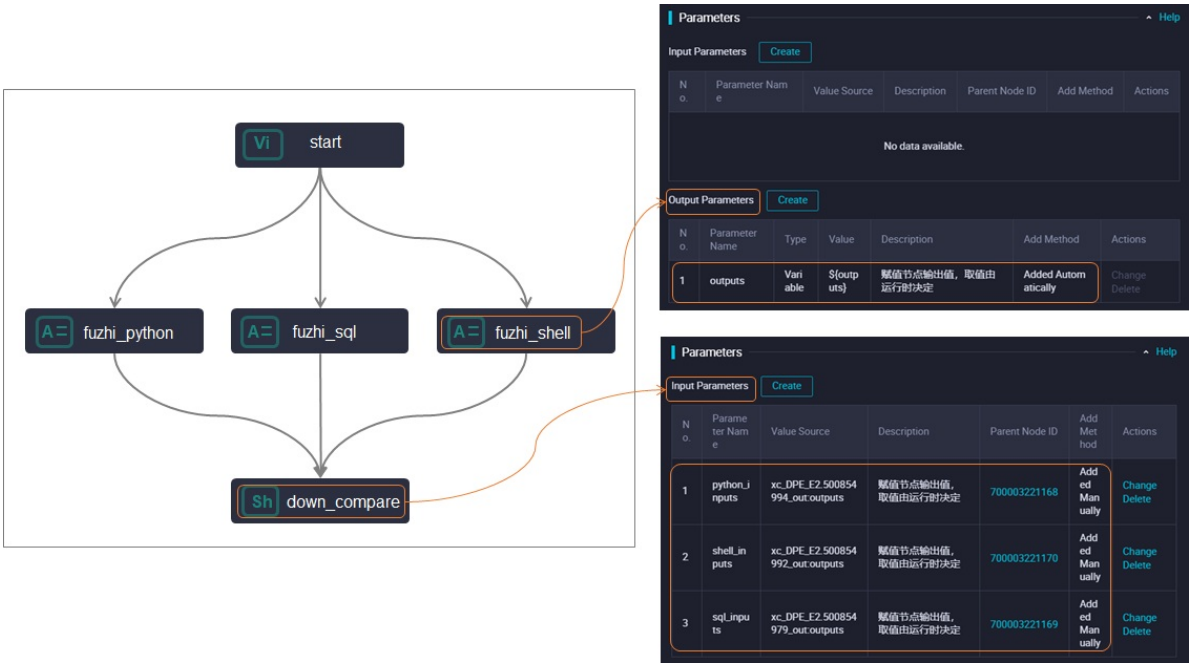
Context

When you use an assignment node to transparently pass the data that is assigned to the outputs parameter, take note of the following items:

- Dependencies between the assignment node and its ancestor and descendant nodes



  In the preceding figure, three assignment nodes are created: fuzhi_python, fuzhi_sql, and fuzhi_shell. Before you use these assignment nodes, you must perform the following operations:
  - Configure the start node as the ancestor node of the assignment nodes and the down_compare node as the descendant node of the assignment nodes to establish dependencies among all these nodes. The down_compare node references the outputs parameters of the assignment nodes.
  - Before you configure the assignment nodes as the ancestor nodes of down_compare, commit the assignment nodes. This ensures that the outputs parameters of the assignment nodes can be parsed when you configure the down_compare node.
- Data passing relationships between the assignment node and its ancestor and descendant nodes



After you configure Output Parameters for the assignment nodes and Input Parameters for down_compare, data passing and parameter reference relationships are established, as shown in the preceding figure.

- The outputs parameter that needs to be referenced by down_compare must be added to **Output Parameters** in the **Parameters** section of the Properties tab for each assignment node.
- The outputs parameter that needs to be referenced by down_compare must be added to **Input Parameters** in the **Parameters** section of the Properties tab for down_compare.

> **Note**
>
> - For some nodes created in DataStudio, you do not need to configure assignment nodes if you want to transparently pass data between these nodes. You can manually add the **outputs** parameter to Output Parameters or Input Parameters for the nodes. The outputs parameter functions the same way as an assignment node. For example, you can manually add the outputs parameter to Output Parameters or Input Parameters only for the EMR Hive, EMR Spark SQL, ODPS Script, Hologres SQL, AnalyticDB for PostgreSQL, and MySQL nodes. For more information about how to add the outputs parameter, see Configure input and output parameters.
> - If you want to transparently pass data between nodes for which you cannot manually add the outputs parameter to Output Parameters or Input Parameters, you must use assignment nodes.
> - Assignment nodes can transparently pass data only to their Level-1 child nodes.

- Data passing verification (If the descendant nodes of the assignment node need to reference the data that is passed to the assignment node, and the descendant nodes and the assignment node need to be run at the same time, you can run all these nodes on their configuration tabs or in the Operation Center to check whether the assignment node can pass the data to the descendant nodes.)
- Relationships between the parameter output format of the assignment node and the way the descendant nodes of the assignment node reference the outputs parameter

The following table describes the value assignment rules of the outputs parameter in the assignment nodes that use different languages.

Language	Value of outputs	Output format of outputs	Size limit on the value of outputs
ODPS SQL	The data in the output of the SELECT statement in the last row is used as the value of the outputs parameter. This outputs parameter is added to Output Parameters for the assignment node.	The data is passed to the descendant nodes of the assignment node as a two-dimensional array.	The value size of the outputs parameter cannot exceed 2 MB. If the value size exceeds 2 MB, the assignment node fails to run.
Shell	The data in the output of the ECHO statement in the last row is used as the value of the outputs parameter. This outputs parameter is added to Output Parameters for the assignment node.	The data is passed to the descendant nodes of the assignment node as a one-dimensional array whose elements are separated by commas (,).	
Python 2	The data in the output of the PRINT statement in the last row is used as the value of the outputs parameter. This outputs parameter is added to Output Parameters for the assignment node.	The data is passed to the descendant nodes of the assignment node as a one-dimensional array whose elements are separated by commas (,).	

Limits

Only DataWorks of the Standard Edition or a more advanced edition supports assignment nodes.

Procedure

This topic describes how to use assignment nodes in the Python 2, ODPS SQL, and Shell languages to pass data to down_compare. In this example, the data in the output of the last row of the code for each assignment node is passed to the descendant node of the assignment node by configuring Input Parameters and Output Parameters. To pass the data, you must perform the following steps:

1. Create assignment nodes and other nodes.
2. Configure dependencies for the created nodes.
3. Configure Output Parameters for fuzhi_sql and Input Parameters for down_compare.
4. Configure Output Parameters for fuzhi_python and Input Parameters for down_compare.
5. Configure Output Parameters for fuzhi_shell and Input Parameters for down_compare.

Create assignment nodes and other nodes

In this example, three assignment nodes that use different languages (Python 2, ODPS SQL, and Shell) are created: fuzhi_python, fuzhi_sql, and fuzhi_shell.

1. Log on to the DataWorks console.
2. In the left-side navigation pane, click **Workspaces**.
3. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **Data Development** in the Actions column.
4. Move the pointer over the +Create icon and choose **General > Assignment Node**.

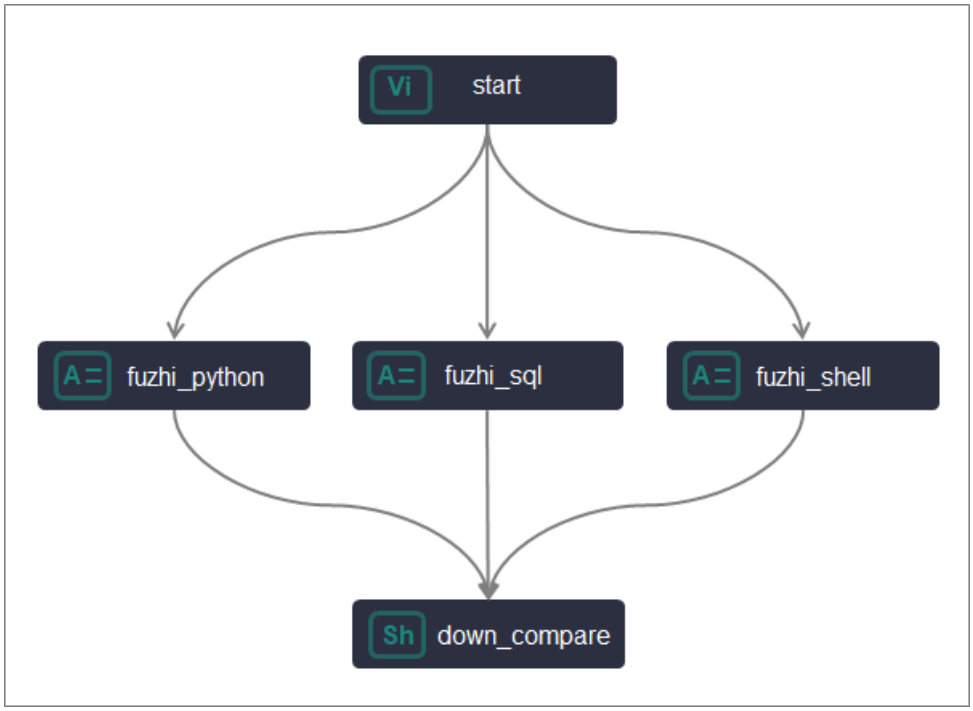
   Alternatively, find the desired workflow in the Scheduled Workflow pane, click the workflow name, right-click **General**, and then choose **Create > Assignment Node**.
5. In the **Create Node** dialog box, configure the **Node Name** and **Location** parameters.

   > **Notice** The node name must be a maximum of 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
6. Click **Commit**.

After you create the assignment nodes, you need to create other nodes based on your business requirements. In this example, you need to create a node named start and a node named down_compare. start is a zero load node, and down_compare is a Shell node. For more information about how to create these two nodes, see Create a zero-load node and Create a Shell node.

Configure dependencies for the created nodes

After you create the assignment nodes and other nodes, you must configure dependencies for these nodes based on your business requirements.



In this example, you can draw lines to connect start to all the three assignment nodes to use start as the ancestor node of the assignment nodes. Similarly, you can draw lines to connect all the three assignment nodes to down_compare to use down_compare as the descendant node of the assignment nodes. For more information, see Configuration by drawing lines to connect nodes.

In addition, you can configure basic properties, time properties, and resource properties for all the nodes based on your business requirements. For more information, see Configure basic properties, Configure time properties, and Configure a resource group.

Configure Output Parameters for fuzhi_sql and Input Parameters for down_compare

This section describes how to configure Output Parameters for fuzhi_sql and Input Parameters for down_compare.

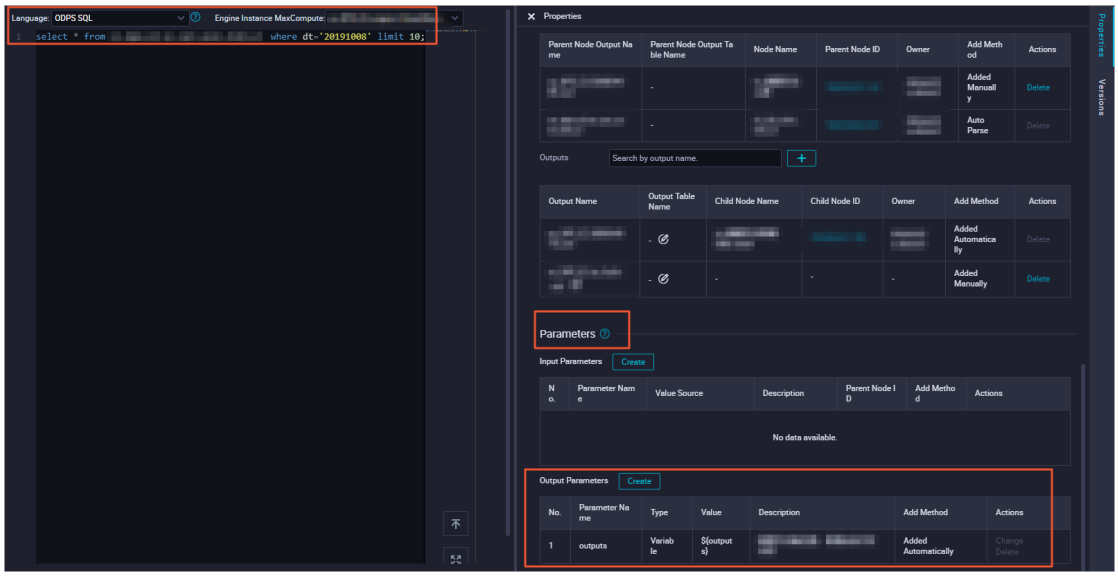
1. Configure fuzhi_sql.
   i. In the desired workflow, find fuzhi_sql and double-click its name.
   ii. On the configuration tab of fuzhi_sql, select ODPS SQL for Language and write value assignment code.

   Sample code:

   ```
   select * from xc_dpe_e2.xc_rpt_user_info_d  where dt='20191008' limit 10;
   ```

iii. Click the **Properties** tab in the right-side navigation pane. Then, configure **Output Parameters** in the **Parameters** section of the Properties tab.

fuzhi_sql assigns the data in the output of the code to the outputs parameter.



2. Configure down_compare.

i. In the desired workflow, find down_compare and double-click its name.

ii. On the configuration tab of down_compare, write code.

Sample code:

```
echo '${sql_inputs}';
echo 'Use the data in the first row in the output of fuzhi_sql as the input'${sql_inputs[0]};
echo 'Use the data in the second row in the output of fuzhi_sql as the input'${sql_inputs[1]};
echo 'Use the value of the second field in the first row in the output of fuzhi_sql as the input'${sql_inputs[0][1]};
echo 'Use the value of the third field in the second row in the output of fuzhi_sql as the input'${sql_inputs[1][2]};
```

iii. Click the **Properties** tab in the right-side navigation pane. Then, configure **Input Parameters** in the **Parameters** section of the Properties tab.

Rename the outputs parameter of fuzhi_sql to sql_inputs and add sql_inputs to Input Parameters for down_compare.

3. Run the code and view the reference result.

i. Click the icon in the top toolbar.

ii. In the **Warning** message, click **Continue to Run**.

iii. View the reference result.

Configure Output Parameters for fuzhi_python and Input Parameters for down_compare

This section describes how to configure Output Parameters for fuzhi_python and Input Parameters for down_compare.

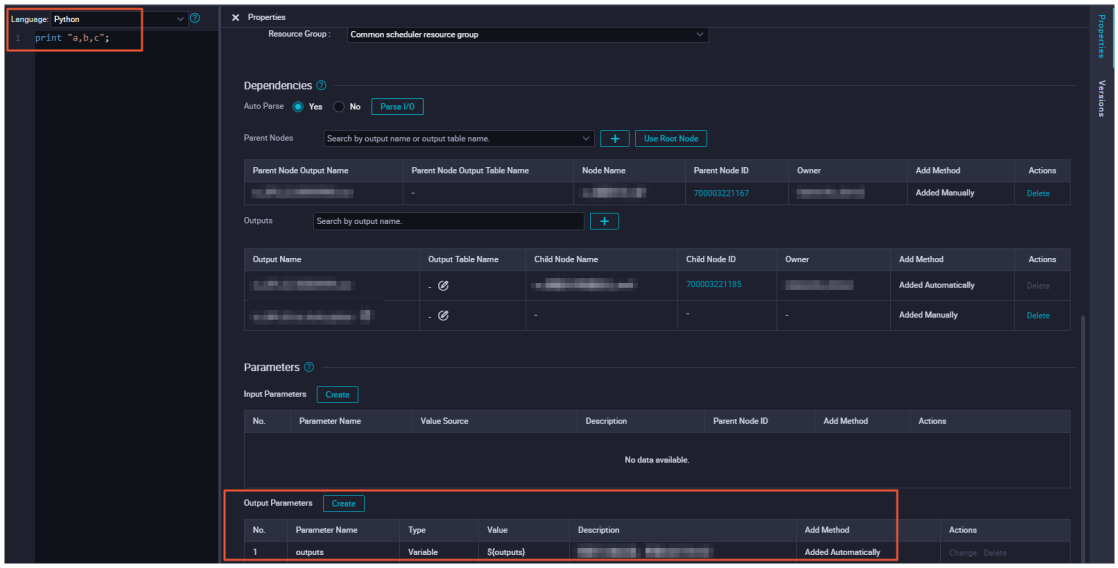
1. Configure fuzhi_python.
   i. In the desired workflow, find fuzhi_python and double-click its name.
   ii. On the configuration tab of fuzhi_python, select Python for Language and write value assignment code.

   Sample code:

   ```
   print "a,b,c";
   ```

   iii. Click the **Properties** tab in the right-side navigation pane. Then, configure **Output Parameters** in the **Parameters** section of the Properties tab.

   fuzhi_python assigns the data in the output of the code to the outputs parameter. In this example, the data that is assigned is a,b,c.

   The data a,b,c is assigned to the **outputs** parameter of fuzhi_python as a one-dimensional array.

2. Configure down_compare.
   i. In the desired workflow, find down_compare and double-click its name.
   ii. On the configuration tab of down_compare, write code.

   Sample code:

   ```
   echo 'The output of fuzhi_python'${python_inputs};
   echo 'Use the first value in the output of fuzhi_python as the input'${python_inputs[0]};
   echo 'Use the second value in the output of fuzhi_python as the input'${python_inputs[1]};
   ```

   iii. Click the **Properties** tab in the right-side navigation pane. Then, configure **Input Parameters** in the **Parameters** section of the Properties tab.

   Rename the outputs parameter of fuzhi_python to python_inputs and add python_inputs to Input Parameters for down_compare.

3. Run the code and view the reference result.
   i. Click the icon in the top toolbar.

ii. In the **Warning** message, click **Continue to Run**.

iii. View the reference result.

Configure Output Parameters for fuzhi_shell and Input Parameters for down_compare

This section describes how to configure Output Parameters for fuzhi_shell and Input Parameters for down_compare.

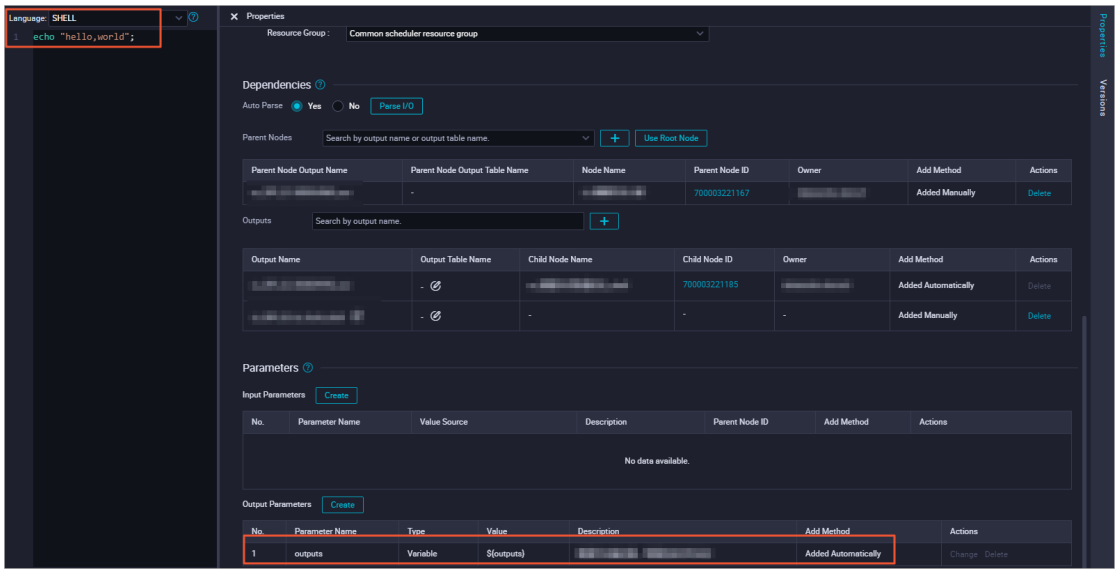
1. Configure fuzhi_shell.
   i. In the desired workflow, find fuzhi_shell and double-click its name.
   ii. On the configuration tab of fuzhi_shell, write value assignment code.

   Sample code:

   ```
   echo "hello,world";
   ```

   iii. Click the **Properties** tab in the right-side navigation pane. Then, configure **Output Parameters** in the **Parameters** section of the Properties tab.

   fuzhi_shell assigns the data in the output of the code to the outputs parameter. In this example, the data that is assigned is hello,world.

   The data hello,world is assigned to the **outputs** parameter of fuzhi_shell as a one-dimensional array.

2. Configure down_compare.
   i. In the desired workflow, find down_compare and double-click its name.
   ii. On the configuration tab of down_compare, write code.

   Sample code:

   ```
   echo 'The output of fuzhi_shell'${shell_inputs};
   echo 'Use the first value in the output of fuzhi_shell as the input'${shell_inputs[
   0]};
   echo 'Use the second value in the output of fuzhi_shell as the input'${shell_inputs
   [1]};
   ```

iii. Click the **Properties** tab in the right-side navigation pane. Then, configure **Input Parameters** in the **Parameters** section of the Properties tab.

Rename the outputs parameter of fuzhi_shell to shell_inputs and add shell_inputs to Input Parameters for down_compare.

3. Run the code and view the reference result.

i. Click the icon in the top toolbar.

ii. In the **Warning** message, click **Continue to Run**.

iii. View the reference result.

5.11.7. Create a Shell node

Shell nodes support standard shell syntax but not interactive syntax.

Context

You can run Shell nodes only on **exclusive resource groups for scheduling**. For more information, see Create and use an exclusive resource group for scheduling.

A Shell node that is run on an exclusive resource group for scheduling may need to access a data source that has a whitelist. In this case, you must add the information about the resource group to the whitelist of the data source. For more information, see Create and use an exclusive resource group for scheduling.

Procedure

1. Go to the **DataStudio** page.

i. Log on to the DataWorks console.

ii. In the left-side navigation pane, click **Workspaces**.

iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.

2. On the Scheduled Workflow tab, move the pointer over the Create icon and choose **General > Shell**.

Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > Shell**.

3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.

5. Configure the Shell node.

i. Write the code of the Shell node in the code editor.

To use the system scheduling parameters in the Shell node, execute the following statement:

```
echo "$1 $2 $3"
```

> **Note** Separate multiple parameters with spaces. Example: Parameter1 Parameter 2. For more information about the system scheduling parameters, see Overview of scheduling parameters.

i. Click the icon in the top toolbar to save the SQL statements.

ii. Click the icon in the toolbar to execute the SQL statements that you have saved.

If you want to use another resource group to test the Shell node on the **DataStudio** page, click the icon in the toolbar and select the exclusive resource group to use.

6. On the node configuration tab, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set the scheduling properties for the node. For more information, see Configure basic properties.
7. Save and commit the node.

> **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

i. Click the icon in the toolbar to save the node.

ii. Click the icon in the toolbar.

iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iv. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.

8. Test the node. For more information, see View auto triggered nodes.

5.11.8. Create a zero-load node

A zero-load node is a control node that supports dry-run scheduling only and does not generate data. It usually serves as the root node of a workflow.

Context

You can configure an output table for a zero-load node so that the output table can be used as an input table of another node. However, the zero-load node does not process the table data.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.

iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.

2. On the **Data Development** tab, move the pointer over the **+Create** icon and choose **Universal > Virtual node**.

   Alternatively, you can click a workflow in the Business process section, right-click **General**, and then choose **New > Virtual node**.

3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.

5. On the node configuration tab, click the **Scheduling configuration** tab in the right-side navigation pane. On the Scheduling configuration tab, set the scheduling properties for the node. For more information, see Configure basic properties.

6. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.

7. Test the node. For more information, see View auto triggered nodes.

5.11.9. Create an HTTP Trigger node

If you use an external scheduling system and want to trigger DataWorks nodes after nodes in the scheduling system are run, you can use an HTTP Trigger node of DataWorks to trigger the DataWorks nodes. This topic describes how to use an HTTP Trigger node of DataWorks if an external scheduling system is used to trigger DataWorks nodes. This topic also describes the precautions of using an HTTP Trigger node.

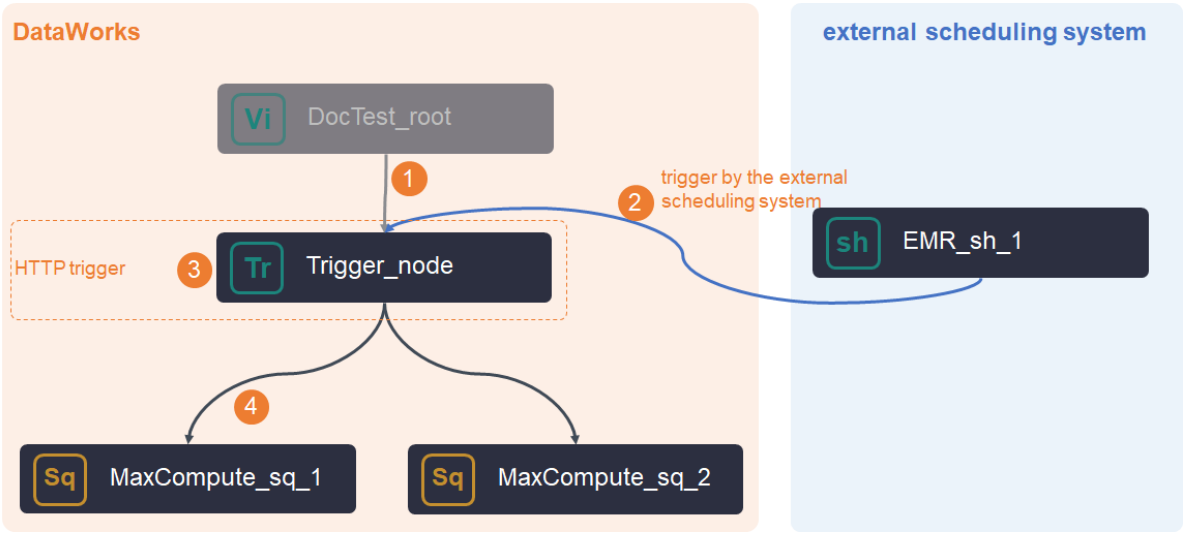
Prerequisites

- DataWorks Enterprise Edition or a more advanced edition is activated.
- A workflow is created. The compute nodes that need to be triggered by an HTTP Trigger node are created. In the example of this topic, ODPS SQL nodes are used as the compute nodes. For more information about how to create an ODPS SQL node, see Create an ODPS SQL node.

Context

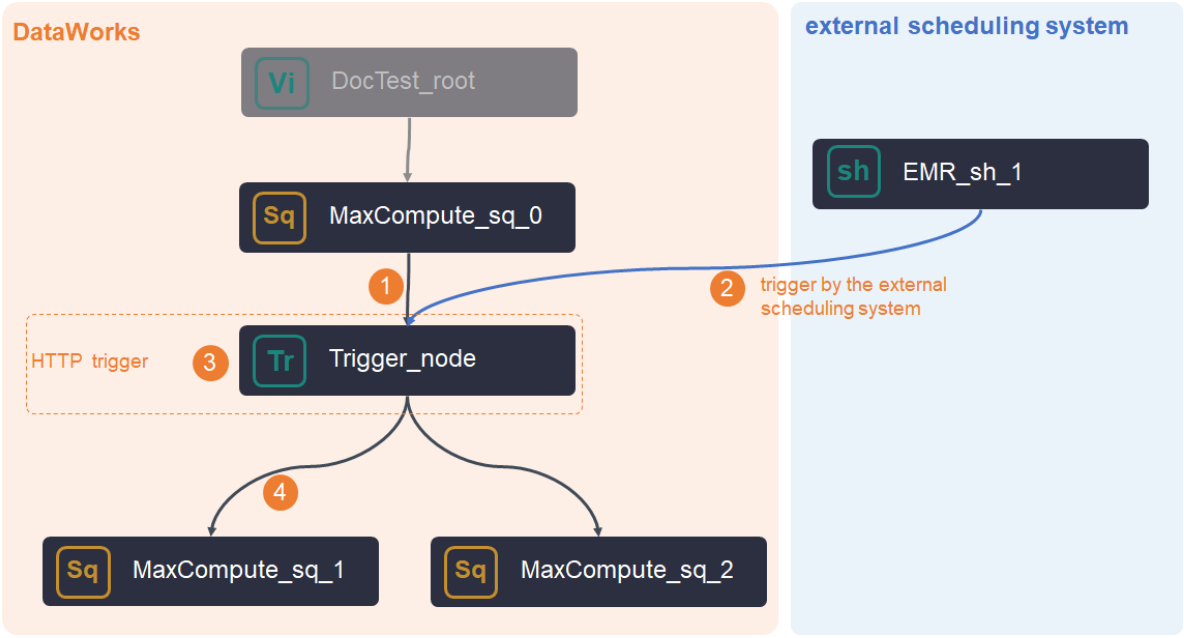
An external scheduling system is used to trigger nodes in the following typical scenarios:

- An HTTP Trigger node has no ancestor nodes other than the root node of the workflow.



In this scenario, you must configure a trigger in the external scheduling system after you create the HTTP Trigger node. Then, you must configure scheduling properties for each node in DataWorks. For more information, see Create an HTTP Trigger node and Configure triggers in an external scheduling system.

- An HTTP Trigger node has an ancestor node.



In this scenario, take note of the following items:

  - You must configure a trigger in the external scheduling system after you create the HTTP Trigger node. Then, you must configure the scheduling properties for each node in DataWorks. For more information, see Create an HTTP Trigger node and Configure triggers in an external scheduling system.
  - By default, the HTTP Trigger node uses the root node of the workflow as its ancestor node. You must manually change the ancestor node of the HTTP Trigger node to the required node.

- The HTTP Trigger node can trigger its descendant nodes only after the ancestor node of the HTTP Trigger node is run as expected and the HTTP Trigger node receives a scheduling instruction from the external scheduling system.

  If the HTTP Trigger node receives a scheduling instruction from the external scheduling system before the ancestor node of the HTTP Trigger node is run, the HTTP Trigger node does not trigger its descendant nodes. The DataWorks scheduling system retains the scheduling instruction from the external scheduling system and schedules the HTTP Trigger node to trigger the descendant nodes after execution of the ancestor node is complete.

  > **Notice** The scheduling instruction from the external scheduling system can be retained only for 24 hours. If the execution of the ancestor node is not complete within 24 hours, the scheduling instruction from the external scheduling system becomes invalid and is discarded.

Limits

- Only DataWorks Enterprise Edition and a more advanced edition support HTTP Trigger nodes. HTTP Trigger nodes are available in the regions within China and the Singapore and Germany (Frankfurt) regions.
- HTTP Trigger nodes serve only as nodes that trigger other compute nodes. HTTP Trigger nodes cannot be used as compute nodes. You must configure the nodes that need to be triggered as the descendant nodes of an HTTP Trigger node.
- If you want to rerun an HTTP Trigger node after a workflow is created and run, you must enable the external scheduling system to send a scheduling instruction again.
- If you want to obtain the historical results of the descendant nodes of an HTTP Trigger node after a workflow is created and run, you must backfill data for the descendant nodes. For more information, see Perform retroactive data generation and view retroactive data generation instances. The HTTP Trigger node does not wait for scheduling instructions on data backfill from the external scheduling system. Instead, the HTTP Trigger node directly triggers its descendant nodes to backfill data.

Remarks

The HTTP Trigger node can be run only if the following requirements are met:

- Auto triggered node instances are generated for the HTTP Trigger node. You can find the instances on the **Cycle Instance** page in Operation Center.
- All ancestor nodes on which the HTTP Trigger node depends are run as expected. The status of the ancestor nodes is Succeeded.
- The scheduled time of the auto triggered node instances generated for the HTTP Trigger node arrives.
- Sufficient scheduling resources are available for use when the HTTP Trigger node is run.
- The status of the HTTP Trigger node is not Freeze.
- The HTTP Trigger node can be triggered only if it is in the Pending state. If the HTTP Trigger node is triggered, it cannot be triggered again.

Create an HTTP Trigger node

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.

iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.

2. On the **DataStudio** page, move the pointer over the **Create** icon and choose **General > HTTP Trigger**.

   Alternatively, find the workflow in which you want to create an HTTP Trigger node, click the workflow name, right-click **General**, and then choose **Create > HTTP Trigger**.

3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.

5. On the configuration tab of the node, click **Properties** in the right-side navigation pane. On the Properties tab, configure scheduling properties for the node. For more information, see Configure basic properties.

   > **Note** By default, the HTTP Trigger node uses the root node of the workflow as its ancestor node. You must manually change the ancestor node of the HTTP Trigger node to the required node.

6. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.

7. Test the node. For more information, see View auto triggered nodes.

Configure triggers in an external scheduling system

You can use Alibaba Cloud SDK for Java or Python to configure a trigger in an external scheduling system or call an API operation to run an HTTP Trigger node.

- Alibaba Cloud SDK for Java

  i. Install Alibaba Cloud SDK for Java.For more information, see Get started with Alibaba Cloud Classic SDK for Java.

  Specify the following Project Object Model (POM) configurations to use DataWorks SDK for Java:

```
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-dataworks-public</artifactId>
  <version>3.4.2</version>
</dependency>
```

ii. Use the following sample code and set the parameters in the code.

```
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) {
// Specify the region ID, AccessKey ID, and AccessKey secret.
// Replace cn-hangzhou with the ID of the region in which the node to be triggered re
sides.
// <accessKeyId> indicates the AccessKey ID.
// <accessSecret> indicates the AccessKey secret.
DefaultProfile profile = DefaultProfile.getProfile("cn-hangzhou", "<accessKeyId>", "<
accessSecret>");
IAcsClient client = new DefaultAcsClient(profile);
RunTriggerNodeRequest request = new RunTriggerNodeRequest();
// Specify the ID of the HTTP Trigger node. You can call the ListNodes operation to q
uery the ID.
request.setNodeId(700003742092L);
// Specify the timestamp for running the HTTP Trigger node. Convert the scheduled tim
e to run the HTTP Trigger node to a timestamp.
// If the region in which the HTTP Trigger node resides and the region in which the s
cheduling system resides are in different time zones, specify the timestamp of the ti
me zone in which the HTTP Trigger node resides.
// For example, the HTTP Trigger node resides in the China (Beijing) region, the node
is scheduled to run at 18:00:00 (UTC+8), and the scheduling system resides in the US
(Silicon Valley) region. In this case, specify the timestamp that corresponds to 18:0
0:00 (UTC+8).
request.setCycleTime(1605629820000L);
// Specify the data timestamp of the HTTP Trigger node instance.
// The data timestamp is one day earlier than the scheduled time of the HTTP Trigger
node and is accurate to the day. The hour, minute, and second are presented as 000000
00. For example, the HTTP Trigger node is scheduled to run on November 25, 2020. In t
his case, you must convert the date-based time to the data timestamp 2020112400000000
.
// If the region in which the HTTP Trigger node resides and the region in which the s
cheduling system resides are in different time zones, specify the timestamp of the ti
me zone in which the HTTP Trigger node resides.
request.setBizDate(1605542400000L);
// Specify the ID of the DataWorks workspace to which the HTTP Trigger node belongs.
You can call the ListProjects operation to query the ID.
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());
}
}
}
```

- Alibaba Cloud SDK for Python
  i. Install Alibaba Cloud SDK for Python.For more information, see Install the Classic SDK and its core library.

     Run the following command to install DataWorks SDK for Python:

     ```
     pip install aliyun-python-sdk-dataworks-public==2.1.2
     ```

  ii. Use the following sample code and set the parameters in the code.

```
#!/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
 # Specify the region ID, AccessKey ID, and AccessKey secret.
# Replace cn-hangzhou with the ID of the region in which the node to be triggered res
ides.
# <accessKeyId> indicates the AccessKey ID.
# <accessSecret> indicates the AccessKey secret.
client = AcsClient('<accessKeyId>', '<accessSecret>', 'cn-hangzhou')
request = RunTriggerNodeRequest()
request.set_accept_format('json')
# Specify the ID of the HTTP Trigger node. You can call the ListNodes operation to qu
ery the ID.
request.set_NodeId(123)
# Specify the timestamp for running the HTTP Trigger node. Convert the scheduled time
to run the HTTP Trigger node to a timestamp.
# If the region in which the HTTP Trigger node resides and the region in which the sc
heduling system resides are in different time zones, specify the timestamp of the tim
e zone in which the HTTP Trigger node resides.
# For example, the HTTP Trigger node resides in the China (Beijing) region, the node
is scheduled to run at 18:00:00 (UTC+8), and the scheduling system resides in the US
(Silicon Valley) region. In this case, specify the timestamp that corresponds to 18:0
0:00 (UTC+8).
request.set_CycleTime(1606321620000)
# Specify the data timestamp of the HTTP Trigger node instance.
# The data timestamp is one day earlier than the scheduled time of the HTTP Trigger n
ode and is accurate to the day. The hour, minute, and second are presented as 0000000
0. For example, the HTTP Trigger node is scheduled to run on November 25, 2020. In th
is case, you must convert the date-based time to the data timestamp 20201124000000000.
# If the region in which the HTTP Trigger node resides and the region in which the sc
heduling system resides are in different time zones, specify the timestamp of the tim
e zone in which the HTTP Trigger node resides.
request.set_BizDate(1606233600000)
# Specify the ID of the DataWorks workspace to which the HTTP Trigger node belongs. Y
ou can call the ListProjects operation to query the ID.
request.set_AppId(11456)
response = client.do_action_with_exception(request)
# python2: print(response)
print(str(response, encoding='utf-8'))
```

- API operation

  For more information about the API operation, see RunTriggerNode.

5.11.10. Create a parameter node

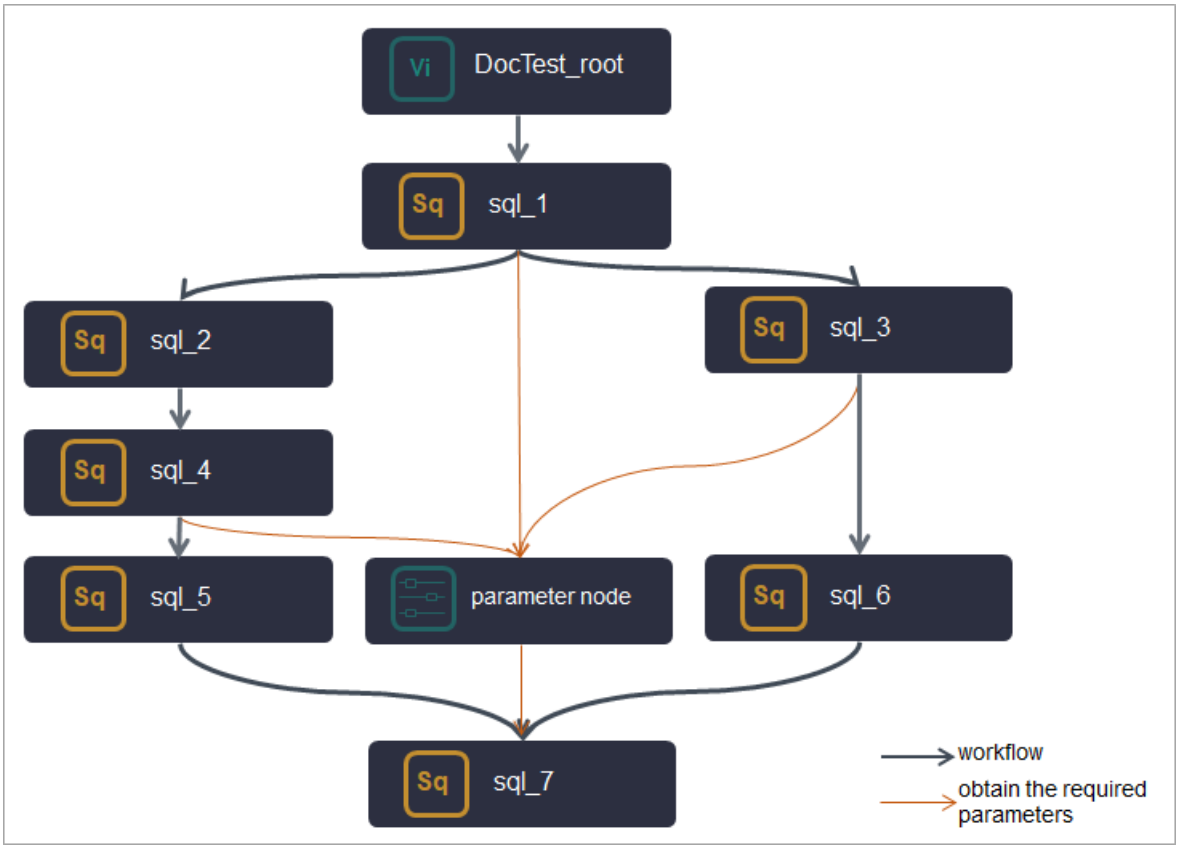
A parameter node is a special type of virtual node. This type of node is used to manage parameters in workflows and pass parameters between nodes. Parameter nodes can be used to manage constant and variable parameters and transparently pass the parameters of ancestor nodes. Nodes that need to use parameters can obtain parameters from parameter nodes. This topic describes how to create a parameter node.

Context

Parameter nodes are virtual nodes. They do not run computing tasks. They are used to pass parameters between nodes and manage parameters in workflows.

- **Parameter passing between nodes**

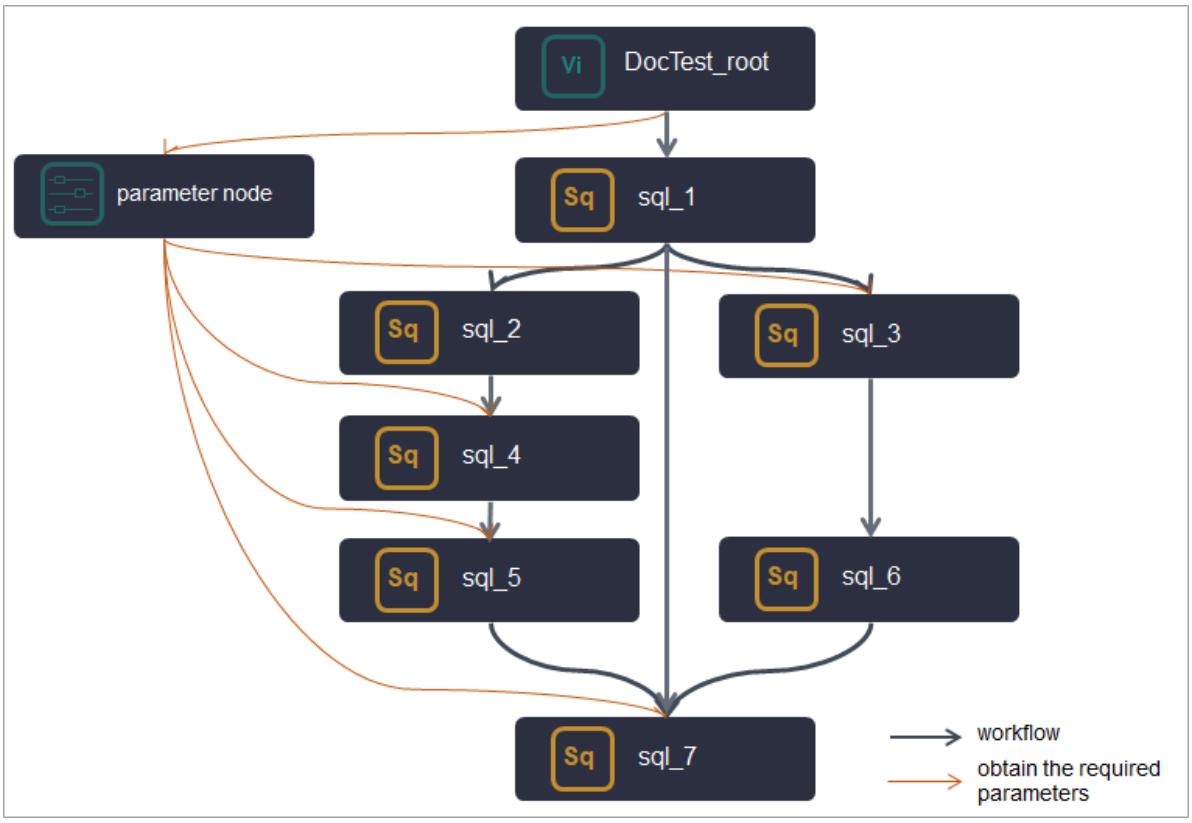
  If Node A in a workflow needs to obtain the output parameters of its ancestor nodes, you can create a parameter node and use it as an ancestor node of Node A and a descendant node of the ancestor nodes of Node A. Then, add all the parameters required by Node A to the parameter node. This way, Node A can obtain all the required parameters from the parameter node.



  In the preceding figure, the sql_7 node needs to obtain the output parameters of the sql_1, sql_3, and sql_4 nodes. A parameter node is created and used as a descendant node of the sql_1, sql_3, and sql_4 nodes and as an ancestor node of the sql_7 node. All the parameters required by the sql_7 node are added to the parameter node, and the sql_7 node can obtain all the required parameters from the parameter node.

- **Parameter management**

If nodes in a workflow need to use some constant and variable parameters, you can create a parameter node and use it as an ancestor node of these nodes. Then, add all the parameters required by these nodes to the parameter node. This way, these nodes can obtain all the required parameters from the parameter node. The parameter node facilitates centralized management of all the parameters used in the workflow.



In the preceding figure, the sql_3, sql_4, sql_5, and sql_7 nodes need to use some parameters. A parameter node is created and used as an ancestor node of the sql_3, sql_4, sql_5, and sql_7 nodes. All the parameters required by these nodes are added to the parameter node, and these nodes can obtain all the required parameters from the parameter node.

Precautions

If a node needs to use parameters in a parameter node, the node must be a direct descendant node of the parameter node.

Create a parameter node

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the **DataStudio** page, move the pointer over the **+Create** icon and choose **General > Params Node**.

   Alternatively, find the required workflow in the Scheduled Workflow pane, click the workflow name, right-click **General**, and then choose **Create > Parameter nodes**.

3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.

Configure scheduling properties for the parameter node

After the parameter node is created, you can configure scheduling properties for the parameter node based on your business requirements.

Scheduling properties include basic properties, time properties, resource properties, and scheduling dependencies. A parameter node does not run data development tasks. It is used only to manage parameters and transparently pass parameters. Therefore, you must pay special attention to the scheduling dependencies configured for a parameter node.

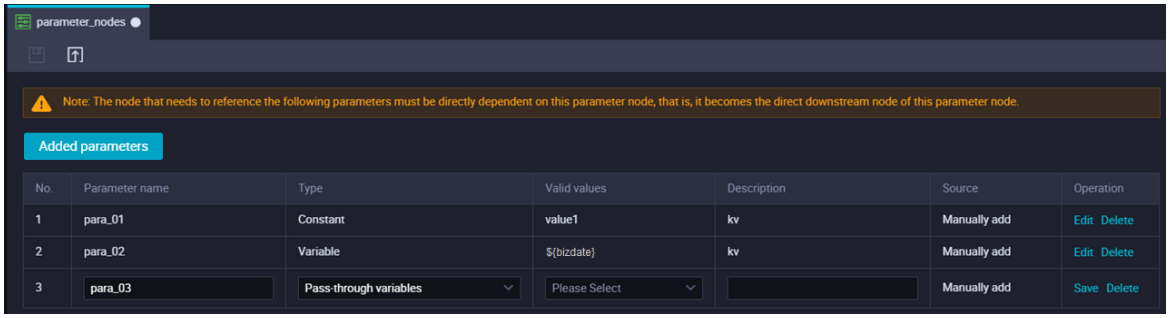
- If a node needs to use the parameters in a parameter node, the node must be a descendant node of the parameter node.
- Ancestor nodes whose parameters need to be transparently passed must be the ancestor nodes of the parameter node.

For more information about how to configure scheduling properties for a parameter node, see Configure basic properties, Configure time properties, Configure a resource group, and Configure same-cycle scheduling dependencies.

Add parameters to the parameter node

After you configure scheduling properties for the parameter node, you can add the parameters that you want to manage and transparently pass to the parameter node for subsequent management and use. This section describes the procedure in detail.

1. On the configuration tab of the parameter node, click **Added parameters**.
2. Specify **Parameter name**, **Type**, **Valid values**, and **Description**, and click **Save**.



   The valid values of Type include **Constant**, **Variable**, and **Pass-through variables**.

   - **Constant**: indicates that the value of the parameter is a constant.
   - **Variable**: indicates that the value of the parameter is a variable. If you want to use a variable such as the system time, set the value of Type to Variable. For more information about variable parameters, see Overview of scheduling parameters.
   - **Pass-through variables**: Pass-through variables are used to transparently pass the output parameters of ancestor nodes of the parameter node to descendant nodes of the parameter node. If you add a parameter of the pass-through variable type, you can set Valid values to an output parameter of an ancestor node of the parameter node.

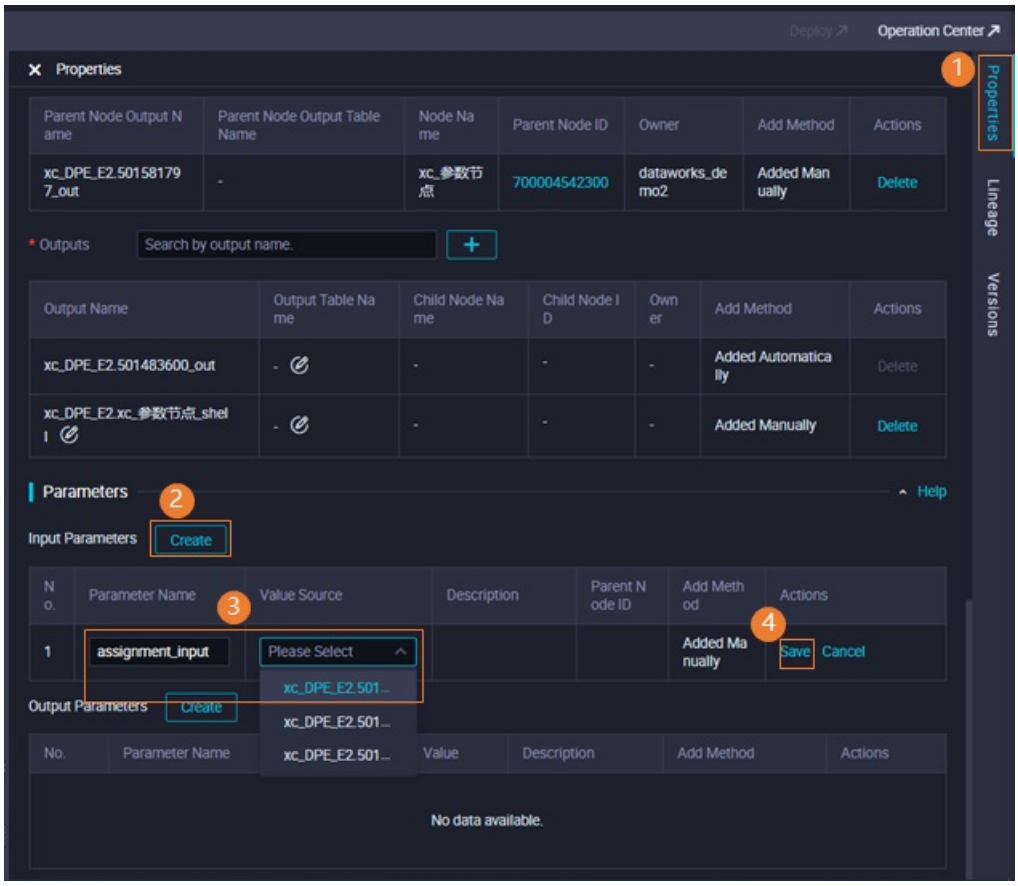
What to do next: Use the parameters in the parameter node

After you configure the parameter node, the descendant nodes of the parameter node can directly use the parameters in the parameter node. This facilitates centralized management of parameters and improves the task development efficiency of descendant nodes.

Before the descendant nodes of the parameter node can use the parameters in the parameter node, you must configure the parameters as the input parameters of the descendant nodes. This way, these parameters can be directly used in the code of the descendant nodes.

1. Configure the parameters as the input parameters of the descendant nodes.

   On the configuration tab of a descendant node, click **Properties** in the right-side navigation pane. On the Properties tab, add the parameters required by the descendant node in the **Input Parameters** section of **Parameters**.



   The following parameters in the Input Parameters section must be specified:

   - Parameter Name: the name of the parameter that the descendant node needs to use. You can view the parameter name in the parameter node.
   - Value Source: the source of the parameter that the descendant node needs to use.

     If the parameter information is not completely displayed, you can move the pointer over it to view the complete information. All the parameters that are added to the parameter node are displayed in the **Output name of the parameter node:Parameter name** format. You can select the source name that is ended with the name of the required parameter.

2. Use the parameters in the code of the descendant node.

5.11.11. Create an FTP Check node

An FTP Check node can be used to periodically detect whether a specific file exists based on File Transfer Protocol (FTP). If the FTP Check node detects that the file exists, the scheduling system starts to run the descendant node of the FTP Check node. Otherwise, the FTP Check node retries the detection based on the configured detection interval. The FTP Check node stops the retry until the condition for stopping the detection is met. In most cases, FTP Check nodes are used for communications between the DataWorks scheduling system and external scheduling systems. This topic describes how to use an FTP Check node and the related precautions.

Prerequisites

- An FTP data source is added. For more information, see Add an FTP data source.
- An exclusive resource group for scheduling is created. For more information, see Billing of exclusive resource groups for scheduling (subscription).
- A workflow is created. For more information, see Create a workflow.

Context

An FTP Check node is typically used in the following scenario: A node in the DataWorks scheduling system needs to access an external database in an external scheduling system, but an ongoing data write task for the database is not performed by DataWorks. In this case, the time when the data write task is completed and the time when the database can be accessed are unknown to DataWorks. If the node accesses the database, the data that is read from the database may be incomplete or the data read fails because the data write task is not completed. To ensure that the node can successfully read data from the external database, you can enable the external scheduling system to generate a mark that indicates the data write task is completed. For example, you can enable the external scheduling system to generate a marker file with the suffix `.done` in the file system to indicate that the data write task is completed. Then, you can create an FTP Check node in the DataWorks scheduling system to periodically detect whether the marker file with the suffix `.done` exists. If the file exists, the node that needs to access the external database can be scheduled.

> **Note**
> - You can specify the file system that can be used to store the marker files.
> - In this example, a marker file with the suffix `.done` is used. You can customize the information such as the format and name for your marker file.

> **Note** External databases include but are not limited to Oracle, MySQL, and SQL Server.

Limits

- Only the China (Beijing), China (Shanghai), China (Hangzhou), China (Shenzhen), China (Zhangjiakou), China (Chengdu), and Singapore (Singapore) regions support FTP Check nodes.
- FTP Check nodes can run only on exclusive resource groups for scheduling.
- If an FTP Check node is scheduled by minute or hour, you can set the **Check stop policy** parameter only to **Number of Check stops** for the node.

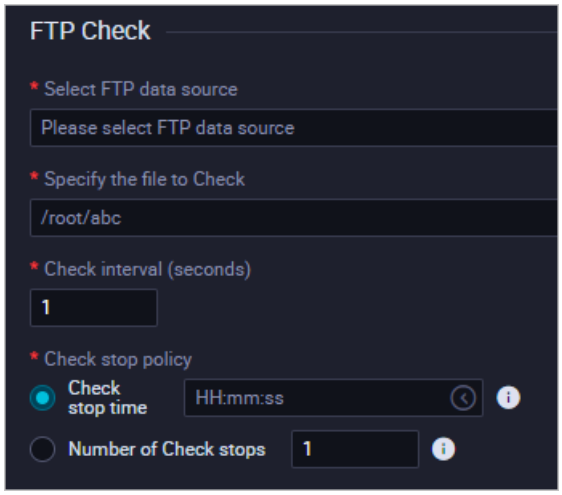
Create an FTP Check node

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the **DataStudio** page, move the pointer over the **+Create** icon and choose **General > FTP Check**.

   Alternatively, you can find the desired workflow, right-click the workflow name, and then choose **Create > General > FTP Check**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. Click the **Properties** tab in the right-side navigation pane and configure properties for the FTP Check node.

   The properties include **basic properties**, **time properties**, **resource properties**, and **scheduling dependencies**. For more information, see Configure basic properties, Configure time properties, Configure a resource group, and Configure same-cycle scheduling dependencies.
6. Configure a detection object and a detection policy.



   i. Select the FTP data source that you want to detect from the **Select FTP data source** drop-down list.

      You can select an FTP or SFTP data source. If no data source is available, you must add one. For more information, see Add an FTP data source.
   ii. Specify the path of the marker file in the **Specify the file to Check** field. If the file path that you specified is dynamic, you can use scheduling parameters to configure variable paths in the file path. For more information, see Overview of scheduling parameters.
   i. Specify an interval at which the detection is performed in the **Check interval (seconds)** field.

ii. Select a policy for **Check stop policy**. The following policies are available:

- **Check stop time**: the point in time when the detection stops. Specify this parameter in the `hh24:mi:ss` format. The time format is based on the 24-hour clock. If no marker file is detected each time the FTP Check node is run, the detection fails. In this case, the system does not schedule the descendant node of the FTP Check node. The system starts to schedule the descendant node only after the detection succeeds. If the previous detection fails, the node continues the detection based on the configured detection interval and stops the detection until the time that you specified for stopping the detection is reached. You can view the node logs to find the detailed cause of the failure.

  > **Note** The scheduling cycle of the FTP Check node affects its stop policy.
  >
  > - If the FTP Check node is scheduled by **minute** or **hour**, the Check stop policy parameter can be set only to **Number of Check stops** for the node. For more information, see Configure a detection policy for an FTP Check node.
  > - If you want to change the scheduling cycle of the FTP Check node for which Check stop policy is set to **Check stop time** from **day** to **minute** or **hour**, the **Check stop time** policy becomes invalid. In this case, you must set Check stop policy to **Number of Check stops**. Otherwise, the FTP Check node cannot be committed.

- **Number of Check stops**: the maximum number of times that the detection can be performed. If no marker file is detected each time the FTP Check node is run, the detection fails. In this case, the system does not schedule the descendant node of the FTP Check node. The system starts to schedule the descendant node only after the detection succeeds. If the detection fails, the node continues the detection based on the specified detection interval and stops the detection when the maximum number of times that the detection can be performed is reached. You can view the node logs to find the detailed cause of the failure.

7. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.

8. Test the node. For more information, see View auto triggered nodes.

5.12. Custom nodes

5.12.1. Node configuration

5.12.1.1. Overview

DataStudio supports default node types such as ODPS SQL and Shell nodes, and also allows you to create custom node types.

Prerequisites

DataWorks Enterprise Edition or a more advanced edition is activated so that you can use custom node type.

Limits

Only DataWorks Enterprise Edition or a more advanced edition supports the custom node type.

Go to the Node Config page

1. Log on to the DataWorks console.
2. In the left-side navigation pane, click **Workspaces**.
3. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **Data Development** in the Actions column.
4. Click **Node Config** in the upper-right corner of the DataStudio page. The **Node Plugin list** tab appears.

   The Node Config page consists of four tabs: **Node Plugin list**, **Default Node Types**, **Custom Node Types**, and **Data quality plug-in list**.

> **Notice** Only the owner and administrators of a workspace can perform operations on the Node Config page of the workspace.

View wrappers

A wrapper defines the core processing logic of a node type. Assume that you write an SQL statement in the code editor for an ODPS SQL node and commit the statement. In this case, DataWorks uses the wrapper for the specific ODPS SQL node type to parse and execute the statement. Before you create a custom node type, you must create a wrapper to develop the processing logic of the node type. You can use only Java to develop processing logic in a wrapper.

The **Node Plugin list** tab displays all existing wrappers. You can click **Create** in the upper-right corner to create a wrapper. For more information, see Create a wrapper.

The values in the **Latest Version**, **Version in Development Environment**, and **Version in Production Environment** columns for each wrapper are displayed based on the following rules:

- If the node is created but not deployed, **Not Deployed** is displayed.
- If the node is deployed, the version and deployment time are displayed.
- If the node is being deployed, **Deploying** is displayed.

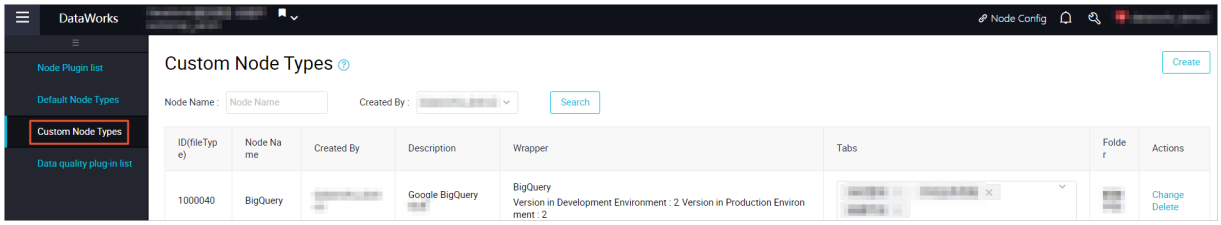
View default node types

In the left-side navigation pane of the **Node Config** page, click **Default Node Types** to view values of **Node Name** and **Tabs** for each default node type. By default, **DataStudio** is displayed in the Tabs column.

> **Note** You can only view data but cannot perform operations on the **Default Node Types** tab.

View custom node types

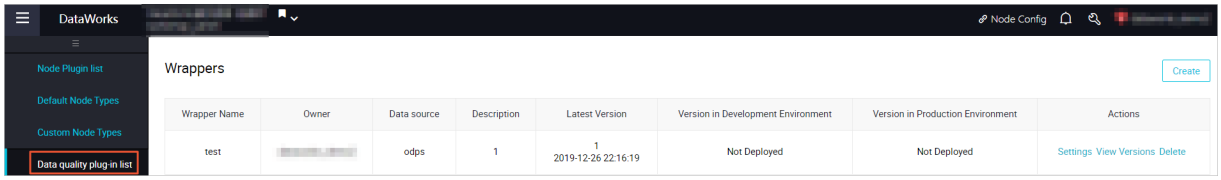
In the left-side navigation pane of the **Node Config** page, click **Custom Node Types**. On this tab, you can create, view, edit, and delete custom node types. These custom node types can be used on the **DataStudio** page. For more information, see Create a custom node type.



View data quality wrappers

DataWorks allows you to create, deploy, and use data quality wrappers to satisfy diversified needs for data quality.

In the left-side navigation pane of the **Node Config** page, click **Data quality plug-in list**. On this tab, you can create, configure, view, and delete data quality wrappers.



5.12.1.2. Develop a custom wrapper package

To run a task based on a custom node, you must use a custom wrapper. Before you use a custom node, you must create a custom wrapper package and upload and deploy the package to DataWorks. This topic describes how to create a custom wrapper package.

Context

The following methods are used during the running of DataWorks custom nodes and the development and use of wrappers:

- `submitJob(String codeFilePath, List args)`: submits a node that corresponds to a wrapper.

  Input parameters:

- codeFilePath: specifies the absolute path that is used to store the developed code.
- List args: specifies the scheduling parameters that you want to configure. The value is in the `{"key"="value","key2"="value2"...}` format.

Returned results:

- 0: indicates that the task succeeded.
- 2: indicates that the scheduling system needs to rerun the task.
- 1, 4, or 3: indicates that the task is terminated.
- Other values: indicates that the task failed.

- `killJob()`: listens to the task termination indication and triggers an operation. This method does not have input parameters or provide returned results.

> **Note** If the task is not processed at the business layer within 9 seconds, the wrapper is forcibly terminated and the task fails.

Download a dependency JAR package

Click alisa-wrapper-face-1.0.0.jar to download the dependency JAR package.

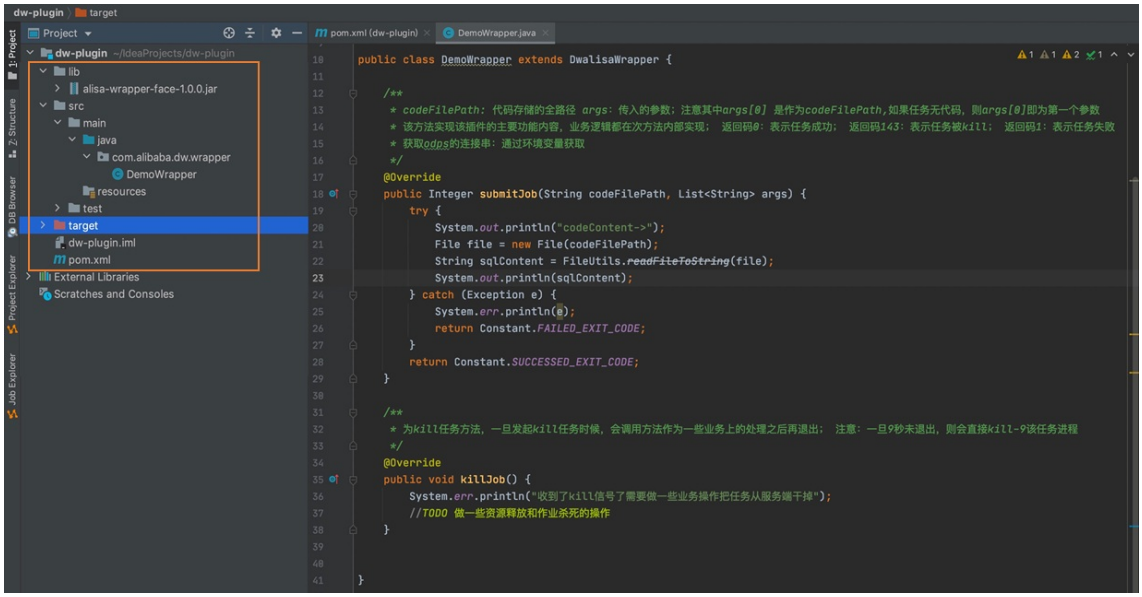
Package code

1. Create a wrapper code project.

   Use an integrated development environment (IDE) to create a Maven project. The following description provides the structure requirements for files and the configuration requirements for the pom.xml file:

   - Structure requirements for files

     The following figure shows the structure requirements for files. alisa-wrapper-face-1.0.0.jar in the lib folder is the downloaded dependency JAR package. This package is introduced to the project in the same way as an on-premises package.



   - Configuration requirements for the pom.xml file

The following code provides an example for the configuration of the pom.xml file. The class name of the wrapper (indicated by `<mainclass>`) is the classpath of the submitJob method, such as `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. Develop code.

   Example 1: Basic logic

```
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 is in the alisa-wrapper-face package.
**/
public class DemoWrapper extends DwalisaWrapper {
   /**
    * codeFilePath: the full path that is used to store the code.
    * args: the input parameters. args[0] is used as codeFilePath. If the task has no c
ode, args[0] is the first parameter.
    * This method implements the main features of the wrapper, and the business logic i
s implemented within the method.
    * The return code 0 indicates that the task succeeded.
    * Return codes 1, 4, and 3 indicate that the task is terminated.
    * Other return codes indicate that the task failed.
    */
   @SuppressWarnings("deprecation")
   @Override
   public Integer submitJob(String codeFilePath, List<String> args) {
       try {
           System.err.println("your code->");
           // Implement the business code. The task is finished after this method is ru
n.
       } catch (Exception e) {
           System.err.println(e);
       }
       System.out.println("task finished...");
       return Constant.SUCCESSED_EXIT_CODE;
   }
   /**
    * After task termination is initiated, the method is called to perform business pro
cessing before the method is terminated.
    * If the method is not terminated within 9 seconds, the system returns kill-9 to te
rminate the task.
    *
    */
   @Override
   public void killJob() {
       System.err.println("After the task termination indication is detected, some busi
ness operations are performed to terminate the task from the server.");
   }
}
```

Example 2: A custom node developed based on a data source

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: the full path that is used to store the code. args: the input parameters. args[0] is used as codeFilePath. If the task has no code, args[0] is the first parameter.
     * This method implements the main features of the wrapper, and the business logic is implemented within the method. The return code 0 indicates that the task succeeded. Return codes 1, 4, and 3 indicate that the task is terminated. The return code 1 indicates that the task failed. Obtain the information of a MaxCompute project: Obtain environment variables: 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 Exception {
        String content = sqlContent;
        if (args == null || args.size() <= 0) {
            return content;
        }
        // Replace ${...} parameters
        for (String keyValue : args) {
            System.out.println(args);
            if (keyValue.contains("=")) {
                String[] param = keyValue.split("=");
                String target = "${" + 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;
    }
    /**
     * The method that is used to kill the task. After task termination is initiated, t
he method is called to perform business processing before the method is terminated. If
the method is not terminated within 9 seconds, the system returns kill-9 to terminate t
he task.
     */
    @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) Perform authentication and obtain the connection string.
             */
            System.out.println("Connecting to Server...");
            String jdbcConn = System.getenv("SKYNET_CONNECTION");
            // Obtain and parse the connection string to establish a connection. The st
ring is in the JSON format.
            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) Execute SQL statements.
                 */
                boolean hasResult = stmt.execute(sql);
                if (hasResult) {
                    /**
                     * (3) Generate results and store the results in a file.
                     */
                    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();
        // Obtain the path that is used to store the result file.
        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. Configure the system to display the query result sets on pages.

   After the configuration, you can view the data that is queried by using a custom node on the related page in the DataWorks console.

   After the code is run, you must store the obtained result sets, such as queried table records, in a specific directory in accordance with the following rules:

   - Method to obtain the result sets

     ```
     String dataFilePath =String.format("%s/%s.data", System.getenv(Constant.TASK_EXEC_PAT
     H), System.getenv(Constant.ALISA_TASK_ID));
     ```

     `System.getenv` indicates the method that is used to obtain environment variables. For more information, see Appendix: Use environment variables to obtain the information of the related node.

   - Storage format
     - File format: CSV. The fields in the file are separated by commas (,).
     - Row distribution: The first row is field names, and other rows are field values.

Example:

```
"name","age"
"Alice","12"
```

4. After code is developed, package the Maven project.

   Compress the dependencies of the project into a JAR package.

5. Upload the wrapper package.

   If external dependencies such as Protobuf and Guava are used, you must compress the dependencies and the preceding JAR package into a ZIP package. Then, upload the ZIP package to DataWorks as a wrapper. For more information, see Create a wrapper.

Appendix: Use environment variables to obtain the information of the related node

You can use `System.getenv("SKYNET_ONDUTY")` to obtain the values of the environment variables.

- SKYNET_ID: the ID of the node. This variable is available only when the node is scheduled to run.
- SKYNET_BIZDATE: the data timestamp.
- SKYNET_ONDUTY:
  - Indicates the operator ID during temporary running, data backfill, or tests.
  - Indicates the ID (employee ID or baseId) of the node owner.
- SKYNET_TASKID: the node instance.
- IDSKYNET_SYSTEM_ENV: the environment where the node runs, such as the development and production environment.
- SKYNET_CYCTIME: the time at which the node instance is scheduled to run.
- SKYNET_CONNECTION: the connection string. In most cases, the string is in the JSON format.
- SKYNET_TENANT_ID: the tenant ID.

5.12.1.3. Create a wrapper

Before you use a wrapper, you must create and configure the wrapper, deploy the wrapper to and test it in the development environment, and then deploy the wrapper to the production environment.

Context

A wrapper defines the core processing logic of a node type. You can use only Java to develop processing logic in a wrapper. For example, after you write an SQL statement in the code editor for an ODPS SQL node and commit the statement, DataWorks uses the wrapper for the ODPS SQL node type to parse and execute the statement. Before you create a custom node type, you must create a wrapper to develop the processing logic of the node type.

Procedure

1. Create a wrapper.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.

iii. In the top navigation bar, select the region where the target workspace resides. Find the target workspace and click **Data Analytics** in the Actions column.

iv. On the DataStudio page, click **Node Config** in the top navigation bar. The **Node Plugin list** tab appears by default.

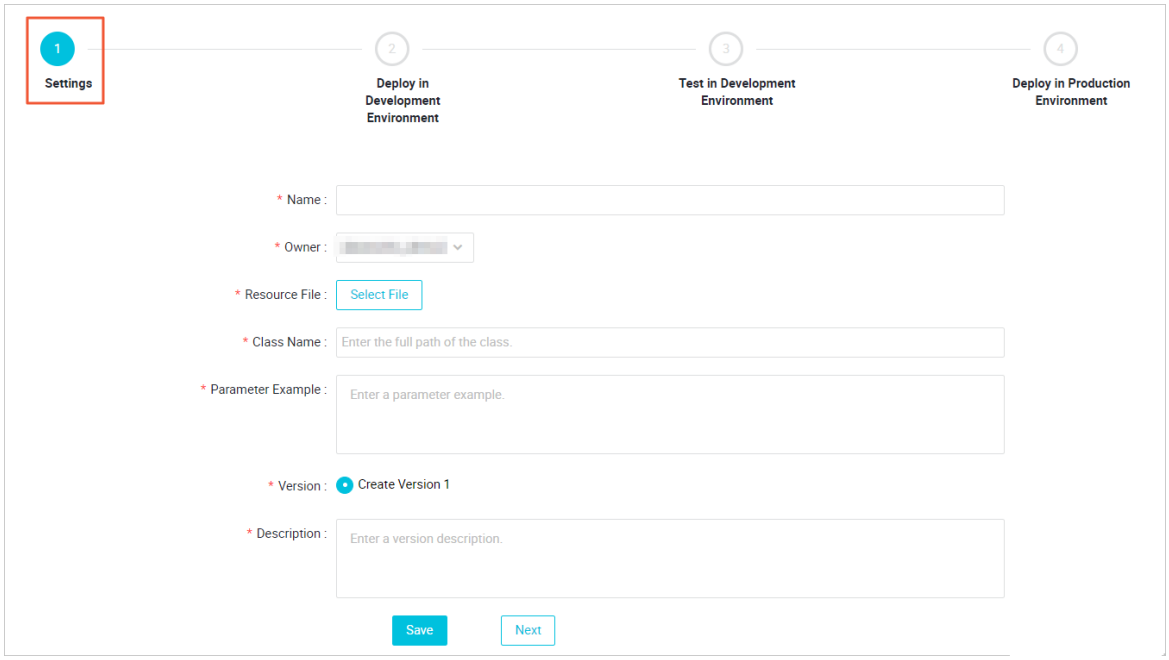
v. Click **Create** in the upper-right corner.

vi. In the **Please select plug-in type** dialog box, select a wrapper type and click **OK**. By default, **Engine type** is selected.

You can select one of the following wrapper types in the Please select plug-in type dialog box:

- **Engine type**: If you select this wrapper type, you must upload a code package, for example, a JAR package, to define the functionality of the wrapper. Generally, a wrapper of this type is applicable to a node type that uses a custom compute engine.
- **Business type**: If you select this wrapper type, you must use a manually triggered workflow to define the functionality of the wrapper. Generally, a wrapper of this type is used to encapsulate multiple nodes to implement specific business logic.

2. Configure the wrapper in the wizard that appears. In the **Settings** step, set relevant parameters.



Parameter	Description
Name	The name of the wrapper. The name must start with a letter and can contain only letters, underscores (_), and digits.
Owner	The owner of the wrapper. You can select yourself or another member of the current workspace. If you select another member as the owner, note the following points: ○ An administrator of a workspace cannot edit the custom wrappers of other members of the workspace. ○ Only the owner of a workspace can edit the wrappers of other members.

Parameter	Description
Resource File	The resource file that defines the functionality of the wrapper. You can use one of the following methods to specify a resource file: **Upload a local file**, **Use an OSS object**, and **AppStudio**. **Note** ○ This parameter is available only when you select **Engine type** in the Please select plug-in type dialog box. ○ The size of a local file can be up to 50 MB, and the size of a file that is stored in an Object Storage Service (OSS) bucket can be up to 200 MB. Click **Select File**. In the **Select File** dialog box, select a method, complete corresponding configurations, and then click **OK**.
Class Name	The full path of the class for implementing the wrapper. **Note** This parameter is available only when you select **Engine type** in the Please select plug-in type dialog box.
Parameter Example	The parameters that are designed based on the specified resource file. **Note** This parameter is available only when you select **Engine type** in the Please select plug-in type dialog box.
Manual business process	The manually triggered workflow that defines the functionality of the wrapper. Click **Select manual business process**. In the Select manual business process dialog box, select a manually triggered workflow from the drop-down list and click **OK**. **Note** This parameter is available only when you select **Business type** in the Please select plug-in type dialog box.
Version	The version of the wrapper. In this example, select Create Version 1. When you are editing or rolling back a version, select Overwrite Version.
Description	The description of the wrapper.

3. Click **Save** and then **Next**.

   After you click **Save**, the configurations in this step are saved by the system. If you are modifying a wrapper, note the following points:

   - If you modify the basic settings of the wrapper, the modification takes effect after you click Save. You do not need to deploy the wrapper again.
   - If you modify the resource file, for example, the JAR package, of the wrapper, the modification takes effect only after you deploy the wrapper again.

4. In the **Deploy in Development Environment** step, confirm the configurations and click **Deploy in Development Environment**. The real-time deployment progress appears.
5. Wait until the deployment is completed and click **Next**.
6. Test the wrapper in the development environment.
   i. In the **Test in Development Environment** step, set the **Parameter configuration in** and **Environment variables** parameters.
   ii. Click **Test**.
   iii. Confirm the test result and select the **Test Passed** check box.
   iv. Click **Next**.
7. In the **Deploy in Production Environment** step, click **Deploy to production environment**. The real-time deployment progress appears.

   > **Note** The wrapper to be deployed to the production environment must be of the latest version, have been deployed to the development environment, and have passed the test. Otherwise, a message appears, indicating that the deployment to the production environment fails.

8. Wait until the deployment is completed. Click **Complete**. The **Node Plugin list** tab appears.

   The information about the wrapper you created appears on this tab. You can perform the following operations on the wrapper:

   - Click **Settings** in the Actions column to configure the wrapper. After you click **Settings**, you are navigated to a step in the configuration wizard of the wrapper. Which step you are navigated to depends on existing configurations of the wrapper.
   - Click **View Versions** in the Actions column to view the version information about the wrapper. After you click **View Versions**, the **View Versions** dialog box appears. You can perform the following operations on each version:
     - **View**: View the basic information about the current version.
     - **Roll Back**: Roll back the wrapper from the current version to the selected version. After you click this button, the system creates a new version for the wrapper. In the new version, the wrapper uses the basic settings and resource file of the selected version. The new version number equals the latest version number among all the versions plus 1.
     - **Download**: Download the resource file of the current version.
   - Click **Delete** in the Actions column to delete the wrapper. After you click **Delete**, the **Delete Wrapper** message appears. Click **OK**.

     > **Notice** Before you delete a wrapper, make sure that the wrapper is not associated with any node types.

5.12.1.4. Create a custom node type

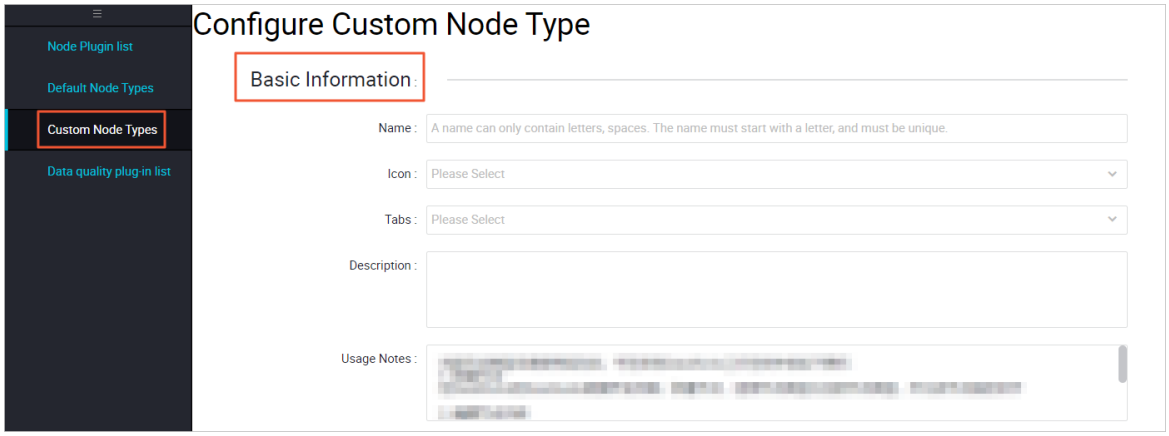
This topic describes how to create a custom node type. To create a custom node type, you must configure the basic information, specify a wrapper, and set code editor and interaction parameters.

Prerequisites

DataWorks Enterprise Edition or higher is activated.

Procedure

1. Go to the **Custom Node Types** tab.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region where the target workspace resides. Find the target workspace and click **Data Analytics** in the Actions column.
   iv. On the DataStudio page, click **Node Config** in the top navigation bar. The **Node Plugin list** tab appears by default.
   v. In the left-side navigation pane, click **Custom Node Types**.
2. On the Custom Node Types tab, click **Create** in the upper-right corner. The **Configure Custom Node Type** page appears.
3. In the **Basic Information** section, set relevant parameters.



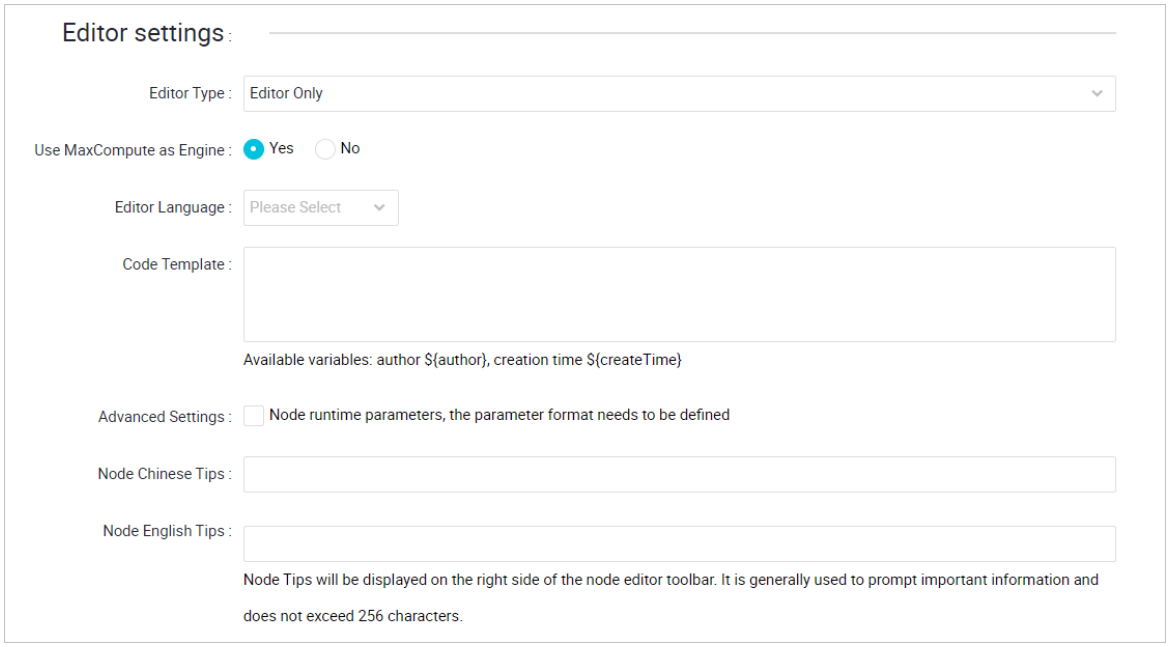
Parameter	Description
Name	The name of the custom node type. Make sure that the name is unique. The name must start with a letter and can contain letters and spaces.
Icon	The icon of the custom node type. Select an icon from the drop-down list.
Tabs	The services where the custom node type can be used. Valid values: **Temporary query**, **Manual business process**, and **Data Development**.
Description	The description of the custom node type.
Usage Notes	The usage notes of the custom node type. You can modify the default usage notes as required.

4. In the **Wrapper** section, set the **Plug-in type** and **Wrapper** parameters.

   The **Plug-in type** parameter specifies the type of wrapper to be associated with the node type. Valid values:

- **Engine type**: If you select this type, you must select a wrapper whose functionality is defined by using a code package, for example, a JAR package. Generally, a wrapper of this type is applicable to a node type that uses a custom compute engine.
- **Business type**: If you select this type, you must select a wrapper whose functionality is defined by using a manually triggered workflow. Generally, a wrapper of this type is used to encapsulate multiple nodes to implement specific business logic.

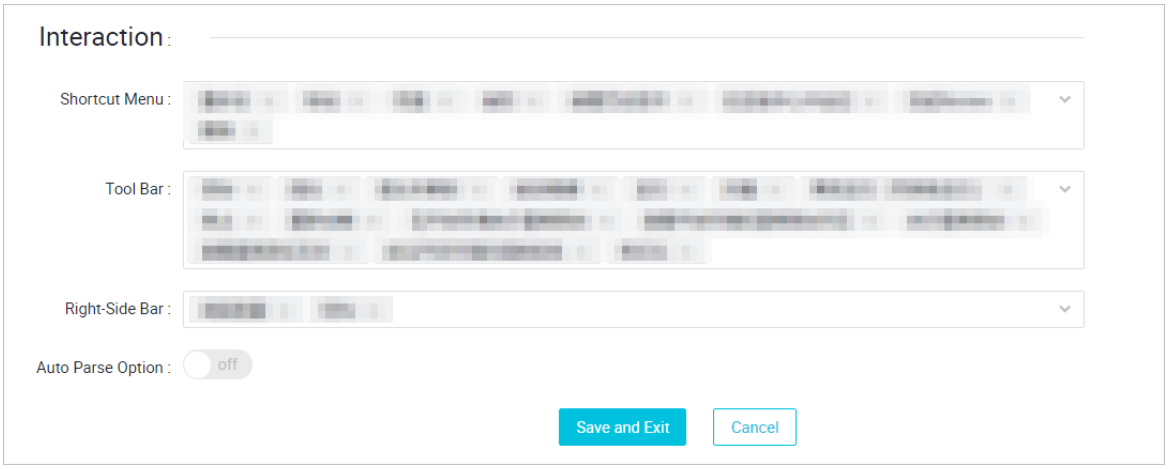
5. In the **Editor settings** section, set relevant parameters.



Parameter	Description
Editor Type	The type of the editor that is supported by the custom node type. Valid values: **Editor Only** and **Data Source Selection Section and Editor**
Use MaxCompute as Engine	Specifies whether to use MaxCompute as the compute engine. If your wrapper uses MaxCompute as the compute engine, select Yes. In other scenarios, select No. **Note** This parameter is available only when **Editor Type** is set to **Editor Only**.
Connection Type	The type of connections that are supported by the custom node type. You can select one or more types from the drop-down list. Only connections that are created by using URLs are supported. **Note** This parameter is available only when **Editor Type** is set to **Data Source Selection Section and Editor**.

Parameter	Description
Editor Language	The type of programming language that is supported by the custom node type. Valid values: ODPS SQL, JSON, Shell, Python, MySQL, XML, and YAML.
Code Template	The code template of the custom node type. You can use the ${author} and {createTime} variables in a code template, which indicate the creator and creation time of the template, respectively.
Advanced Settings	Specifies whether to configure a template for the runtime parameters of nodes of this type.
Parameter Template	The template of the runtime parameters of nodes of this type. **Note** This parameter is available only when the **Advanced Settings** check box is selected.
Node Chinese Tips	The tips that help users write code in the code editor. The tips are planned to appear in the upper-right corner of the code editor. The tips can be up to 256 characters in length.
Node English Tips	

6. In the **Interaction** section, set relevant parameters.

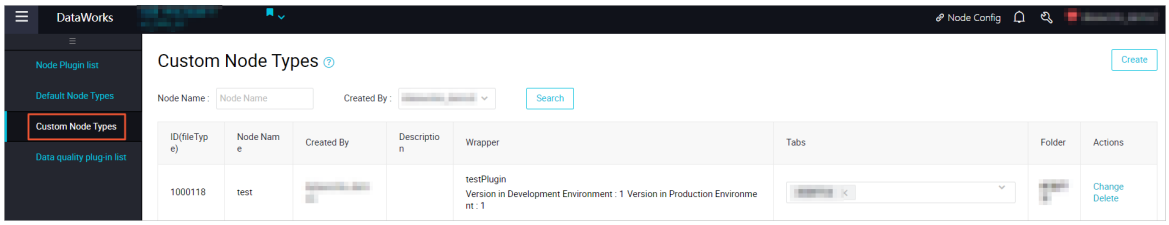


Parameter	Description
Shortcut Menu	○ The options to appear in the shortcut menu. By default, the following operations are selected: Rename, Move, Clone, Steal Lock, View Versions, Locate in Operation Center, Delete, and Submit for Review. ○ More options include Edit, Copy Resource Name, and Send to DataWorks Desktop (Shortcut).

Parameter	Description
Tool Bar	○ The options to appear in the toolbar. By default, the following options are selected: Save, Commit, Commit and Unlock, Steal Lock, Run, Show/Hide, Run with Arguments, Stop, Reload, Run Smoke Test in Development Environment, View Smoke Test Log in Development Environment, Run Smoke Test, View Smoke Test Log, Go to Operation Center of Development Environment, and Format. ○ More options include Operation Center, Deploy, and Precompile.
Right-Side Bar	○ The tabs to appear on the right-side navigation pane of the code editor. By default, the Scheduling configuration and Structure options are selected. ○ More options include Version, Lineage, and Parameters.
Auto Parse Option	Specifies whether to display the Automatic parsing parameter for this type of nodes. If you turn on this switch, the Automatic parsing parameter is displayed on the Scheduling configuration tab of this type of nodes. Otherwise, this parameter is not displayed. In an automatic parsing process, the system parses the input and output of a node based on the lineage that is specified in the code.

7. Click **Save and Exit**. The custom node type appears on the **Custom Node Types** tab.

   In the **Tabs** column, you can select one or more services where the custom node type can be used based on your needs. Valid values are **Data Development**, **Manual business process**, and **Temporary query**.



   In the Actions column, you can click **Change** or **Delete** to edit or delete the custom node type. Only the owner of the current workspace or the creator of the custom node type can perform these operations.

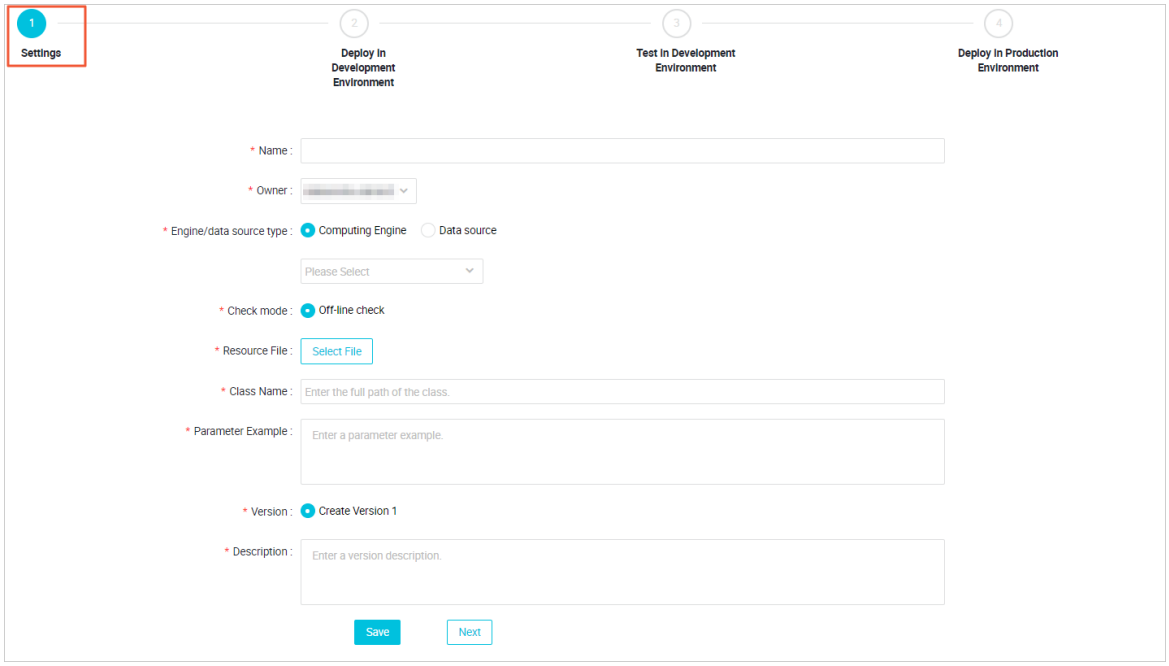
   - **Change**: You can click **Change** to go to the Configure Custom Node Type page. On this page, you can modify the configurations of the custom node type based on your needs.
   - **Delete**: If no nodes of this custom node type are in the Deployed state, you can delete the custom node type. If a node of this custom node type is in the Deployed state, an error message appears after you click Delete. You must undeploy the node before you delete the custom node type.

5.12.1.5. Create a data quality wrapper

DataWorks allows you to create, deploy, and use data quality wrappers to satisfy diversified needs for data quality.

Procedure

1. Create a data quality wrapper.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. After you select the region where the required workspace resides, find the workspace and click **Data Analytics**.
   iv. Click **Node Config** in the upper-right corner. The **Node Plugin list** page appears.
   v. In the left-side navigation pane, click **Data quality plug-in list**.
   vi. On the **Data quality plug-in list** page, click **Create** in the upper-right corner.
2. In the **Settings** step, set the parameters as required.



Parameter	Description
Name	The name of the data quality wrapper. **Note** The wrapper name must start with a letter and can contain letters, underscores (_), and digits.
Owner	The owner of the wrapper. Select an owner from the drop-down list.
Engine/data source type	You must first select **Computing Engine** or **Data source** and then select a compute engine or a connection type from the drop-down list.
Check mode	Set the value to **Off-line check**.

Parameter	Description
Resource File	You can upload a file or use an Object Storage Service (OSS) object to specify a resource file. Click **Select File**. In the **Select File** dialog box, select a method, complete corresponding configurations, and then click **OK**.
Class Name	The full path of the class that is used to implement the data quality wrapper.
Parameter Example	The parameters that are designed based on the specified resource file.
Version	When you create a wrapper, select Create Version. When you are editing or rolling back a version, select Overwrite Version.
Description	The description of the wrapper.

3. Click **Save** and then **Next**.
4. In the **Deploy in Development Environment** step, confirm the configurations and click **Deploy in Development Environment**. The real-time deployment progress appears.
5. Wait until the deployment is completed and click **Next**.
6. Click **Next**. In the **Deploy in Development Environment** step, confirm the configurations and click **Deploy in Development Environment**.

   Wait until the message **The wrapper is deployed in the development environment** appears. Click **Next**. The **Test in Development Environment** step appears.
7. Test the wrapper in the development environment.
   i. In the **Test in Development Environment** step, set the parameters on the left.

Parameter	Description
Sampling method	The method to be used for data sampling.
Storage Engine/data source	The compute engine or data store where the data to be sampled resides.
Table GUID	The globally unique identifier (GUID) of the table. Example: database.table.
Partition name	The name of the partition.
Field name	The name of the field.
Filter condition	The filter condition. Enter the condition that is specified by the WHERE clause. Do not enter the WHERE keyword.
Custom SQL	A custom SQL statement.

   ii. Click **Test**.
   iii. Confirm the test results and select **Test Passed**.

iv. Click **Next**.

8. In the **Deploy in Production Environment** step, click **Deploy in Production Environment**. The real-time deployment progress appears.
9. Click **Complete**. The **Wrappers** page appears.

   The created wrapper appears on this page. You can configure the wrapper, view the version information about the wrapper, or delete the wrapper.

   - Click **Settings**. You are navigated to a step in the configuration wizard of the wrapper. The step you are navigated to appears based on the existing configurations of the wrapper.
   - Click **View Versions**. In the **View Versions** dialog box, you can view, roll back, or download each version. Then, click **OK**.
     - Click **View**. In the Settings step, view the basic information about the current version.
     - Click **Roll Back**. In the **Roll Back** message, click **OK**. The wrapper is rolled back to the previous version.
     - Click **Download** to download the resource file of the current version.
   - Click **Delete**. In the **Delete Wrapper** message, click **OK**.

5.12.2. Create a Hologres Development node

You can create and edit Hologres Development nodes and update the node versions.

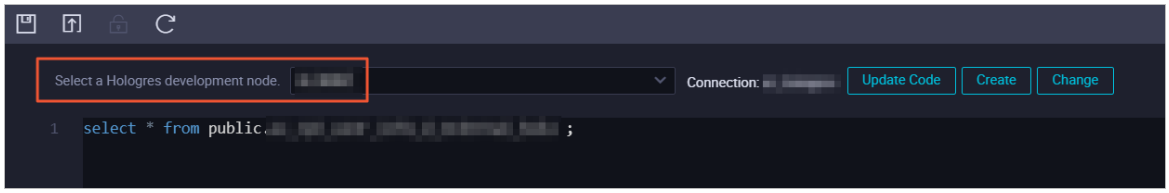
Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the DataStudio page, move the pointer over the **Create** icon and choose **Custom > Hologres Development**.

   Alternatively, you can open the desired workflow, right-click **UserDefined**, and then choose **Create > Hologres Development**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.
5. On the configuration tab of the **Hologres Development** node, select a Hologres Development node that is developed in HoloStudio from the **Select a Hologres development node** drop-down list.



If no such node is available, click **Create** to create one. You can also click **Change** to modify the code of the Hologres Development node that is developed in HoloStudio.

> **Note** A Hologres Development node that is developed in HoloStudio is associated with a Hologres Development node that is created in DataStudio. If the Hologres Development node that you select already has an associated Hologres Development node that is created in DataStudio, the system reports an error.

6. On the configuration tab of the node, click **Properties** in the right-side navigation pane. On the Properties tab, configure properties for the node. For more information, see Configure basic properties.
7. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

   i. Click the icon in the toolbar to save the node.

   ii. Click the icon in the toolbar.

   iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

   iv. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.
8. Test the node. For more information, see View auto triggered nodes.

5.12.3. Create a Data Lake Analytics node

Data Lake Analytics nodes are supported in DataWorks. You can create a Data Lake Analytics node in the DataWorks console to build an online extract, transform, and load (ETL) process.

Context

Data Lake Analytics nodes are used to connect to Data Lake Analytics (DLA), an interactive query and analytics service that is provided by Alibaba Cloud. For more information, see DLA documentation.

> **Notice** You can run Data Lake Analytics nodes only on exclusive resource groups for scheduling. For more information, see Create and use an exclusive resource group for scheduling.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.

ii. In the left-side navigation pane, click **Workspaces**.

iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.

2. On the Data Development tab, move the pointer over Create and choose **Customize > Data Lake Analytics**.

   Alternatively, you can click the target workflow, right-click **UserDefined**, and then choose **New > Data Lake Analytics**.

3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

4. Click **Commit**.

5. Configure the Data Lake Analytics node.

   i. Select a connection.

   On the configuration tab of the Data Lake Analytics node, select a connection from the Select data source drop-down list. If you cannot find the required connection in the drop-down list, click **New data source** to add a connection on the **Data Source** page. For more information, see Add a DLA data source.

   ii. Write the SQL statements of the node.

   After you select a connection, write SQL statements based on the syntax that is supported by DLA. You can write data manipulation language (DML) or data definition language (DDL) statements in the code editor.

   iii. Click in the toolbar.

   iv. Click in the toolbar to run the SQL statements you have saved.

   If you need to change the resource group on which you test the Data Lake Analytics node on the **DataStudio** page, click in the toolbar and select your desired exclusive resource group.

   > **Note** To access a data store in a virtual private cloud (VPC), a node must be run on an exclusive resource group for scheduling. In this example, the data store is in a VPC. You must select an exclusive resource group for scheduling that is connected to the target DLA data store.

6. Click the **Scheduling configuration** tab in the right-side navigation pane. On the Scheduling configuration tab, set the scheduling properties for the node. For more information, see Configure basic properties.

   You must select an exclusive resource group for scheduling that is connected to the target DLA data store to periodically run the node.

7. Save and commit the node.

   > **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

i. Click the icon in the toolbar to save the node.

ii. Click the icon in the toolbar.

iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iv. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.

8. Test the node. For more information, see View auto triggered nodes.

5.12.4. Create an AnalyticDB for MySQL node

You can create an AnalyticDB for MySQL node in the DataWorks console to build an online extract, transform, and load (ETL) process.

Context

AnalyticDB for MySQL nodes are used to connect to AnalyticDB for MySQL of Alibaba Cloud. For more information, see AnalyticDB for MySQL documentation.

> **Note** You can run AnalyticDB for MySQL nodes only on exclusive resource groups for scheduling. For more information, see Create and use an exclusive resource group for scheduling.

Procedure

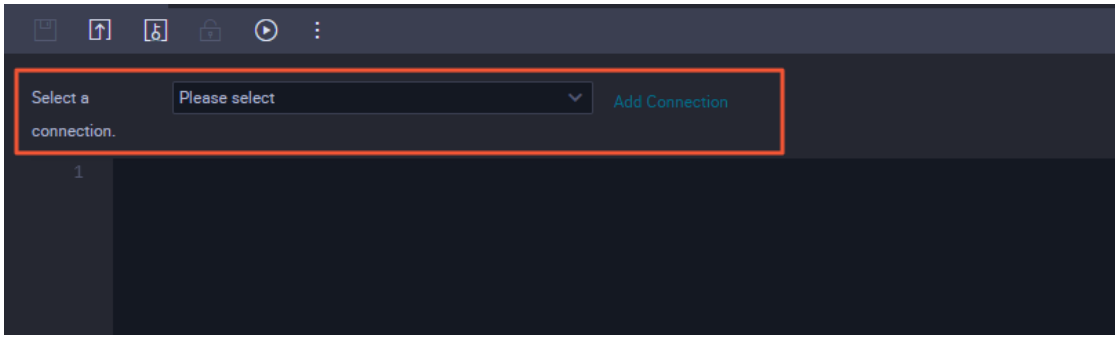
1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Move the pointer over the Create icon and choose **Custom > AnalyticDB for MySQL**.

   Alternatively, you can click the required workflow, right-click **UserDefined**, and then choose **Create > AnalyticDB for MySQL**.
3. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
4. Click **Commit**.
5. Configure the AnalyticDB for MySQL node.

i. Select a connection.

Select a connection for the node. If you cannot find the required connection in the drop-down list, click **Add Connection** and create a connection on the **Data Source** page. For more information, see Supported data stores and plug-ins.



ii. Write the SQL statements of the node.

After you select a connection, write SQL statements based on the syntax that is supported by AnalyticDB for MySQL. You can write data manipulation language (DML) or data definition language (DDL) statements.

iii. Click the icon in the toolbar to save the SQL statements to the server.

iv. Click the icon in the toolbar to execute the SQL statements you have saved.

When you run the node for the first time, the **Parameters** dialog box appears. You must select a resource group for running the node from the **Scheduling Resource Group** drop-down list, set other parameters as required, and then click **Confirm**.

When you run the node later, the system uses the resource group and parameter settings that you specify for the first running of the node. If you need to change the resource group or modify the parameter settings, click the icon in the toolbar.

> **Note** To access a data store in a virtual private cloud (VPC), a node must be run on an exclusive resource group for scheduling. In this example, you must select an exclusive resource group for scheduling that is connected to the AnalyticDB for PostgreSQL instance.

6. Click the **Properties** tab in the right-side navigation pane and set the scheduling properties for the node. For more information, see Configure basic properties.

   You must select an exclusive resource group for scheduling that is connected to the specified AnalyticDB for MySQL data store to periodically run the node.

7. Save and commit the node.

> **Notice** You must set the **Rerun** and **Parent Nodes** parameters before you can commit the node.

i. Click the icon in the toolbar to save the node.

ii. Click the icon in the toolbar.

iii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.

iv. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Deploy nodes.

8. Test the node. For more information, see View auto triggered nodes.

5.13. Node management

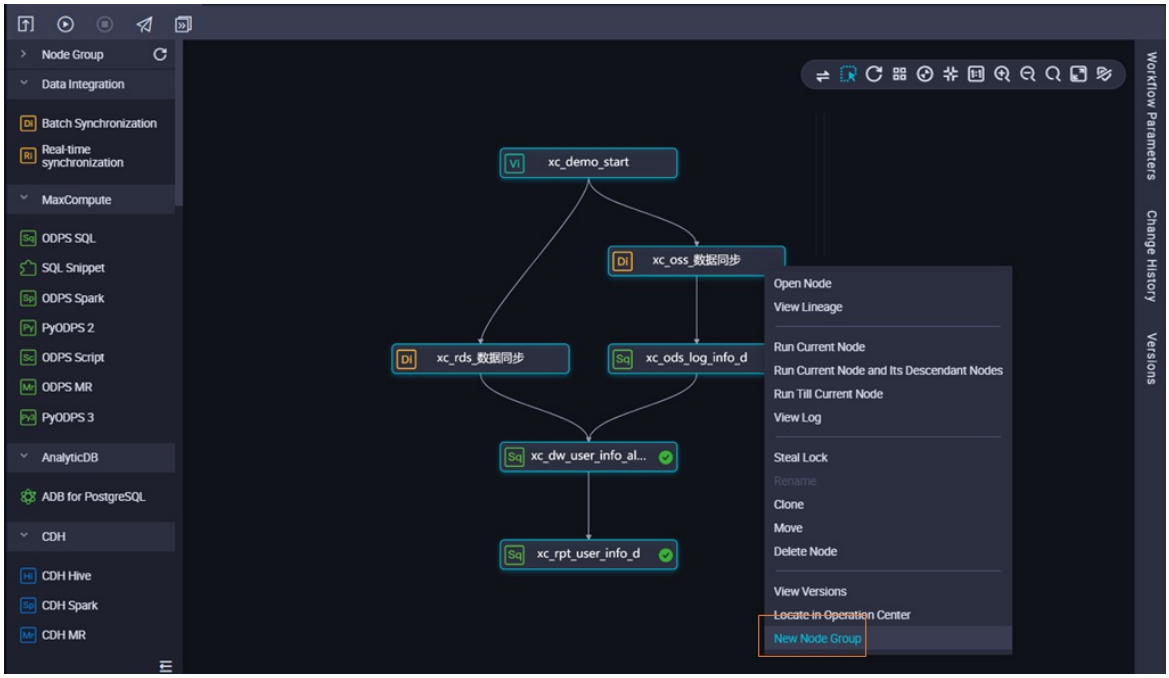
5.13.1. Create and reference a node group

You can group nodes that are frequently reused in a workflow as a node group. Then, you can reference the node group in other workflows to reuse these nodes. The configuration of each node remains unchanged after the nodes are added to a node group. This topic describes how to create and reference a node group.

Create a node group

1. On the **DataStudio** page, create a workflow. For more information, see Manage workflows.
2. Go to the workflow configuration tab. Click the Box icon in the upper-right corner and drag a box to select the nodes to be included in a node group.

3. Right-click a node among the selected nodes and select **New Node Group**.



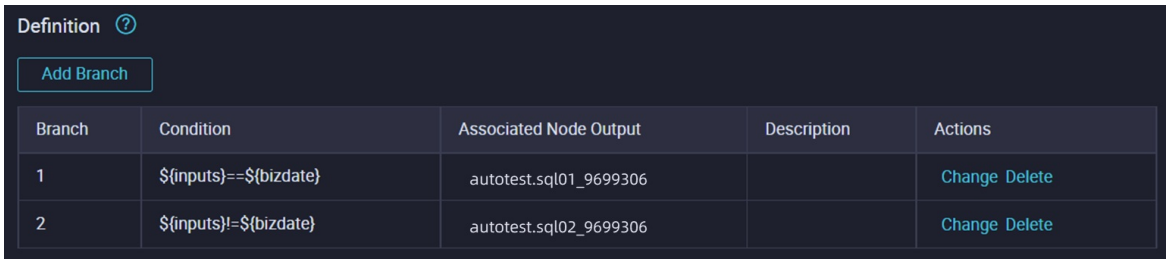
4. In the **New Node Group** dialog box, enter a name in the **Name** field and click **OK**.

5. Right-click the node group and select **Save node group**. In the dialog box that appears, click OK. Then, you can view the created node group in the Node Group section.

Item in the shortcut menu	Description
Save node group	Save the node group. The node group that you have created appears in the Node Group section only after you click **Save node group**. A node group that is not saved cannot be referenced in other workflows.

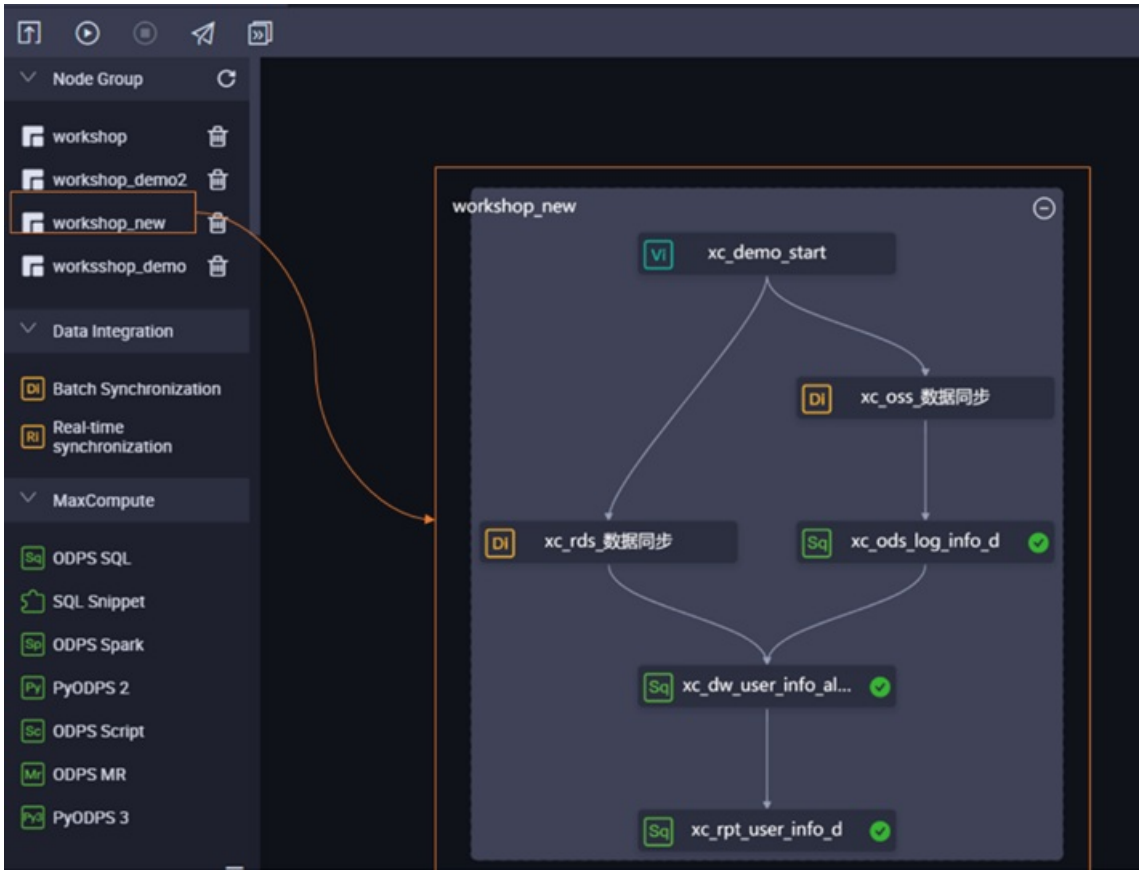
Item in the shortcut menu	Description
Delete node group	You can click **Delete Node Group** **Note** to delete the nodes in the node group.
Split node group	Dismiss the node group. After the node group is dismissed, the selected nodes no longer form a node group in the workflow. However, the node group still exists in the node group list.

If the created node group contains a Machine Learning experiment node, create a Machine Learning Platform for AI (PAI) experiment in another workflow to reference the node group. If the created node group contains a branch node, add digits to the value in the **Associated Node Output** parameter.



Reference a node group

You can directly drag a node group to another workflow to reference the node group in the workflow. The dependencies among the nodes in the node group remain unchanged.



You can run the workflow or commit and deploy the workflow. Then, go to **Operation Center** to view the execution result.

5.13.2. Recycle bin

DataWorks provides a recycle bin to store all deleted nodes in the current workspace. You can restore or permanently delete the nodes.

In the left-side navigation pane, click **Recycle Bin** to go to the Recycle Bin page.

In the recycle bin, you can view all deleted nodes in the current workspace. You can right-click a node to restore or permanently delete it.

> **Note**
> - The recycle bin displays only 100 nodes. If more than 100 nodes are deleted, the nodes deleted earlier are deleted permanently from the recycle bin.
> - Deleted node groups are not displayed in the recycle bin.

Click **Show My Nodes Only** in the upper-right corner to view the deleted nodes in the current workspace.

> **Note** Nodes permanently deleted from the recycle bin cannot be restored. The recycle bin can only store deleted nodes, but not deleted workflows, tables, or resources.

5.13.3. Script template management

5.13.3.1. Create a script template

This topic describes the definition of a script template, the content of a script template, and how to create a script template.

Limits

You can use only DataWorks Standard Edition or a more advanced edition to create a script template. You must activate DataWorks Standard Edition or a more advanced edition before you create a script template. For more information, see Billing of DataWorks advanced editions.

Definition

A script template defines an SQL code process that includes multiple input and output parameters. Each SQL code process references one or more source tables. You can use an SQL code process to filter source table data, join source tables, and aggregate the source tables to generate a new result table required for business.

Value

In actual business scenarios, a large number of SQL code processes are similar. The input tables or output tables in these processes may have the same schema or compatible data types but different table names. In this case, developers can abstract an SQL code process as a script template to reuse the SQL code. The script template extracts input parameters from input tables and output parameters from output tables.

To create an SQL script template, you need to only select an existing script template from the script template list based on your business process and configure specific parameters for input tables and output tables in your business for the selected script template. This way, you do not need to repeatedly copy the code. This helps improve the development efficiency and avoid repeated operations during development. You can deploy and run the SQL script templates that are created in the same manner in which you deploy and run other SQL nodes.

Template contents

Similar to a function, a script template consists of input parameters, output parameters, and an SQL code process.

Input parameters

The input parameters of a script template include properties such as the parameter name, parameter type, parameter description, and parameter definition. The parameter type can be table or string.

- A table-type parameter specifies the table that you want to reference in an SQL code process. When you use a script template, you can specify the input table that is required for the specific business.
- A string-type parameter specifies the variable control parameter in an SQL code process. For example, to export only the sales amount of the top N cities in each region in a result table of an SQL code process, you can use a string-type parameter to specify the value of N.

  To export the total sales amount of a province in a result table of an SQL code process, you can use a string-type parameter to specify the name of the province and obtain the sales data of the specified province.

- The parameter description specifies the functionality of a parameter in an SQL code process.
- The parameter definition is a text definition of the table schema and is required only for table-type parameters. When you specify the parameter definition for a table-type parameter, specify an input table that contains the same field names and compatible filed types that are defined by using the table-type parameter. This way, the SQL code process can properly run. If you do not specify an input table that contains the same field names and compatible field types that are defined by using the table-type parameter, an error is returned because the specified field name cannot be found in the input table when the SQL code process runs. The input table must contain the field names and field types that are defined by using the table-type parameter. The input table can also contain other fields. The field names and field types in the input table can be specified in an order based on your business requirements. In this topic, the parameter definition is provided only for reference.
- We recommend that you define parameters in the following format:

  ```
  Name of field 1 Type of field 1 Description of field 1
  Name of field 2 Type of field 2 Description of field 2
  Name of field n Type of field n Description of field n
  ```

  Example:

  ```
  area_id string 'Region ID'
  city_id string 'City ID'
  order_amt double 'Order amount'
  ```

Output parameters

- The output parameters of a script template include properties such as the parameter name, parameter type, parameter description, and parameter definition. The parameter type must be table. String-type output parameters are not supported.
- A table-type parameter specifies the table to be generated in an SQL code process. When you use a script template, you can specify the result table that the SQL code process generates for the specific business.
- The parameter description specifies the functionality of a parameter in an SQL code process.
- The parameter definition is a text definition of the table schema. When you specify the parameter definition for a table-type parameter, specify an output table that contains the same number of fields and compatible field types that are defined by using the table-type parameter. This way, the SQL code process can properly run. If you do not specify an output table that contains the same number of fields and compatible field types that are defined by using the table-type parameter, an error is returned because the number of fields does not match the specified value or the field types are incompatible when the SQL code process runs. The field names of the output table do not need to be the same as the field names that are defined by using the table-type parameter. In this topic, the parameter definition is provided only for reference.
- We recommend that you define parameters in the following format:

  ```
  Name of field 1 Type of field 1 Description of field 1
  Name of field 2 Type of field 2 Description of field 2
  Name of field n Type of field n Description of field n
  ```

  Example:

```
area_id string 'Region ID'
city_id string 'City ID'
order_amt double 'Order amount'
rank bigint 'Ranking'
```

SQL code process

The parameters in an SQL code process are referenced in the following format: `@@{Parameter name}`.

By defining an abstract SQL code process, a script template controls and processes an input table based on input parameters to generate an output table with business value.

Before you develop an SQL code process, correctly configure the input and output parameters. In this case, correct SQL code can be generated and run during the process.

Create a script template

1. Log on to the DataWorks console. In the left-side navigation pane, click Workspaces. On the Workspaces page, find the workspace that you want to manage and click **DataStudio** in the Actions column.
2. In the left-side navigation pane of the DataStudio page, click **Snippets**.
3. In the Snippets pane, move the pointer over 

and choose **Create > Snippet**.
4. In the **Create Snippet** dialog box, configure the **Snippet Name**, **Description**, and **Location** parameters.
5. After you complete the configuration, click **Commit**.

Source table schema

The following table describes the schema of a source MySQL table that contains sales data.

Field name	Data type	Description
order_id	varchar	The ID of the order.
report_date	datetime	The date on which the order is generated.
customer_name	varchar	The name of the customer.
order_level	varchar	The level of the order.
order_number	double	The number of orders.
order_amt	double	The amount of the order.
back_point	double	The discount.
shipping_type	varchar	The transportation method.
profit_amt	double	The amount of the profit.
price	double	The unit price.

Field name	Data type	Description
shipping_cost	double	The cost of transportation.
area	varchar	The region.
province	varchar	The province.
city	varchar	The city.
product_type	varchar	The type of the product.
product_sub_type	varchar	The subtype of the product.
product_name	varchar	The name of the product.
product_box	varchar	The packaging of the product.
shipping_date	datetime	The date of transportation.

Business implication

Script template name: get_top_n

This script template uses the specified sales data table as the table-type input parameter, the number of top cities as the string-type input parameter, and the total sales amount of the cities for ranking. You can obtain the rankings of the specified top cities in each region with ease by using this SQL code process.

Script template parameters

Input parameter 1

- Parameter name: myinputtable
- Type: table

Input parameter 2

- Parameter name: topn
- Type: string

Output parameter 3

- Parameter name: myoutput
- Type: table

Parameter definition:

- area_id string
- city_id string
- order_amt double
- rank bigint

You can execute the following statement to create a table to store the sales data of a specified number of top cities:

```
CREATE TABLE IF NOT EXISTS company_sales_top_n
(
area STRING COMMENT 'Region',
city STRING COMMENT 'City',
sales_amount DOUBLE COMMENT 'Sales amount',
rank BIGINT COMMENT 'Ranking'
)
COMMENT 'Company sales rankings'
PARTITIONED BY (pt STRING COMMENT '')
LIFECYCLE 365;
```

Example on how to define an SQL code process

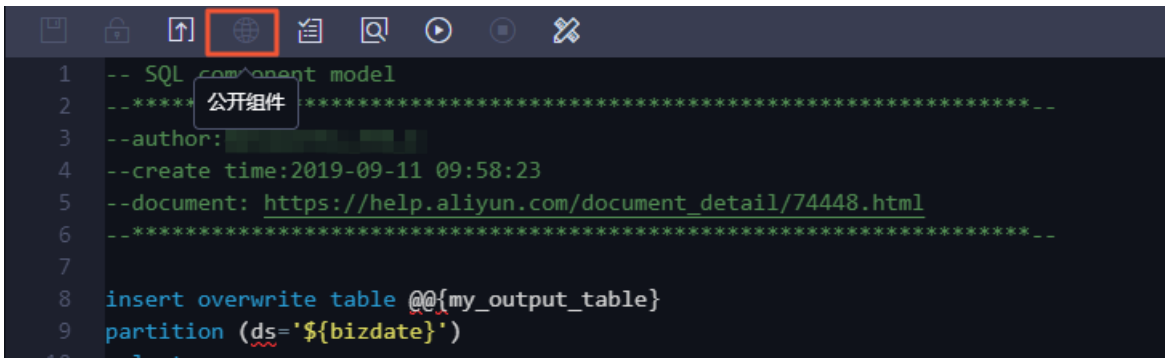
```
INSERT OVERWRITE TABLE @@{myoutput} PARTITION (pt='${bizdate}')
    SELECT r3.area_id,
    r3.city_id,
    r3.order_amt,
    r3.rank
from (
SELECT
    area_id,
    city_id,
    rank,
    order_amt_1505468133993_sum as order_amt ,
    order_number_150546813****_sum,
    profit_amt_15054681****_sum
FROM
    (SELECT
    area_id,
    city_id,
    ROW_NUMBER() OVER (PARTITION BY r1.area_id ORDER BY r1.order_amt_1505468133993_sum DESC
)
AS rank,
    order_amt_15054681****_sum,
    order_number_15054681****sum,
    profit_amt_1505468****_sum
FROM
    (SELECT area AS area_id,
     city AS city_id,
     SUM(order_amt) AS order_amt_1505468****_sum,
     SUM(order_number) AS order_number_15054681****_sum,
     SUM(profit_amt) AS profit_amt_1505468****_sum
FROM
    @@{myinputtable}
WHERE
    SUBSTR(pt, 1, 8) IN ( '${bizdate}' )
GROUP BY
    area,
    city )
    r1 ) r2
WHERE
    r2.rank >= 1 AND r2.rank <= @@{topn}
ORDER BY
    area_id,
    rank limit 10000) r3;
```

Sharing scope

Script templates can be shared within a workspace or made public.

By default, a script template that is deployed can be used by users within the current workspace. The developer of a script template can click the **Publish Snippet** icon to make the general-purpose script template public to the current tenant. This way, all users of the tenant can view and use the script template.

You can check whether the **Publish Snippet** icon that is shown in the following figure is clickable on the configuration tab of a script template. If the icon is clickable, the script template can be made public.

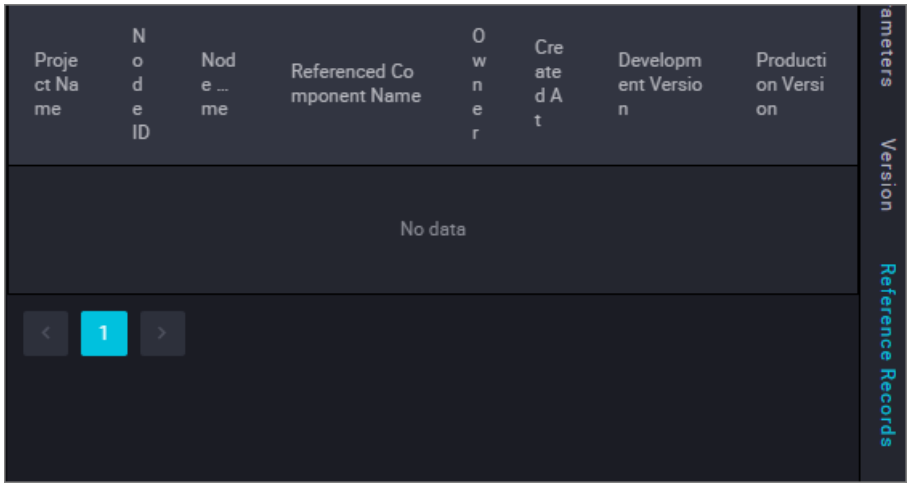


Use a script template

For information about how to use a script template, see Use a script template.

Reference records

In the Components section, double-click a script template. On the configuration tab that appears, click the **Snippet Nodes** tab in the right-side navigation pane to view the reference records of the script template.



5.13.3.2. Use a script template

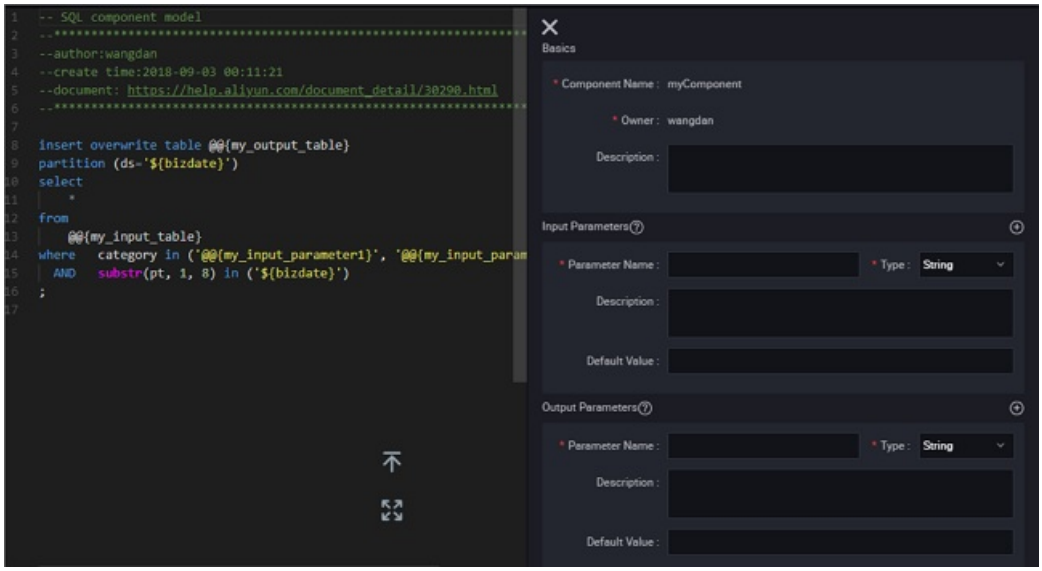
To improve development efficiency, you can create data analytics nodes by using the script templates provided by workspace members and tenants.

Note the following points when using script templates:

- The script templates provided by members of the current workspace are available on the **Workspace-Specific** tab.
- The script templates provided by tenants are available on the **Public** tab.

For more information about how to use script templates, see Create an SQL component node.

Buttons on the interface



No.	Button	Description
1	**Save**	Save the settings of the current script template.
2	**Steal Lock**	Steal the lock of the current script template and then edit it if you are not the owner of the script template.
3	**Submit**	Commit the current script template to the development environment.
4	**Publish Snippet**	Make the current general-purpose script template public to the current tenant account so that all users under the account can view and use the script template.
5	**Parse I/O Parameters**	Parse input and output parameters from the code. **Note** Typically, the parameters entered here are table names instead of scheduling parameters.
6	**Precompile**	Edit custom and system parameters for the current script template.
7	**Run**	Run the current script template in the development environment.
8	**Stop**	Stop running the current script template.
9	**Format Code**	Format the code based on keywords.
10	**Properties**	View the basic information and set input and output parameters for the current script template.
11	**Versions**	View the deployed versions of the current script template.
12	**Snippet Nodes**	List the reference records of the current script template.

6.Create and manage resources and functions

6.1. Create resources and register functions

6.1.1. Create a MaxCompute resource

This topic describes how to create, reference, and download a JAR or Python resource.

Prerequisites

A MaxCompute compute engine instance is associated with the desired workspace. The MaxCompute folder is displayed on the DataStudio page only after you associate a MaxCompute engine instance with the workspace on the **Workspace Management** page. For more information, see Configure a workspace.

Context

If your code or function requires resource files such as .jar files, you can upload resources to your workspace and reference them.

If the existing built-in functions do not meet your requirements, you can create user-defined functions (UDFs) and customize processing logic. You can upload the required JAR packages to your workspace so that you can reference them when you create UDFs.

> Note
> - You can view built-in functions in the **Built-In Functions** pane. For more information, see Functions.
> - You can view the UDFs that you have committed or deployed in DataWorks in the **MaxCompute Functions** pane. For more information, see MaxCompute functions.

You can upload different types of resources, such as text files, Python code, and compressed packages in the .zip, .tgz, .tar.gz, .tar, and .jar formats to MaxCompute. You can read or use these resources when you run UDFs or MapReduce.

MaxCompute provides API operations for you to read and use resources. The following types of resources are supported:

- **Python**: the Python code you have written. You can use Python code to register Python UDFs.
- **JAR**: the compiled Java JAR packages.
- **Archive**: the compressed files that can be identified by the file name extension. Supported file types include .zip, .tgz, .tar.gz, .tar, and .jar.
- **File**: the files in the .zip, .so, or .jar format.

JAR resources and **file** resources have the following differences:

- You can write Java code in an offline Java environment, compress the code to a JAR package, and then upload the package as a JAR resource to DataWorks.

- You can create and edit a small-sized file resource in the DataWorks console.
- If you want to upload a resource file whose size is greater than 500 KB from your on-premises machine when you create a file resource, you can select **Large File (> 500 KB)**.

> **Note** You can upload a resource file whose size is no more than 50 MB. If you want to upload a resource file whose size is greater than 50 MB, you can use the MaxCompute client to perform this operation. Then, commit the resource file to DataWorks in the MaxCompute Resources pane. For more information, see MaxCompute resources.

Create a JAR resource

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the DataStudio page, move the pointer over the Create icon and choose **MaxCompute > Resource > JAR**.

   Alternatively, you can click the name of the desired workflow in the **Business Flow** section, right-click **MaxCompute**, and then choose **Create > Resource > JAR**.

   For more information about how to create a workflow, see Create a workflow.
3. In the **Create Resource** dialog box, configure the **Resource Name** and **Location** parameters.

   > **Note**
   > - If the selected JAR package has not been uploaded from the MaxCompute client, select **Upload to MaxCompute**. If you do not select it, an error occurs during the upload process. If the selected JAR package has been uploaded from the MaxCompute client, clear **Upload to MaxCompute**. If you do not clear it, an error occurs during the upload process.
   > - The resource name can be different from the name of the uploaded file.
   > - The resource name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). The name is not case-sensitive. A JAR resource name must end with .jar, and a Python resource name must end with .py.
4. Click **Upload** and select the file that you want to upload.
5. Click **Create**.
6. Click the icon in the top toolbar to commit the resource to the development environment.

   If the workspace that you use is in standard mode, you must click Deploy in the upper-right corner to deploy the resource after you commit the resource. For more information, see Deploy nodes.

Create a Python resource and register a UDF

1. Create a Python resource.

i. On the **DataStudio** page, move the pointer over the **+Create** icon and choose **MaxCompute > Resource > Python**.

Alternatively, you can click the name of the desired workflow in the **Business Flow** section, right-click **MaxCompute**, and then choose **Create > Resource > Python**.

ii. In the **Create Resource** dialog box, configure the **Resource Name** and **Location** parameters.

> **Notice** The resource name can contain letters, digits, periods (.), underscores (_), and hyphens (-). It must end with .py.

iii. Click **Create**.

iv. Write code for the created Python resource in the code editor. Sample code:

```
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. Click the icon in the top toolbar to commit the resource.

If the workspace that you use is in standard mode, you must click Deploy in the upper-right corner to deploy the resource after you commit the resource. For more information, see Deploy nodes.

2. Register a function.

i. On the **DataStudio** page, move the pointer over the **+Create** icon and choose **MaxCompute > Function**.

Alternatively, you can click the required workflow, right-click **MaxCompute**, and then choose **Create > Function**.

ii. In the **Create Function** dialog box, configure the **Function Name** and **Location** parameters.

iii. Click **Create**.

iv. In the **Register Function** section of the configuration tab that appears, enter the class name of the function in the Class Name field and the name of the Python resource that you created in the Resources field, and then click the icon in the top toolbar. In this example, the class name is `ipint.ipint`.

v. Check whether the ipint function is valid and meets your expectation. You can create an ODPS SQL node on the DataStudio page to test the ipint function by executing an SQL statement.

You can also create an ipint.py file on your on-premises machine and upload the file by using the MaxCompute client. For more information about how to use the MaxCompute client, see MaxCompute client.

```
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.
```

After the resource file is uploaded, you can register a UDF on the MaxCompute client. For more information, see Functions operations. You can use the UDF after it is registered.

Reference and download a resource

- For more information about how to reference a resource in a function, see Functions operations.
- For more information about how to reference a resource in a node, see Create an ODPS MR node.

If you want to download a resource, find the desired resource in the **Resource** folder, right-click the resource name, and then select View Versions. In the Versions dialog box, click **Download**. For more information about how to download a resource by using the MaxCompute client, see Resource operations.

Other operations

After you create a resource, you can rename, reference, or delete the resource in the **MaxCompute** folder of the desired workflow. For more information about how to delete a resource, see DataStudio.

6.1.2. Create and use an EMR JAR resource

DataWorks allows you to create E-MapReduce (EMR) Java Archive (JAR) resources in the DataWorks console. You can upload a JAR file that contains user-defined functions (UDFs) or open source MapReduce code as an EMR JAR resource. Then, you can reference the resource in compute nodes such as an EMR MR node. This topic describes how to create an EMR JAR resource by uploading a file, commit the resource, and reference the resource in compute nodes such as an EMR MR node.

Prerequisites

- An Alibaba Cloud EMR cluster is created, and an inbound rule that contains the following content is added to the security group to which the cluster belongs.
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- An EMR compute engine instance is associated with your workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Configure a workspace.
- If you integrate Hive with Ranger in EMR, you must modify whitelist configurations and restart Hive before you develop EMR Hive nodes in DataWorks. Otherwise, the error message **Cannot modify spark.yarn.queue at runtime** or **Cannot modify SKYNET_BIZDATE at runtime** is returned when you run EMR Hive nodes.
  i. You can modify the whitelist configurations by using custom parameters in EMR. You can append key-value pairs to the value of a custom parameter. In this example, the custom parameter for Hive components is used. The following code provides an example:

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

> **Note** In the preceding code, `ALISA.*` and `SKYNET.*` are specific to DataWorks.

ii. After the whitelist configurations are modified, you must restart the Hive service to make the configurations take effect. For more information, see Restart a service.

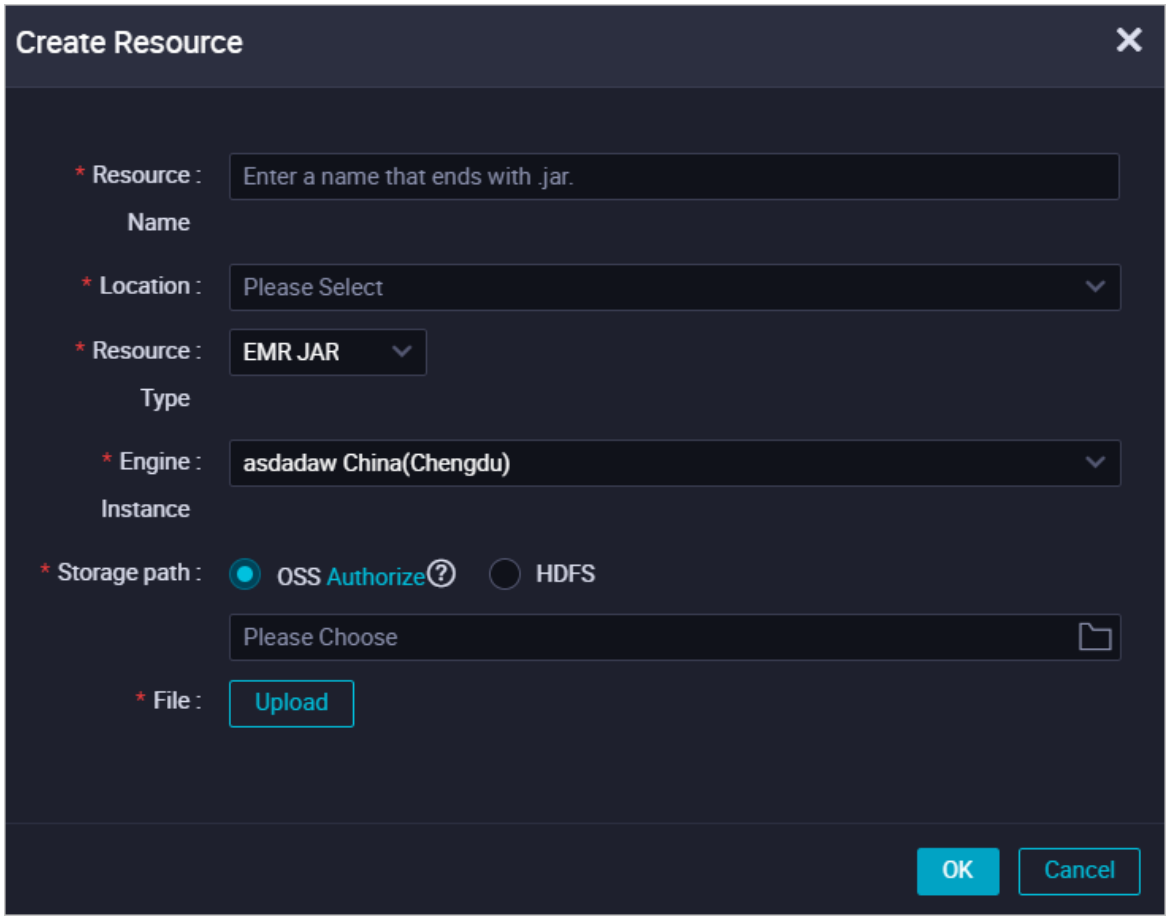
- An exclusive resource group for scheduling is created, and the resource group is associated with the virtual private cloud (VPC) where the EMR cluster resides. For more information, see Create and use an exclusive resource group for scheduling.

> **Note** You can use only exclusive resource groups for scheduling to run EMR Hive nodes.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the DataStudio page, move the pointer over the **Create** icon and choose **EMR > Resource > EMR JAR**.

   You can also find the workflow in which you want to create an EMR JAR resource, right-click the workflow name, and then choose **Create > EMR > Resource** > EMR JAR.
3. In the **Create Resource** dialog box, set the parameters.



Parameter	Description
Resource Name	The name of the resource that you want to create. The resource name must have the suffix .jar.
Location	The folder for storing the resource. The default value is the path of the current folder. You can modify the path based on your business requirements.
File Type	The type of the resource. Set the value to EMR JAR.
Engine Instance	The EMR compute engine instance to which the resource belongs. Select an instance from the drop-down list.
Storage path	The storage path of the resource. Valid values: **OSS** and **HDFS**. ◦ If you select **OSS**, you must click Authorize next to OSS to authorize DataWorks and EMR to access Object Storage Service (OSS). Then, select a folder. ◦ If you select **HDFS**, enter a storage path.
File	The file that you want to upload. You can click **Upload**, select a file from your on-premises machine, and then click **Open**.

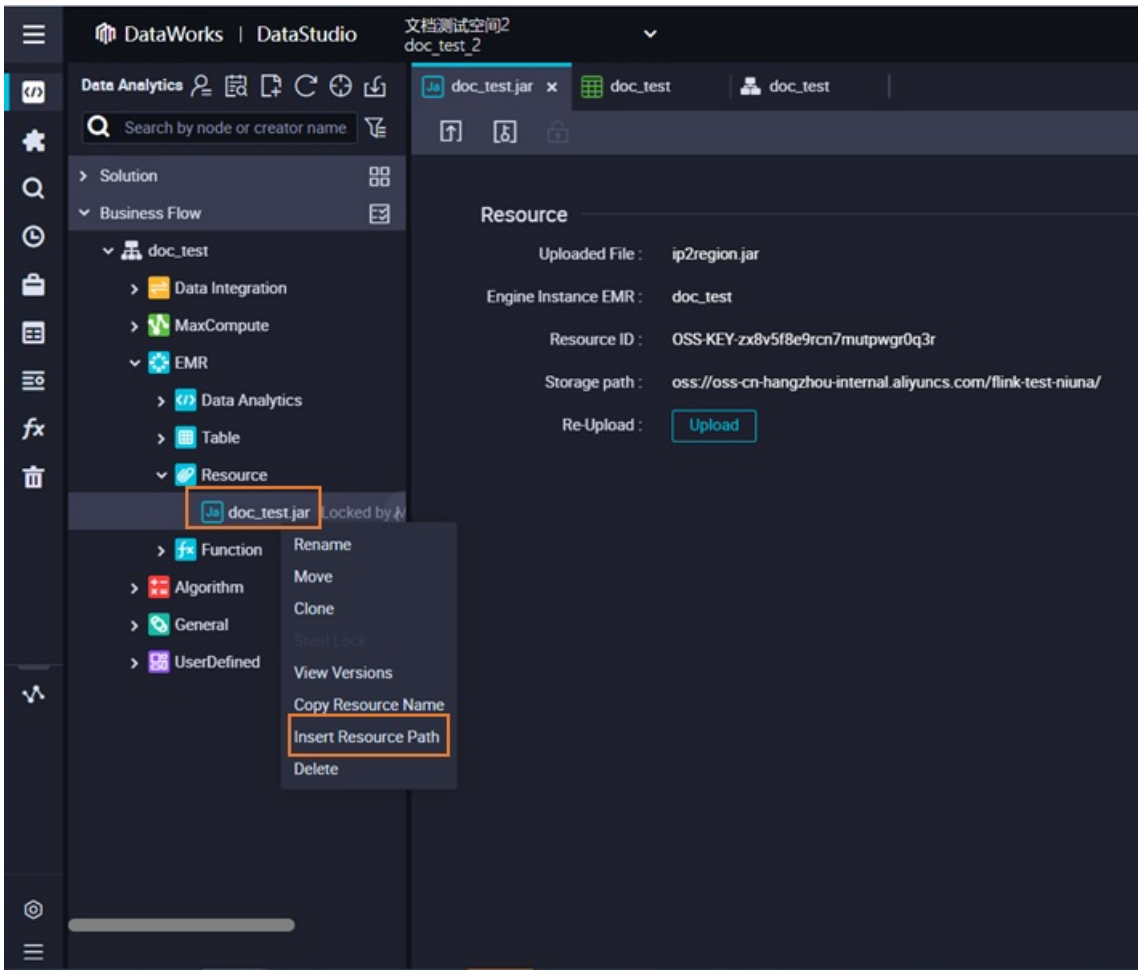
4. Click **Create**.

5. Click the and icons in the top toolbar to save and commit the resource to the development environment.

> **Note**
>
> - You must select a resource group for scheduling when you commit the EMR JAR resource. We recommend that you use an exclusive resource group for scheduling. If no exclusive resource groups for scheduling are available, you can purchase and configure one. For more information, see Create and use an exclusive resource group for scheduling.
> - When you commit the resource by using an exclusive resource group for scheduling, you can view the logs generated during the commission process in the log area of the page. After the resource is committed, messages are displayed in the log area and on the page to inform you of the result.

What's next

After you create an EMR JAR resource, you can reference the resource in the code of compute nodes such as an EMR MR node. The following figure shows how to reference the resource. For more information, see Create and use an EMR MR node.



6.1.3. Register a MaxCompute function

DataWorks supports Python and Java APIs. This topic describes how to register a function.

Prerequisites

The required resources are uploaded.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Create a workflow. For more information, see Create a workflow.
3. Create a Java ARchive (JAR) or Python resource, and commit and deploy the resource. For more information, see Create a MaxCompute resource.
4. Create a function.
   i. Expand the desired workflow, right-click **MaxCompute**, and then choose **Create > Function**.
   ii. In the **Create Function** dialog box, configure the **Function Name** and **Location** parameters.
   iii. Click **Create**.
   iv. In the **Register Function** section of the configuration tab that appears, configure the parameters that are described in the following table.

Parameter	Description
Function Type	The type of the function. Valid values: **Mathematical Operation Functions**, **Aggregate Functions**, **String Processing Functions**, **Date Functions**, **Window Functions**, and **Other Functions**.
Engine Instance MaxCompute	The MaxCompute compute engine instance. The value of this parameter cannot be changed.

Parameter	Description
Function Name	The name of the user-defined function (UDF). You can use this name to reference the UDF in SQL statements. The UDF name must be globally unique and cannot be changed after the UDF is registered.
Owner	The owner of the UDF. The account that is used to log on to the DataWorks console is automatically displayed. You can change the value of this parameter.
Class Name	The name of the class that implements the UDF. The name is in the format of `Resource name.Class name`. The resource name can be the name of a Java or Python package. In DataWorks, you can reference MaxCompute resources including JAR and Python packages to create UDFs. The value format of this parameter varies based on the resource type: ■ If the resource type is JAR, set the **Class Name** parameter in the `JAR package name.Actual class name` format. You can query the class name by executing the `copy reference` statement in IntelliJ IDEA. For example, if `com.aliyun.odps.examples.udf` is the Java package name and `UDAFExample` is the class name, the value of the **Class Name** parameter is `com.aliyun.odps.examples.udf.UDAFExample`. ■ If the resource type is Python, set the **Class Name** parameter in the `Python resource name.Actual class name` format. For example, if `LcLognormDist_sh` is the Python resource name and `LcLognormDist_sh` is the class name, the value of the **Class Name** parameter is `LcLognormDist_sh.LcLognormDist_sh`. **Note** ■ You do not need to include the .jar or .py suffix in the resource name. ■ You can use a resource after the resource is committed and deployed. For more information about how to create a MaxCompute resource, see Create a MaxCompute resource.

Parameter	Description
Resources	Required. You can perform a fuzzy match to search for existing resources in the current workspace. **Note** ■ You do not need to specify the path of the added resources. ■ If multiple resources are referenced in the UDF, separate them with commas (,).
Description	The description of the UDF.
Expression Syntax	The syntax of the UDF. Example: `test`.
Parameter Description	The description of the input and output parameters that are supported.
Return Value	Optional. The return value. Example: 1.
Example	Optional. The example of the UDF.

5. Click the icon in the top toolbar.
6. Commit the UDF.
   i. Click the icon in the top toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iii. Click **OK**.

6.1.4. Create an EMR function

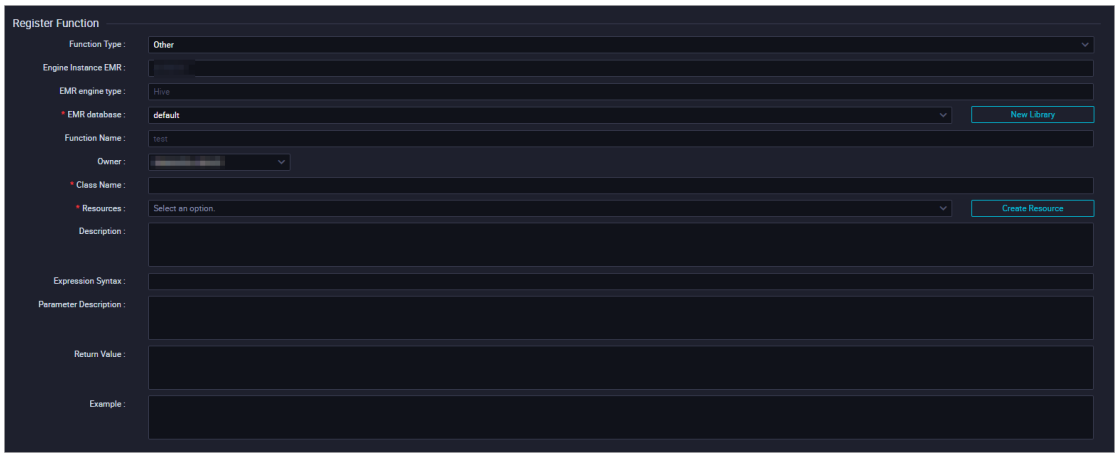
This topic describes how to create an E-MapReduce (EMR) function.

Prerequisites

- An Alibaba Cloud EMR cluster is created, and an inbound rule that contains the following content is added to the security group to which the cluster belongs.
  - Action: Allow
  - Protocol type: Custom TCP
  - Port range: 8898/8898
  - Authorization object: 100.104.0.0/16
- An EMR compute engine instance is associated with your workspace. The EMR folder is displayed only after you associate an EMR compute engine instance with the workspace on the Workspace Management page. For more information, see Configure a workspace.
- The required resources are uploaded.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Create a workflow. For more information, see Create a workflow.
3. Write Java code in an offline Java environment, compress the code to a JAR package, and then upload the package as a JAR resource to DataWorks. For more information, see Create and use an EMR JAR resource.
4. Create a function.
   i. Click the workflow in the Business Flow section, right-click **EMR**, and then choose **Create > Function**.
   ii. In the **Create Function** dialog box, set the **Function Name**, **Engine Instance**, and **Location** parameters.
   iii. Click **Create**.
   iv. In the **Function information** section of the configuration tab that appears, set the parameters.



Parameter	Description
Function Type	The type of the function. Valid values: **Mathematical Operation Functions**, **Aggregate Functions**, **String Processing Functions**, **Date Functions**, **Window Functions**, and **Other Functions**.
Engine Instance	The EMR cluster that is associated with the current workspace. By default, you cannot change the engine instance.
Engine Type	The type of the compute engine. By default, you cannot change the engine type.
EMR database	The database where the EMR cluster resides. Select a database from the drop-down list. To create a database, click **New Library**. In the **New Library** dialog box, set the parameters and click **OK**.

Parameter	Description
Function Name	The name of the function. You can use this name to reference the function in SQL statements. The function name must be globally unique and cannot be changed after the function is created.
Owner	This parameter is automatically set.
Class Name	Required. The name of the class that implements the function.
Resource	Required. The resource to be used in the function. Select a resource from the ones that are created in the current workspace from the drop-down list. To create a resource, click **Create Resource**. In the **Create Resource** dialog box, set the parameters and click **Create**.
Description	The description of the function.
Expression Syntax	The syntax of the function. Example: `test`.
Parameter Description	The description of the input and output parameters that are supported.
Return Value	Optional. The return value. Example: 1.
Example	Optional. The example of the function.

5. Click the icon in the top toolbar.
6. Commit the function.
   i. Click the icon in the top toolbar.

   > **Note** You must select a resource group for scheduling when you commit the EMR function. We recommend that you use an exclusive resource group for scheduling. If no exclusive resource groups for scheduling are available, you can purchase and configure one. For more information, see Create and use an exclusive resource group for scheduling.

   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iii. Click **OK**.
7. Commit the UDF.
   i. Click the icon in the top toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iii. Click **OK**.

6.2. Manage MaxCompute resources and functions

6.2.1. Functions

The Built-In Functions page displays functions built in MaxCompute. You can view the types, descriptions, and examples of functions on this page.

1. Log on to the DataWorks console. In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
2. In the left-side navigation pane, click **Built-In Functions** to view the functions.

   The built-in functions include Aggregate functions, Window functions, Date functions, Mathematical functions, String functions, and Other functions.

   The preceding functions are built in MaxCompute. You can click a function to view its description.

6.2.2. MaxCompute modules

MaxCompute modules include MaxCompute resources and MaxCompute functions. This topic describes how to add and delete MaxCompute modules.

Context

MaxCompute modules are hidden by default. To show MaxCompute modules in the left-side navigation pane, click **Setup** and then the **Configuration Center** tab. In the DataStudio Tabs section, click MaxCompute and then required modules under All Modules.

Procedure

1. Log on to the DataWorks console. In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
2. Click  in the lower-left corner to go to the **Setup** page. Click the **Configuration Center** tab.
3. Click **MaxCompute** in the **DataStudio Tabs** section and view modules under **Tabs on DataStudio Page** and **All Modules**.
4. Click modules under **All Modules** to add them. The added modules are displayed under MaxCompute in the left-side navigation pane of the **DataStudio** page.

   To delete a module, click  next to the module under **Tabs on DataStudio Page**. After you delete the module, it no longer appears in the left-side navigation pane of the **DataStudio** page.

6.2.3. MaxCompute functions

On the MaxCompute Functions tab, you can view functions in the MaxCompute compute engine and the function change history. You can also add functions to workflows on the Data Analytics tab with a few clicks.

Prerequisites

The MaxCompute Functions module is hidden by default. To show this module in the left-side navigation pane of the **DataStudio** page, select the module on the **Configuration Center** tab of the **Setup** page. For more information, see MaxCompute modules.

View functions

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Show **MaxCompute** in the left-side navigation pane and click **MaxCompute Functions**.

   MaxCompute functions are sorted in descending order of creation time by default. Click the icon to switch the order of entries.

   On the MaxCompute Functions tab, you can view the functions that are committed or deployed on the Data Analytics tab. For more information, see Register a MaxCompute function.
3. Click a function to view its details.

   The MaxCompute Functions tab displays functions in the **production environment** by default. To view committed but undeployed functions, click the icon to switch the environment.

   > **Note**
   > - For workspaces in basic mode, only the production environment is available.
   > - Functions that are uploaded by using tools except DataWorks, such as the MaxCompute client and MaxCompute Studio, are not displayed on the **Data Analytics** tab. However, these functions are displayed on the MaxCompute Functions tab.

Delete a function

To delete a function, switch to the **Data Analytics** tab, right-click the name of the function in the required workflow, and then select **Delete**.

Add a function to the Data Analytics tab

1. On the **MaxCompute Functions** tab, click a function to view its details in the lower part.
2. Click **Add to Data Analytics Tab**.

   This operation allows you to synchronize functions on the **MaxCompute Functions** tab to a specified workflow on the **Data Analytics** tab.
3. In the **Create Function** dialog box, set the **Function Name** and **Location** parameters.

   > **Note** When you upload a function, you can rename the function and select a folder to modify the workflow in which the function resides. However, you cannot modify the function definition.
4. Click **OK**.

> Note
>
> - After a function is created, you must manually save, commit, and deploy the function. The operations are the same as those performed when you create a function in a workflow. For more information, see Register a MaxCompute function.
> - When you commit and deploy a function, the function is also uploaded to the development and production environments of MaxCompute. In addition, the function is updated as a custom function on the **MaxCompute Functions** tab.
> - A function is unique in a MaxCompute project. If a function with the same name already exists in the workspace, the newly created function overrides the existing one. If the existing function resides in a different workflow, the function is overridden in the new workflow.

View the function change history

Click the required function and then click **View Change History** to view the creation and change records of the function.

6.2.4. MaxCompute resources

On the MaxCompute Resources tab, you can view resources in the MaxCompute compute engine and the resource change history. You can also add resource files to workflows on the Data Analytics tab with a few clicks.

Prerequisites

The MaxCompute Resources module is hidden by default. To show this module in the left-side navigation pane of the **DataStudio** page, select the module on the **Configuration Center** tab of the **Setup** page. For more information, see MaxCompute modules.

View resources

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Show **MaxCompute** in the left-side navigation submenu and click **MaxCompute Resources**.

No.	Icon	Description

No.	Icon	Description
1		The MaxCompute Resources tab displays resources in the **production environment** by default. To view committed but undeployed resources, click the  icon to switch the environment. **Note** For workspaces in basic mode, only the production environment is available.
2		Click this icon to filter resources by type. **Note** JAR, archive, file, and Python resources are supported.
3		Click this icon to switch the order of entries. Entries are sorted in descending order of update time by default.

3. Click a resource to view its detailed information, including the name, type, and size of the resource.
   - Resources that are displayed on the **MaxCompute Resources** tab may be different from those on the **Data Analytics** tab.
     - When you create a resource on the **Data Analytics** tab, you must perform the following operations to show the resource in the development or production environment of the **MaxCompute Resources** tab: Select **Upload to MaxCompute**. Then, commit or deploy the resource.
     - Resource files that are uploaded by using tools except DataWorks, such as the MaxCompute client and MaxCompute Studio, are not displayed on the **Data Analytics** tab. However, these resources are displayed on the **MaxCompute Resources** tab.
   - The use of resources on the **MaxCompute Resources** tab is different from that of resources on the **Data Analytics** tab.

Scenario	Data Analytics	MaxCompute Resources
Run an ODPS SQL node	Supported. The resources must be uploaded to MaxCompute.	Supported
Run an ODPS MR node	Supported. The resources must be uploaded to MaxCompute.	Not supported
Run a Shell node	Supported	Not supported
Run an ad hoc query	Supported. The resources must be uploaded to MaxCompute.	Supported

Scenario	Data Analytics	MaxCompute Resources
Create a function in a workflow	Supported. The resources must be uploaded to MaxCompute.	Not supported

Add a resource to the Data Analytics tab

1. Find the required resource and click **Add to Data Analytics**.

   This operation allows you to synchronize resource files on the **MaxCompute Resources** tab to a specified workflow on the **Data Analytics** tab.

2. In the **Create Resource** dialog box, set the **Resource Name** and **Location** parameters, click **Upload**, and then select the file to upload.

   You can perform the following operations during the upload:

   - Rename the resource.
   - Select a destination folder to modify the workflow in which the resource resides.

   You cannot perform the following operations during the upload:

   - Change the resource type.
   - Select or clear Upload to MaxCompute.
   - Re-upload the resource file.

3. After the configuration is completed, click **OK**. Then, the resource is created.

> **Note**
>
> - After a resource is created, you must manually save, commit, and deploy the resource. The operations are the same as those performed when you create a resource in a workflow. For more information, see Create a MaxCompute resource.
> - When you commit and deploy a resource, the resource is also uploaded to the development and production environments of MaxCompute. In addition, the resource is displayed on the **MaxCompute Resources** tab.
> - A resource is unique in a MaxCompute project. If a resource with the same name already exists in the workspace, the newly created resource overrides the existing one. If the existing resource resides in a different workflow, the resource is overridden in the new workflow.

View the resource change history

Click the required resource and then click **Change History** to view the creation and change records of the resource file.

7.Code Development and Quality Assurance

7.1. SQL coding guidelines and specifications

This topic describes the basic guidelines and detailed specifications of SQL coding.

Coding guidelines

When you write SQL code, take note of the following guidelines:

- The code is comprehensive.
- Code lines are clear, neat, well-organized, and structured.
- The optimal execution speed is considered during SQL coding.
- Comments need to be added if necessary to enhance the readability of your code.
- The guidelines impose non-mandatory constraints on the coding behavior of developers. In practice, reasonable deviations are allowed if developers obey general rules.
- All SQL keywords and reserved words must be entirely uppercase or lowercase, such as select/SELECT, from/FROM, where/WHERE, and/AND, or/OR, union/UNION, insert/INSERT, delete/DELETE, group/GROUP, having/HAVING, or count/COUNT. Do not use mixed-case letters, such as Select or seLECT.
- One unit of indentation contains four spaces. All indentations must be an integral multiple of one indentation unit. The code is aligned based on its hierarchy.
- The `SELECT *` statement is prohibited. A column name must be specified in all statements.
- Matching opening and closing parentheses must be placed in the same column.

SQL coding specifications

When you write SQL code, take note of the following specifications:

- Code header

  The code header contains information such as the subject, description, author, and date. Reserve a line for change logs and a title line so that users can add change records in the future. Each line can contain a maximum of 80 characters. The following code provides an example template:

```
-- MaxCompute(ODPS) SQL
--********************************************************************
-- ** Subject: Transaction
-- ** Description: Transaction refund analysis
-- ** Author: Youma
-- ** Created on: 20170616
-- ** Change log:
-- ** Modified on  Modified by  Content
-- yyyymmdd name comment
-- 20170831  Wuma  Add a comment on the biz_type=1234 transaction
--********************************************************************
```

- Field arrangement
  - Use a line for each field that is selected for the SELECT statement.
  - Reserve one unit of indentation between the word SELECT and the first selected field.
  - Start each field name in a new line with two units of indentation and a comma (,).
  - Place a comma (,) between two fields right before the second field.
  - Place the AS statement in the same line as its matching field and keep AS statements of multiple fields in the same column.

```
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   (
```

- Clause arrangement for an INSERT statement

  Arrange the clauses of an INSERT statement in the same line.

- Clause arrangement for a SELECT statement

  The clauses such as FROM, WHERE, GROUP BY, HAVING, ORDER BY, JOIN, and UNION in a SELECT statement must be arranged in compliance with the following requirements:

  - Use a line for each clause.
  - The clauses are left aligned with the SELECT statement.
  - Reserve two units of indentation between the first word of a clause and its content.
  - The logical operators and and or in a WHERE clause must be left aligned with the keyword WHERE.
  - If the length of a clause name exceeds two units of indentation, add a space between the clause name, such as ORDER BY and GROUP BY and its content.

```
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)
```

- Spacing before and after operators

Reserve one space before and after each arithmetic operator and logical operator. Keep all the operators in the same line unless the length of the code exceeds 80 characters.

```
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)
```

- Arrangement for a SELECT CASE statement

  The SELECT CASE statement is used to evaluate the value of a variable. Take note of the following specifications on SELECT CASE statements:

  - Write the WHEN clause one unit of indentation after the CASE statement in the same line.
  - Use a single line for each WHEN clause. Wrap the line if the clause is excessively long.

```
,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
```

  - The CASE statement must contain the ELSE clause. The ELSE clause must be aligned with the WHEN clause.

- Nested query

  Nested queries are often used to implement the extract, transform, load (ETL) operations of data warehouse systems. The following figure shows an example of arranging nested queries.

```
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
;
```

- Table alias
  - If an alias is defined for a table in a SELECT statement, you must use the alias whenever you reference the table in the statement. Specify an alias for each table in the SELECT statement.
  - We recommend that you define the table aliases in alphabetical order, and do not use keywords for the aliases.
  - In a nested query, levels 1 to 4 of SQL statements are named part, segment, unit, and detail, which are abbreviated as P, S, U, and D. You can also use a, b, c, and d to represent levels 1 to 4.

    To differentiate multiple clauses at the same level, add numbers such as 1, 2, 3, and 4 after the letter that represents the level. Add comments to the table aliases as needed.

```
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 comments
  - Add a comment to each SQL statement.
  - Use a separate line for the comment of each SQL statement and place the comment before the SQL statement.
  - Place the comment of a field right after the field.
  - Add comments to clauses that are difficult to understand.
  - Add a comment to important code.
  - If a statement is long, we recommend that you add comments based on the purposes of each segment.
  - The description for a constant or variable is required. The comment on the valid value range is optional.

- To add a comment to an SQL statement, place the mouse cursor at the end of the SQL statement and press `Ctrl+/` or `cmd+/`. To comment on multiple lines of code, you can select the code on which you want to comment and press `Ctrl+/` or `cmd+/`.

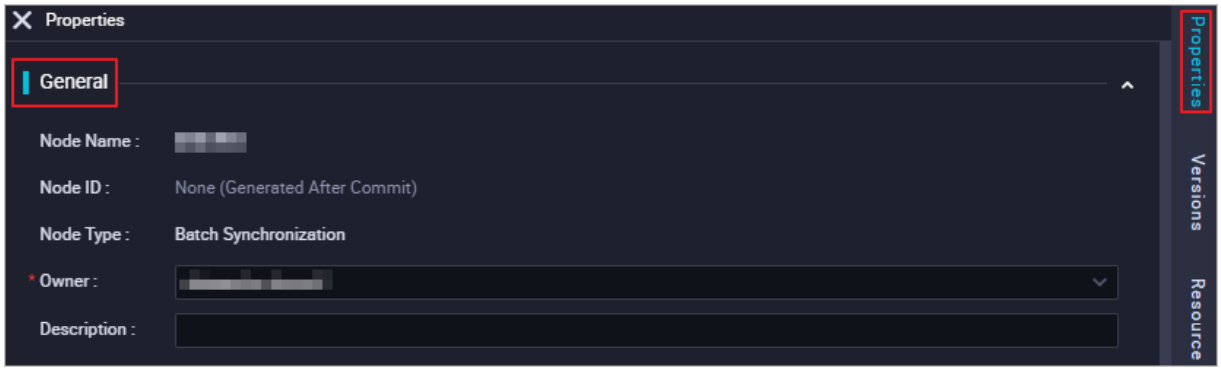
  > **Note**
  > - If your computer runs a Windows operating system, press `Ctrl+/` to add a comment to an SQL statement.
  > - If your computer runs a macOS, press `cmd+/` to add a comment to an SQL statement.

8.Schedule

8.1. Configure basic properties

In the General section of the Properties tab, you can view the basic properties a node, including the name, ID, type, owner, and description. This topic describes the parameters that are used to configure basic properties for a node.

On the DataStudio page, create a node and go to the configuration tab of the node. In the right-side navigation pane of the configuration tab, click the **Properties** tab. In the **General** section of the Properties tab, configure basic properties for the node.



Parameter	Description
Name	The name that you specify for the node when you create the node. The name cannot be changed.
Node ID	The unique ID of the node. The ID is generated when the node is committed for the first time. The ID cannot be changed.
Node Type	The node type that you select when you create the node. The node type cannot be changed.
Owner	The owner of the node. The default value is the current logon user. You can change the owner based on your business requirements. **Note** You can change the owner only to a member of the current DataWorks workspace.
Description	The description of the node. You can specify the business information or usage of the node for this parameter.

8.2. Configure scheduling parameters

8.2.1. Overview of scheduling parameters

In DataWorks, nodes are scheduled to run based on scheduling parameters. Scheduling parameters are automatically replaced with specific values based on the data timestamps of the nodes, the time when the nodes are scheduled to run, and the value formats of the scheduling parameters. This implements dynamic parameter settings for node scheduling. This topic describes the types of scheduling parameters and the details of specific scheduling parameters.

The following sections provide more information about scheduling parameters for you:

- Value types of scheduling parameters: describes the value types of scheduling parameters. The values of scheduling parameters are assigned based on the data timestamps of the nodes and the time when the nodes are scheduled to run.
- Types of scheduling parameters: describes the types of scheduling parameters. Scheduling parameters are classified into **custom parameters** and **built-in variables** based on their value assignment methods.
- Scheduling parameters: describes specific scheduling parameters.

After you understand what scheduling parameters are, you can configure and use them in the **Properties** panel. For more information, see Configure and use scheduling parameters.

The following topics provide some examples on how to configure and use scheduling parameters:

- Configure scheduling parameters for different types of nodes: describes how to configure scheduling parameters for different types of nodes.
- Compare custom parameters: compares the value formats of custom parameters.
- Process the return values of scheduling parameters: provides examples on how to configure scheduling parameters.

Value types of scheduling parameters

The following table describes the value types of scheduling parameters.

Parameter value type	Description
Data timestamp	The date indicated by the data timestamp of a node. The date is one day earlier than the time when the node is scheduled to run. The value is accurate to days. The data timestamp of a node can be obtained by using the `$bizdate` or `${yyyymmdd}` parameter. Generally, the date indicated by the data timestamp of a node is calculated by subtracting `1` from the date when the node is scheduled to run.
Scheduled time	The time when the node is scheduled to run. The value is accurate to seconds. The scheduled time of a node can be obtained by using the `$cyctime` or `$[yyyymmddhh24miss]` parameter.

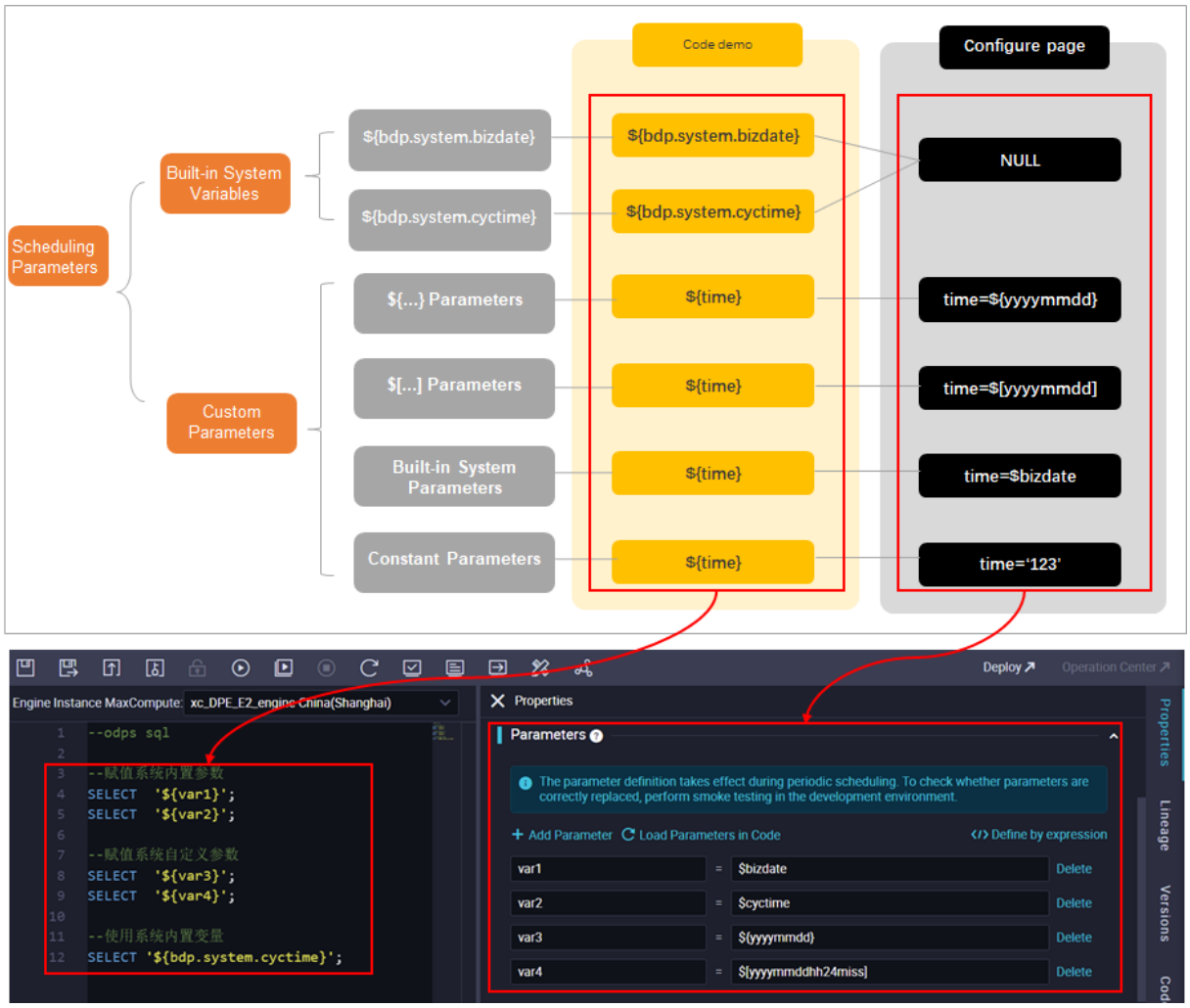
Types of scheduling parameters

Scheduling parameters are classified into **custom parameters** and **built-in variables** based on their value assignment methods.

- **Built-in variables**: If you use built-in variables in the code, they can directly obtain the data timestamp of a node and the time when the node is scheduled to run.
- **Custom parameters**: You can customize variable names in the code based on your business requirements. In this case, you must assign custom parameters to the variables in the **Parameters** section of the **Properties** panel. This way, the variables can obtain time information in the specified

formats.

The following figure shows the types of scheduling parameters.



> **Note**
>
> - For most types of nodes, you can define parameters and assign values to the parameters by using the method shown in the preceding figure. Specific configurations differ for common Shell nodes and PyODPS nodes. For more information, see Configure scheduling parameters for different types of nodes.
> - Some nodes, such as HTTP Trigger nodes, do not support scheduling parameters. For more information about whether a type of node supports scheduling parameters, see the documentation of the type of node.

- **Custom parameters**

  The following table describes the types of custom parameters that DataWorks supports.

Type	Description	Format	Remarks

Type	Description	Format	Remarks
Built-in parameters	You can assign `$bizdate` and `$cyctime` to variables in the Parameters section so that the variables can obtain the data timestamp of a node and the time when the node is scheduled to run.	The value format is fixed. ○ The value format of `$bizdate` is `yyyymmdd`. ○ The value format of `$cyctime` is `yyyymmddhh24miss`.	N/A
${...} parameters	${...} parameters are time parameters that are based on the built-in parameter `$bizdate`. ${...} parameters are generated by combining `yyyy`, `yy`, `mm`, and `dd`.	You can customize the value format. For example, you can specify a ${...} parameter in the format of `${yyyy}`, `${yyyymm}`, `${yyyymmdd}`, or `${yyyy-mm-dd}`. **Note** The value of the `$bizdate` parameter is the same as that of the `${yyyymmdd}` parameter.	○ If you want to specify a time that is N years or months ago, use a ${...} parameter. **Note** ■ ${...} parameters can be accurate only to days. Therefore, you cannot use a ${...} parameter to specify a time that is accurate to hours, minutes, or seconds, such as `${yyyy-mm-dd-1/24}`. ■ If you want to specify a calculation-based time, we recommend that you use a ${...} parameter. For example, you can specify a parameter in the format of `${yyyy-N}` or `${mm-N}`.

Type	Description	Format	Remarks
$[...] parameters	$[...] parameters are time parameters that are based on the built-in parameter `$cyctime`. $[...] parameters are generated by combining `yyyy`, `yy`, `mm`, `dd`, `hh24`, `mi`, and `ss`.	You can customize the value format. For example, you can specify a $[...] parameter in the format of `$[yyyymmdd]`, `$[yyyy-mm-dd]`, `$[hh24miss]`, `$[hh24:mi:ss]`, or `$[yyyymmddhh24miss]`. **Note** The value of the `$cyctime` parameter is the same as that of the `$[yyyymmddhh24miss]` parameter.	◦ If you want to specify a time that is N hours or minutes ago, use a $[...] parameter. **Note** ■ You cannot specify a calculation-based time by using a $[...] parameter in the format of `$[yyyy-N]` or `$[mm-N]`. The variable N indicates the number of years or months. That is, you cannot specify a time that is N years or months ago by using a $[...] parameter in the preceding format. ■ If you want to specify a time that is accurate to seconds, we recommend that you use a $[...] parameter. For example, you can specify a parameter in the format of `$[yyyy-mm-dd-1-1/24]`. After you configure the scheduling parameters, you can test whether the parameters work as expected. For more information, see Test scheduling parameters.

Type	Description	Format	Remarks
Constants	You can assign constants to variables in the Parameters section of the Properties panel. For example, you can assign `"abc"` or `1234` to a variable.	The value format is not fixed.	N/A

For more information about the comparison of value formats of custom parameters, see Compare custom parameters.

- **Built-in variables**

  The following table describes the built-in variables that DataWorks supports.

Type	Format	Description
Data timestamp: `${bdp.system.bizdate}`	Fixed format: `yyyymmdd`	You can reference built-in variables in the code of your node without the need to assign values to them.
Scheduled time: `${bdp.system.cyctime}`	Fixed format: `yyyymmddhh24miss`	

Scheduling parameters

- **Built-in parameters**

  The following table describes the built-in parameters that DataWorks supports.

Built-in parameter	Description
$bizdate	The data timestamp of the node, in the format of `yyyymmdd`. This parameter is widely used. By default, the date indicated by the data timestamp of a node is one day earlier than the time when the node is scheduled to run.
$cyctime	The time when the node is scheduled to run, in the format of `yyyymmddhh24miss`.
$gmtdate	The current date, in the format of `yyyymmdd`. By default, the value of this parameter is the current date. During data backfill, the input value is `one day after the date indicated by the data timestamp`.

Built-in parameter	Description
$bizmonth	The month indicated by the data timestamp, in the format of `yyyymm`. ○ If the month indicated by the data timestamp is the current month, the value of this parameter is `one month before the month indicated by the data timestamp`. ○ If the month indicated by the data timestamp is not the current month, the value of this parameter is `the month indicated by the data timestamp`.
$jobid	The ID of the workflow to which the node belongs.
$nodeid	The ID of the node.
$taskid	The ID of the node instance.

- **Custom parameters ${...}**

  The following table describes how to add or remove specific intervals based on the built-in parameter `$bizdate`. The built-in parameter $bizdate specifies the date that is one day earlier than the time when the node is scheduled to run.

Interval that you want to add or remove	Custom parameter
N years before or later	`${yyyy±N}`
N months before or later	`${yyyymm±N}`
N weeks before or later	`${yyyymmdd±7*N}`
N days before or later	`${yyyymmdd±N}`
N days before or later	`${yyyymmdd±N}`
N years before or later in yyyy format	`${yyyy±N}`
N years before or later in yy format	`${yy±N}`

  Take note of the following variables:

  ○ `yyyy`: indicates the four-digit year, which is the year in the value of the `$bizdate` parameter.
  ○ `yy`: indicates the two-digit year, which is the year in the value of the `$bizdate` parameter.
  ○ `mm`: indicates the month, which is the month in the value of the `$bizdate` parameter.
  ○ `dd`: indicates the day, which is the day in the value of the `$bizdate` parameter.

> **Note**
> - To specify a time that is N months or years ago, we recommend that you use a ${...} parameter.
> - The `$bizdate` parameter can be accurate only to days. Therefore, you cannot use a ${...) parameter to specify a time that is accurate to hours, minutes, or seconds.
> - You can combine engine functions with ${...} parameters to obtain time information in more formats. For more information, see Process the return values of scheduling parameters.

- **Custom parameters $[...]**

  The following table describes how to add or remove specific intervals based on the built-in parameter `$cyctime`. The built-in parameter $cyctime specifies the time when the node is scheduled to run.

Interval that you want to add or remove	Custom parameter
N years later	`$[add_months(yyyymmdd,12*N)]`
N years before	`$[add_months(yyyymmdd,-12*N)]`
N months later	`$[add_months(yyyymmdd,N)]`
N months before	`$[add_months(yyyymmdd,-N)]`
N weeks before or later	`$[yyyymmdd±7*N]`
N days before or later	`$[yyyymmdd±N]`
N hours before or later	You can specify a parameter in one of the following formats: ○ `$[hh24miss±N/24]` ○ `$[Custom time format±N/24]`. Example: `$[hh24±N/24]`
N minutes before or later	You can specify a parameter in one of the following formats: ○ `$[hh24miss±N/24/60]` ○ `$[Custom time format±N/24/60]`. Examples: `$[mi±N/24/60]` and `$[yyyymmddhh24miss±N/24/60]`

  Take note of the following variables:
  - `yyyy`: indicates the four-digit year, which is the year in the value of the `$cyctime` parameter.
  - `yy`: indicates the two-digit year, which is the year in the value of the `$cyctime` parameter.
  - `mm`: indicates the month, which is the month in the value of the `$cyctime` parameter.
  - `dd`: indicates the day, which is the day in the value of the `$cyctime` parameter.
  - `hh24 (24-hour format)` and `hh (12-hour format)`: indicate the hour, which is the hour in the value of the `$cyctime` parameter.

- `ss`: indicates the second, which is the second in the value of the `$cyctime` parameter.
- `mi`: indicates the minute, which is the minute in the value of the `$cyctime` parameter.

> **Note**
>
> - To specify a time that is N hours or minutes ago, we recommend that you use a $[...] parameter.
> - The `$cyctime` parameter is accurate to seconds. Therefore, $[...] parameters are also accurate to seconds.
> - You can combine engine functions with ${...} parameters to obtain time information in more formats. For more information, see Process the return values of scheduling parameters.
> - The values that replace scheduling parameters are determined when instances are generated and do not change with the time when instances actually run.
> - If you use a $[...] parameter to specify a time that is accurate to hours or minutes, the value that replaces the $[...] parameter is determined by the time when the instances of the related node are scheduled to run.
>
>   The following examples show you how the $[...] parameter is replaced:
>
>   - If the node is scheduled to run at `01:00` every day, the value of hh24 or hh is `01`.
>   - If the node is scheduled to run every hour from 00:00 to 23:59, the following situations occur:
>   - The first instance generated for the node is scheduled to run at 00:00, and the value of hh24 or hh is `00`.
>   - The second instance generated for the node is scheduled to run at 01:00, and the value of hh24 or hh is `01`.
>   - The preceding process is repeated every one hour until the last one.

8.2.2. Configure and use scheduling parameters

Scheduling parameters are automatically replaced with specific values based on the data timestamps of nodes and value formats of scheduling parameters. This enables dynamic parameter configuration for node scheduling. This topic describes how to configure and use scheduling parameters. This topic also describes how to perform smoke testing to check whether the scheduling parameters configured for an ODPS SQL node work as expected.

Process of configuring and using scheduling parameters

To configure and use scheduling parameters, perform the following steps:

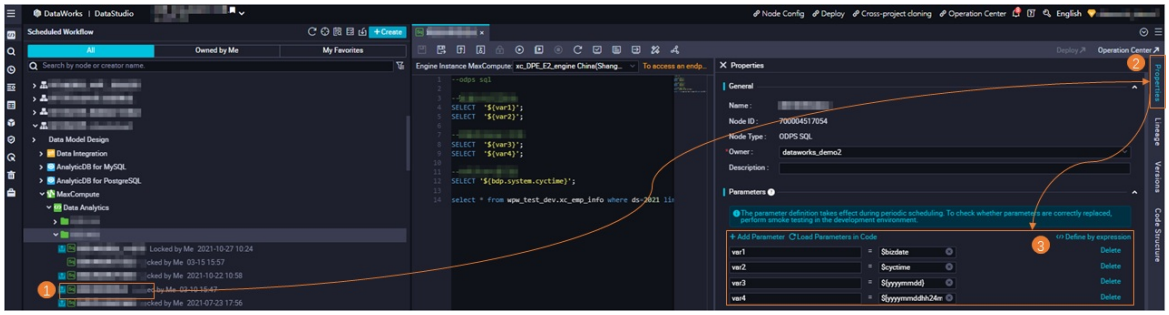
1. Go to the Parameters section of the Properties tab on the configuration tab of a desired node. For more information, see Go to the Parameters section of the Properties tab.
2. Configure scheduling parameters for the node. For more information, see Configure scheduling parameters.
3. Test whether the scheduling parameters work as expected. For more information, see Test scheduling parameters.

4. Check the configurations of the scheduling parameters in the production environment. For more information, see View the configurations of scheduling parameters in Operation Center in the production environment.

This topic provides an example on how to configure scheduling parameters for an ODPS SQL node. For more information, see Example on how to configure scheduling parameters. For more information about how to configure scheduling parameters for other types of nodes, see Configure scheduling parameters for different types of nodes.

Go to the Parameters section of the Properties tab

1. Go to the DataStudio page.
   i. Log on to the DataWorks console. Select a region in the top navigation bar. Then, click **Workspaces** in the left-side navigation pane.
   ii. On the Workspaces page, find your workspace and click **DataStudio** in the Actions column. The DataStudio page appears.
2. Go to the Parameters section of the Properties tab.

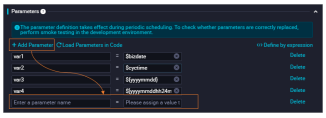
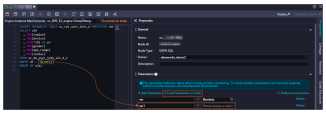


   i. In the Scheduled Workflow pane of the DataStudio page, double-click a desired node to go to the configuration tab of the node.
   ii. On the configuration tab, click the **Properties** tab in the right-side navigation pane.
   iii. In the **Parameters** section of the **Properties** tab, configure scheduling parameters for the node.

Configure scheduling parameters

In the **Parameters** section, you can add parameters or load existing parameters in the code of the node on a visual interface. Alternatively, you can use expressions to define scheduling parameters.

Method to define scheduling parameters	Feature	Description	Screenshot

Method to define scheduling parameters	Feature	Description	Screenshot
Visual interface	**Add Parameter**	You can click **Add Parameter** to configure multiple scheduling parameters for a node. DataWorks provides the following scheduling parameters: • **Previous day:** `${yyyymmdd}` • **Current day:** `$[yyyymmdd]` • **Previous hour:** `$[hh24-1/24]` • **First day of the current month:** `${yyyymm}01` • **First day of the previous month:** `${yyyymm-1}01` For more information about scheduling parameters, see Overview of scheduling parameters.	
	Load Parameters in Code	After you click Load Parameters in Code, DataWorks identifies the variable names defined in the code of the current node and adds the identified variable names to the Parameters section. **Note** • You do not need to assign values to built-in variables. The variables that are loaded are custom variables. You must manually assign values to them. • In most cases, custom variables are defined in the format of `${Custom variable name}` in the code. For PyODPS nodes or common Shell nodes, variable names are defined by using a different method from that for other types of nodes. For more information about how to define custom variables for different types of nodes, see Configure scheduling parameters for different types of nodes.	

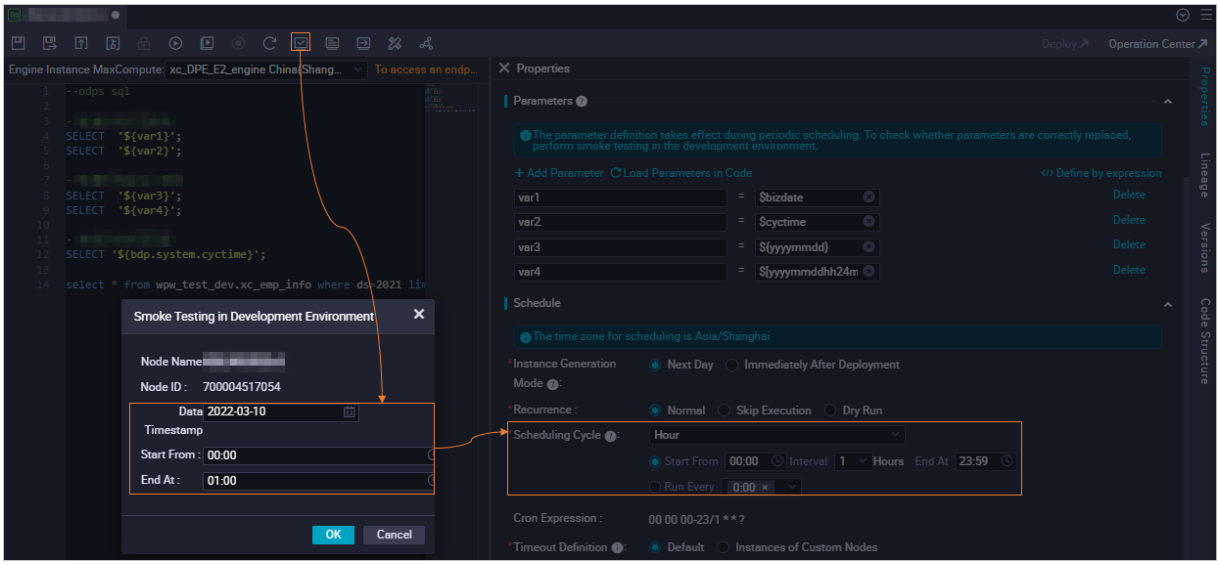
Method to define scheduling parameters	Feature	Description	Screenshot
Expressions	**Define by expression**	By default, the system provides a visual interface on which you can configure scheduling parameters for a node. You can also click **Define by expression** to configure scheduling parameters by using an expression. **Note** • When you use an expression to configure multiple scheduling parameters, separate multiple assignment equations with spaces. For example, if you want to configure two assignment equations `datetime1=$[yyyymmdd]` and `datetime2=$bizdate`, the expression is `datetime1=$[yyyymmdd] datetime2=$bizdate`. • If you add, delete, or modify scheduling parameters by using expressions, DataWorks checks the syntax of the expressions. If the check fails, the scheduling parameters fail to be configured. For example, if a space exists on either side of the equal sign (=), DataWorks reports an error.	

Note If you use the built-in variable `${bdp.system.bizdate}` or `${bdp.system.cyctime}` in the code of a node, you do not need to assign values to the variable in the **Parameters** section.

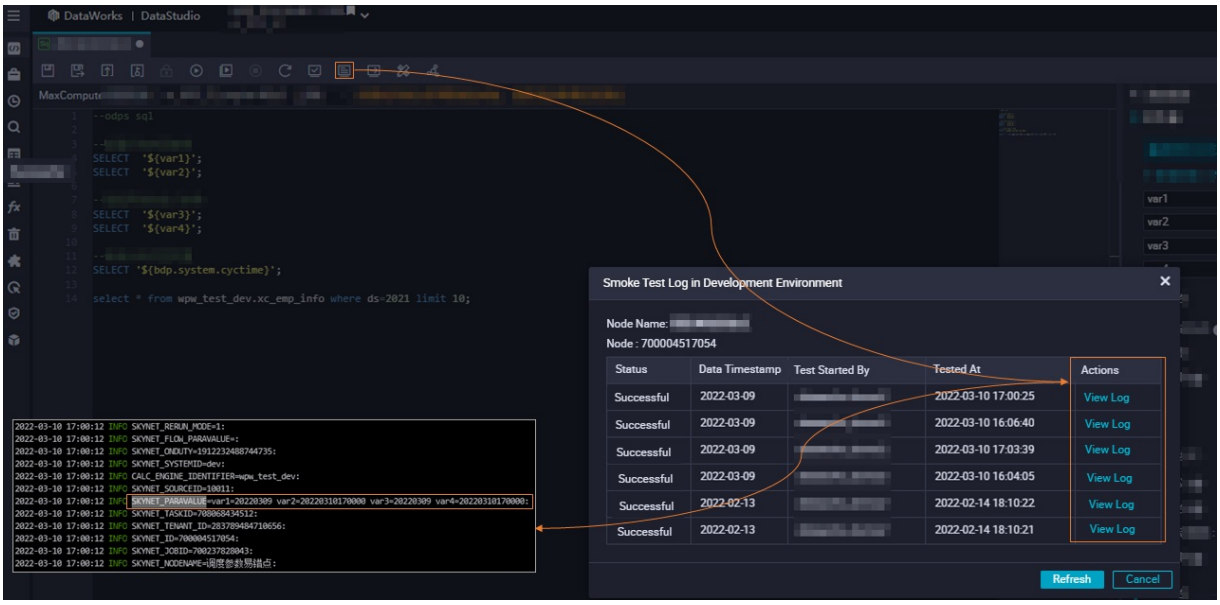
Test scheduling parameters

After you configure scheduling parameters, you must click the icon to perform smoke testing in the development environment. You can configure a data timestamp to simulate the scenario in which the node is scheduled to run. Then, you can check whether the scheduling parameters work as expected.

> **Note** When you perform smoke testing in the development environment, you are charged for the generated test instances. For more information about instance pricing, see Shared resource group for scheduling and Billing of exclusive resource groups for scheduling (subscription).



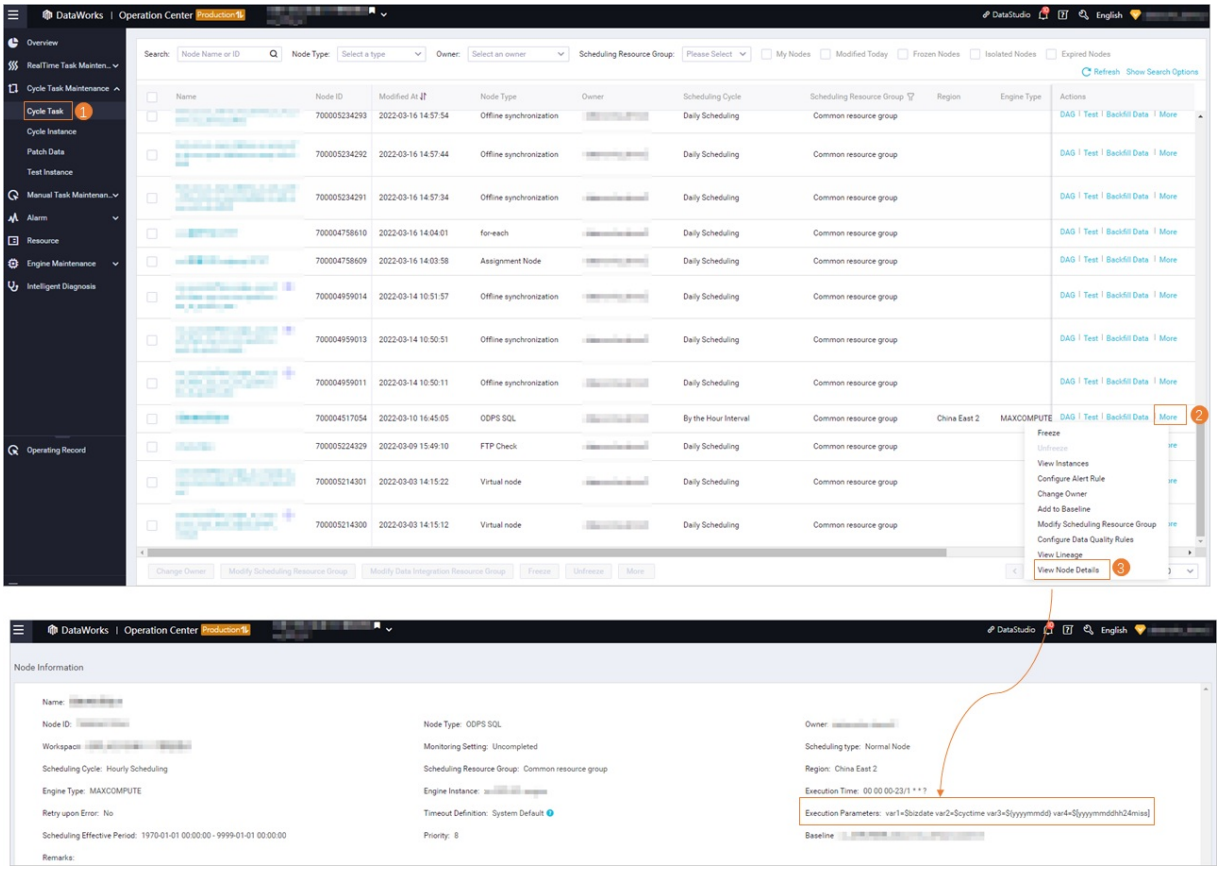
After the smoke testing is complete, click the icon to check whether the results in smoke testing logs meet your expectations.



> **Note**
>
> - Before you click the or icon in the toolbar to run a node, you must manually assign constants to the variables that you want to reference in the code of the node. In this case, you cannot check whether the configured scheduling parameters work as expected.
> - After you modify the code of a node, you must click the icon and then the icon to save the modifications and commit the node to the development environment. Otherwise, you cannot perform smoke testing on the node in the development environment.

View the configurations of scheduling parameters in Operation Center in the production environment

To prevent unexpected configurations of scheduling parameters from affecting the running of an auto triggered node, we recommend that you check the configurations of the scheduling parameters for the auto triggered node on the **Cycle Task** page in Operation Center in the production environment after the auto triggered node is deployed. For more information about auto triggered nodes, see View auto triggered nodes.



> **Note** If the scheduling parameters of an auto triggered node are not configured as expected or you cannot find the auto triggered node in Operation Center, check whether the node is successfully deployed. For more information about how to deploy nodes, see Deploy nodes.

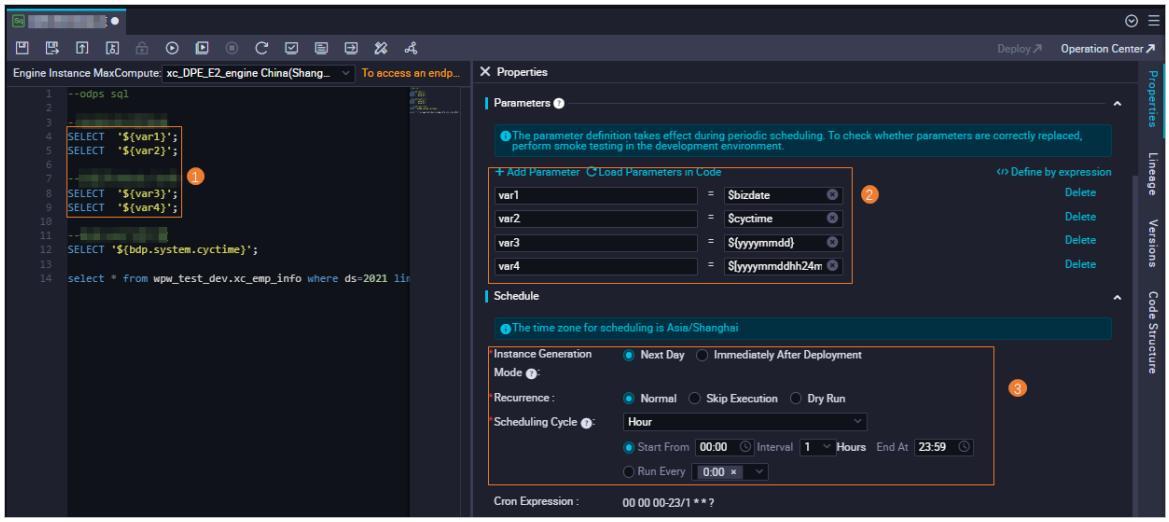
Example on how to configure scheduling parameters

This section describes how to perform smoke testing in the development environment to check whether the scheduling parameters of a node work as expected. In this section, an ODPS SQL node is used. After the node is deployed, you can view the configurations of the scheduling parameters in Operation Center in the production environment.

> **Note** For more information about how to configure scheduling parameters for different types of nodes, see Configure scheduling parameters for different types of nodes.

1. Edit the code of the node and configure the scheduling parameters.

   The following figure shows the code and scheduling parameter configuration of the ODPS SQL node.



   i. Define variables in the code.

   Define variables `'${var1}'` and `'${var2}'` to reference built-in parameters and define variables `'${var3}'` and `'${var4}'` to reference custom parameters in the code of the ODPS SQL node, as shown in Section 1 in the preceding figure.

   ii. Assign values to the variables.

   Go to the **Parameters** section of the **Properties** tab and assign values to the variables, as shown in Section 2 in the preceding figure.

   - `var1=$bizdate`: the data timestamp in the `yyyymmdd` format.
   - `var2=$cyctime`: the time at which the node is scheduled to run in the `yyyymmddhh24miss` format.
   - `var3=${yyyymmdd}`: the data timestamp in the `yyyymmdd` format.
   - `var4=$[yyyymmddhh24miss]`: the time at which the node is scheduled to run in the `yyyymmddhh24miss` format.

iii. (Optional)Configure a scheduling cycle.

Configure the ODPS SQL node to run at an interval of a specified number of hours, as shown in Section 3 in the preceding figure.

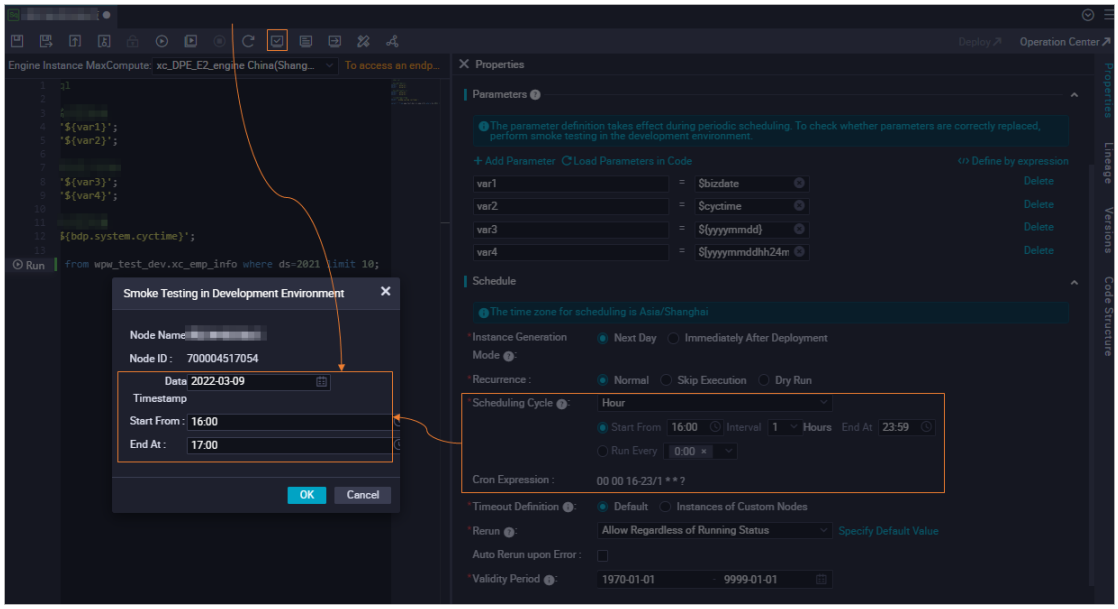
> **Note** You can determine whether to configure a scheduling cycle for a node based on your business requirements. In this example, a scheduling cycle is configured.

- Start from: `16:00`
- End at: `23:59`
- Interval: `1` hour

For more information about how to configure a scheduling cycle, see Configure time properties.

2. In the top toolbar of the configuration tab of the node, click the icon and then the icon to save the configurations of the ODPS SQL node and commit the ODPS SQL node.
3. Perform smoke testing in the development environment.

i. Click the icon. In the **Smoke Testing in Development Environment** dialog box, configure a data timestamp to simulate the scheduling cycle of the node.



In this example, configure a data timestamp in the following way:

- Data timestamp: `2022-03-09`
- Start from: `16:00`
- End at: `17:00`

The ODPS SQL node is scheduled by hour. Two instances are expected to be generated for the node at `16:00` and `17:00` on March 10, 2022.

> **Note** The data timestamp is one day earlier than the time at which the node is scheduled to run. Therefore, the node is scheduled to run on March 10, 2022.

The following results are expected for the instance that is generated at `16:00`:

- `var1=20220309`
- `var2=20220310160000`
- `var3=20220309`
- `var4=20220310160000`

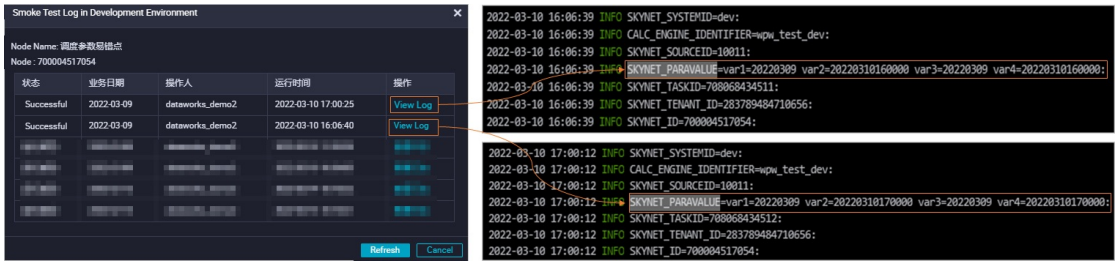
The following results are expected for the instance that is generated at `17:00`:

- `var1=20220309`
- `var2=20220310170000`
- `var3=20220309`
- `var4=20220310170000`

ii. Click **OK**. The node runs at the scheduled time.

iii. After the smoke testing is complete, click the icon to view the smoke testing logs.

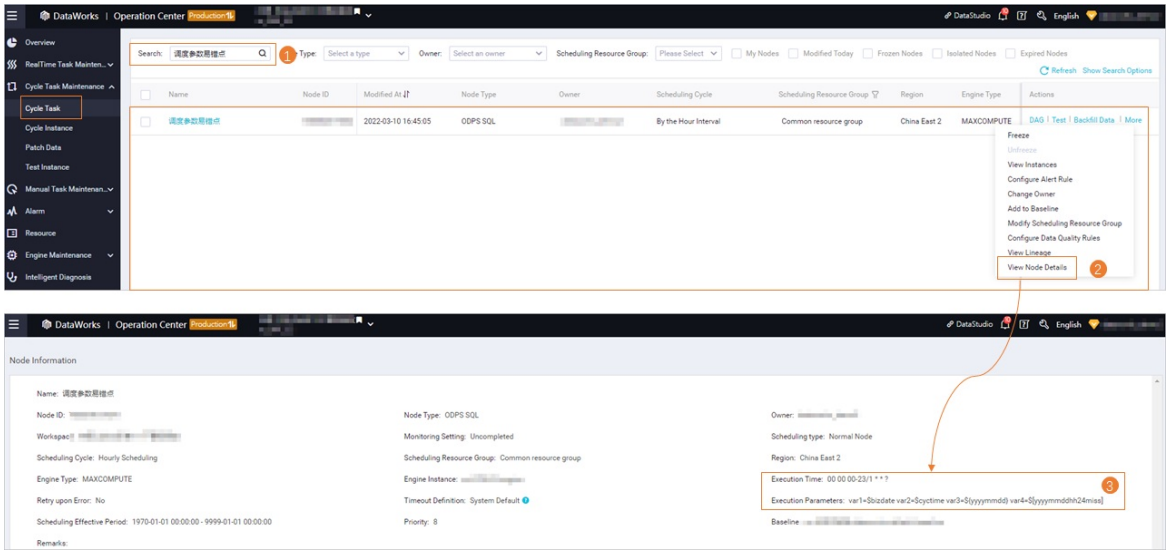
The two instances generated by the node are successfully run and the test result meets expectations.



4. On the configuration tab of the ODPS SQL node, click **Deploy** on the right side of the top navigation bar to deploy the current node.

   For more information about how to deploy nodes, see Deploy nodes.

5. Go to Operation Center and check the configurations of the scheduling parameters of the node.



i. On the **DataStudio** page, click **Operation Center** on the right side of the top navigation bar to go to the Operation Center page.

ii. In the left-side navigation pane, choose **Cycle Task Maintenance > Cycle Task**. On the Cycle Task page, find the node that you want to manage.

> **Note** You can find a node on the Cycle Task page only after the node is deployed.

iii. Choose **More > View Node Details** in the **Actions** column of the node. On the **Node Information** page, view the value of **Execution Parameters**.

In this example, the scheduling parameters of the node are configured as expected in the following way: `var1=$bizdate var2=$cyctime var3=${yyyymmdd} var4=$[yyyymmddhh24miss]` .

8.2.3. Typical scenarios

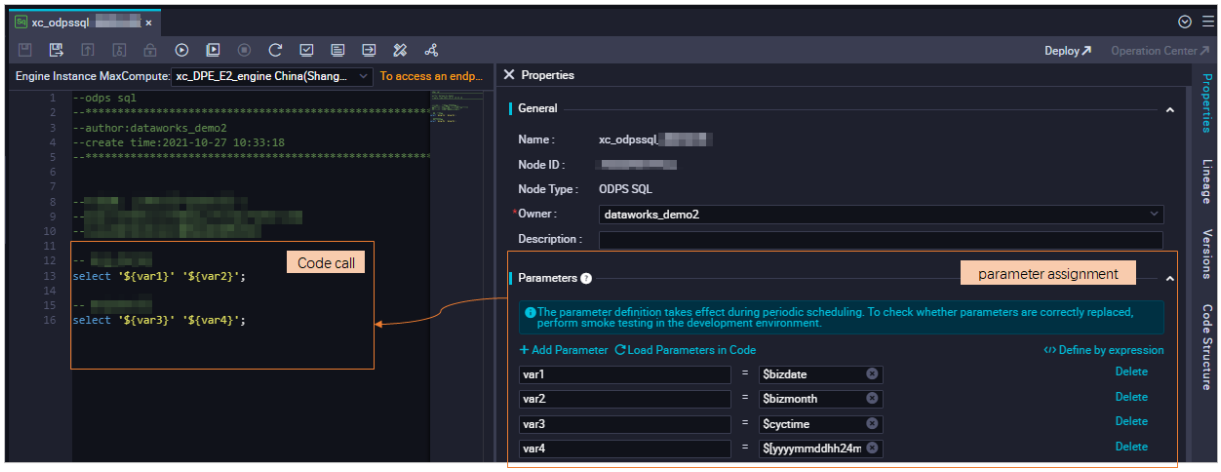
8.2.3.1. Configure scheduling parameters for different types of nodes

You can reference the method of configuring scheduling parameters for SQL nodes, such as ODPS SQL nodes, to define scheduling parameters and assign values to the scheduling parameters for other types of nodes, except common Shell nodes and PyODPS nodes. Specific configurations differ for common Shell nodes and PyODPS nodes. This topic provides examples on how to configure scheduling parameters for different types of nodes.

SQL nodes and batch synchronization nodes

Scheduling parameters are configured for SQL nodes and batch synchronization nodes by using a method that is applicable to most types of nodes. Therefore, you can reference this method to configure scheduling parameters for most types of nodes. In this section, an ODPS SQL node is used as an example. This section describes how to assign values to built-in variables and custom parameters, and how to reference scheduling parameters in code.

> **Note** Some nodes do not support scheduling parameters. For more information about whether a type of node supports scheduling parameters, see the documentation of the type of node.



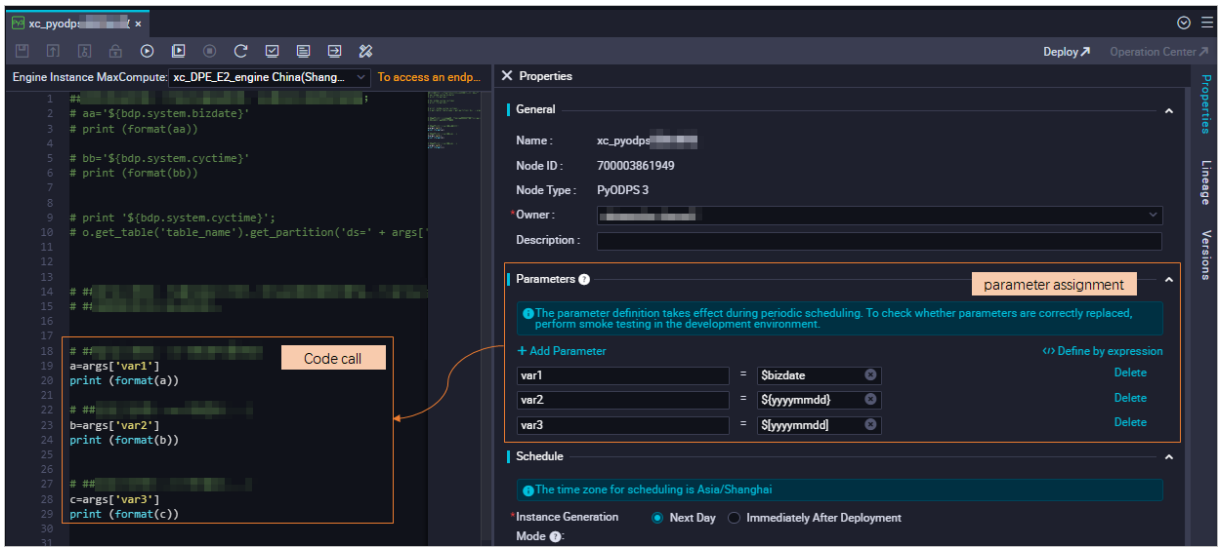
Assign values to the custom variables var1, var2, var3, var4, and var5 in the Parameters section and reference the custom variables in the code editor, as shown in the preceding figure. In this example, the following variables are defined:

- `var1`: Specify $bizdate as var1. This variable is used to obtain the data timestamp.
- `var3`: Specify $cyctime as var3. This variable is used to obtain the scheduled time.
- `var2`: Specify ${yyyymmdd} as var2. This variable is used to obtain the data timestamp.
- `var4`: Specify $[yyyymmddhh24:mi:ss] as var4. This variable is used to obtain the scheduled time.
- `var5`: Assign the constant abc to this variable.

For more information about how to configure and use scheduling parameters, see Configure and use scheduling parameters. For more information about how to assign values to scheduling parameters, see Overview of scheduling parameters.

PyODPS nodes

To avoid changing the code, a PyODPS node does not allow you to replace defined variables with strings that are in the `${param_name}` format in the code. You must add a dictionary object named `args` to global variables before the code is run. This way, the PyODPS node can obtain the values of scheduling parameters from the dictionary object.



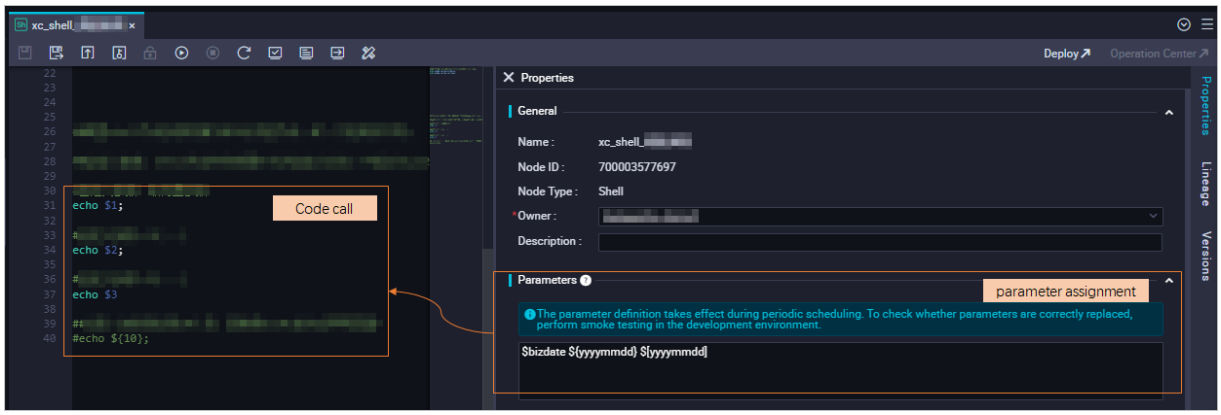
Assign values to the custom variables var1, var2, and var3 in the Parameters section, add a dictionary object named args, and then reference the custom variables in the formats of `args['var1']`, `args['var2']`, and `args['var3']` in the code editor, as shown in the preceding figure. In this example, the following variables are defined:

- `var1`: Specify $bizdate as var1. This variable is used to obtain the data timestamp.
- `var2`: Specify ${yyyymmdd} as var2. This variable is used to obtain the data timestamp.
- `var3`: Specify $[yyyymmdd] as var3. This variable is used to obtain the data timestamp.

For more information about how to configure and use scheduling parameters, see Configure and use scheduling parameters. For more information about how to assign values to scheduling parameters, see Overview of scheduling parameters.

Common Shell nodes

You cannot customize variable names for common Shell nodes. The variables must be named based on their ordinal numbers, such as `$1`, `$2`, and `$3`. If the number of parameters reaches or exceeds 10, use `${number}` to declare the excess variables. For example, use ${10} to declare the tenth variable.



Assign values to the custom variables $1, $2, and $3 in the Parameters section and reference the custom variables in the code editor, as shown in the preceding figure. In this example, the following variables are defined:

> **Note** For common Shell nodes, you can assign values to scheduling parameters only by using expressions. Separate multiple values with spaces. Make sure that you assign the values to the parameters in the same order in which the parameters are defined. For example, the first value `$bizdate` that you enter in the Parameters section is assigned to the first parameter $1.

- `$1`: Specify $bizdate as $1. This variable is used to obtain the data timestamp.
- `$2`: Specify ${yyyymmdd} as $2. This variable is used to obtain the data timestamp.
- `$3`: Specify $[yyyymmdd] as $3. This variable is used to obtain the data timestamp.

For more information about how to configure and use scheduling parameters, see Configure and use scheduling parameters. For more information about how to assign values to scheduling parameters, see Overview of scheduling parameters.

8.2.3.2. Compare custom parameters

Custom parameters are classified into built-in parameters, custom parameters ${...}, custom parameters $[...], and constants. Values are assigned to each type of parameter in the specified formats. This topic compares the value formats of different types of custom parameters.

How different custom parameters are used

For example, the current date is `November 1, 2021`, and a node is scheduled to run at `00:00` every day. The following table describes how different customer parameters are assigned values.

> **Note** In this example, the parameters are referenced in the `pt=${datetime}` format in the code.

Parameter	Description	Assignment	Replacement result
`${yyyymmdd}`	Obtains the data timestamp of the node.	`datetime=${yyyymmdd}`	`datetime=20211031`
`$[yyyymmddhh24miss]`	Obtains the time when the node is scheduled to run, which is accurate to seconds.	`datetime=$[yyyymmddhh24miss]`	`datetime=202111010000`
`$bizdate`	Obtains the data timestamp of the node.	`datetime=$bizdate`	`datetime=20211031`
`$cyctime`	Obtains the time when the node is scheduled to run, which is accurate to seconds.	`datetime=$bizdate`	`datetime=20211101000000`
`$gmtdate`	Obtains the current date, which is accurate to days.	`datetime=$gmtdate`	`datetime=20211101`
`$bizmonth`	Obtains the month indicated by the data timestamp of the node.	`datetime=$bizmonth`	• If the month indicated by the data timestamp is the current month, the value of the `$bizmonth` parameter is the previous month. • If the month indicated by the data timestamp is not the current month, the value of the `$bizmonth` parameter is the month indicated by the data timestamp. For example, the current date is `November 1, 2021`: • If the date indicated by the data timestamp is `November 2, 2021`, which is in the current month, the value of `datetime` is 202110. • If the date indicated by the data timestamp is `October 31, 2021`, which is not in the current month, the value of `datetime` is 202110.

Differences between ${...} and $[...] parameters

The following table describes the differences between the ${...} and $[...] parameters.

Item	${...} parameter	$[...] parameter
Benchmark	The value of the `$bizdate` parameter is used as a benchmark to run nodes. The `$bizdate` parameter indicates the data timestamp. By default, the date indicated by the data timestamp is one day earlier than the current date.	The value of the `$cyctime` parameter is used as a benchmark to run nodes. The `$cyctime` parameter indicates the time when a node is scheduled to run. For example, the value `yyyy-mm-dd 00:30:00` indicates `00:30` on the current day.
Data backfill	The parameter is replaced with the selected data timestamp.	During data backfill, the parameter is replaced with the date that is one day after the date indicated by the selected data timestamp. For example, `20220315` is the date indicated by the selected data timestamp for data backfill. In this case, the date indicated by the value of the `$cyctime` parameter is `20220316`.
Time granularity	The value is accurate to days.	The value is accurate to seconds. **Note** `${yyyy-mm-dd-1/24}` is not supported. We recommend that you use `$[yyyy-mm-dd-1-1/24]`.

- For more information about how to use ${...} and $[...] parameters, see Overview of scheduling parameters.
- For more information about how to configure and use custom parameters, see Configure and use scheduling parameters.

In this example, an ODPS SQL node is used, and the current time is `10:30:00 on July 20, 2021`. The following table describes how to obtain the required time data by using the ${...} and $[...] parameters.

Time to obtain	${...} parameter	$[...] parameter

Time to obtain	${...} parameter	$[...] parameter
Year: 2021	• Assignment: `datetime=${yyyy}` • Reference in code: `pt=${datetime}` • Replacement result: `pt=${yyyy}=2021`	• Assignment: `datetime=$[yyyy]` • Reference in code: `pt=${datetime}` • Replacement result: `pt=$[yyyy]=2021`
Year: 21	• Assignment: `datetime=${yy}` • Reference in code: `pt=${datetime}` • Replacement result: `pt=${yy}=21`	• Assignment: `datetime=$[yy]` • Reference in code: `pt=${datetime}` • Replacement result: `pt=$[yy]=21`
Year: 2020	• Assignment: `datetime=${yyyy-1}` • Reference in code: `pt=${datetime}` • Replacement result: `pt=${yyyy-1}=2020`	Not supported
Month: 07	• Assignment: `datetime=${mm}` • Reference in code: `pt=${datetime}` • Replacement result: `pt=${mm}=07`	• Assignment: `datetime=$[mm]` • Reference in code: `pt=${datetime}` • Replacement result: `pt=$[mm]=07`
Day: 20	• Assignment: `datetime=${dd}` • Reference in code: `pt=${datetime}` • Replacement result: `pt=${dd}=20`	• Assignment: `datetime=$[dd]` • Reference in code: `pt=${datetime}` • Replacement result: `pt=$[dd]=20`
Date: 2021-06-20	• Assignment: `datetime=${yyyy-mm-dd-29}` • Reference in code: `pt=${datetime}` • Replacement result: `pt=${yyyy-mm-dd-29}=2021-06-20`	• Assignment: `datetime=$[add_months(yyyymmdd,-1)]` • Reference in code: `pt=${datetime}` • Replacement result: `pt=$[add_months(yyyymmdd,-1)]=2021-06-20`

Time to obtain	${...} parameter	$[...] parameter
Date: 2021-07-19	• Assignment: `datetime=${yyyy-mm-dd}` • Reference in code: `pt=${datetime}` • Replacement result: `pt=${yyyy-mm-dd}=2021-07-19`	• Assignment: `datetime=$[yyyy-mm-dd-1]` • Reference in code: `pt=${datetime}` • Replacement result: `pt=$[yyyy-mm-dd-1]=2021-07-19`
Date: 2020-07-20	• Assignment: `datetime=${yyyy-mm-dd-364}` • Reference in code: `pt=${datetime}` • Replacement result: `pt=${yyyy-mm-dd}=2020-07-20`	• Assignment: `datetime=$[add_months(yyyymmdd,-12*1)]` • Reference in code: `pt=${datetime}` • Replacement result: `pt=$[add_months(yyyymmdd,-12*1)]=2020-07-20`
Time: 10:30:00	Not supported	• Assignment: `datetime=$[hh24:mi:ss]` • Reference in code: `pt=${datetime}` • Replacement result: `pt=$[hh24:mi:ss]=10:30:00`

Time: 2021-07-20 10:30:00	Not supported	• Assignment: `datetime1=$[yyyy-mm-dd] datetime2=$[hh24:mi:ss]`
		Note You must customize two variables datetime1 and datetime2, and separate the key-value pairs of the variables with a space.
		• Reference in code: `pt=${datetime1} ${datetime2}`
		• Replacement result:
		  ◦ `datetime1=$[yyyy-mm-dd]=2021-07-20`
		  ◦ `datetime2=$[hh24:mi:ss]=10:30:00`
		  ◦ `pt=2021-07-20 10:30:00`

Time to obtain	${...} parameter	$[...] parameter
Time: 2021-07-20 10:29:00	Not supported	• Assignment: `datetime1=$[yyyy-mm-dd] datetime2=$[hh24:mi:ss-1/24/60]` Note You must customize two variables datetime1 and datetime2, and separate the key-value pairs of the variables with a space. • Reference in code: `pt=${datetime1} ${datetime2}` • Replacement result: ◦ `datetime1=$[yyyy-mm-dd]=2021-07-20` ◦ `datetime2=$[hh24:mi:ss-1/24/60]=10:29:00` ◦ `pt=2021-07-20 10:29:00`
Time: 2021-07-20 09:30:00	Not supported	• Assignment: `datetime1=$[yyyy-mm-dd] datetime2=$[hh24:mi:ss-1/24]` Note You must customize two variables datetime1 and datetime2, and separate the key-value pairs of the variables with a space. • Reference in code: `pt=${datetime1} ${datetime2}` • Replacement result: ◦ `datetime1=$[yyyy-mm-dd]=2021-07-20` ◦ `datetime2=$[hh24:mi:ss-1/24]=09:30:00` ◦ `pt=2021-07-20 09:30:00`

8.2.3.3. Process the return values of scheduling parameters

You can obtain only time data by configuring scheduling parameters for nodes. After you configure scheduling parameters for specific nodes, such as batch synchronization nodes, the nodes cannot use the return values of the scheduling parameters unless you process the return values by using methods such as functions. This topic describes how to process the return values of scheduling parameters in typical scenarios.

Background information

The time data that scheduling parameters support is not applicable to all scenarios. If you need time data in special formats based on your business requirements, you can use engine functions to process the time data. You cannot use functions to process the return values of the scheduling parameters for some nodes. In this case, you can use assignment nodes to process the return values of the scheduling parameters.

You can process the return values of scheduling parameters by using functions or assignment nodes in the following scenarios:

- **Use functions**

  In this topic, an ODPS SQL node is used. You can process the return values of scheduling parameters by using functions in the following scenarios:

  - Obtain the last day of the previous month.
  - Obtain the quarter to which the scheduled time belongs
  - Obtain the year, month, day, hour, and minute of the time that is 15 minutes before the scheduled time
  - Obtain the interval for data synchronization with the interval for periodic scheduling being one day
  - Obtain the interval for data synchronization with the interval for periodic scheduling being one hour

- **Use assignment nodes**

  For specific nodes, you cannot use functions to process the return values of scheduling parameters. If such a node requires timestamps or time data in another format, you can use assignment nodes to convert the return values of scheduling parameters to the required format, and pass the processed values to the node. For more information about how to use assignment nodes, see Configure an assignment node.

  For example, if a batch synchronization node needs to use timestamp data for incremental synchronization, you can convert time data to timestamp data by using a function defined for an assignment node, and pass the timestamp data to the batch synchronization node.

For more information about operations related to scheduling parameters, see the following topics:

- Configure and use scheduling parameters and SQL nodes and batch synchronization nodes: describe how to configure scheduling parameters for ODPS SQL nodes.
- Overview of scheduling parameters: describes how to assign values to scheduling parameters.
- I run an instance of a node at 00:00 on the current day to analyze the data in the partition that corresponds to 23:00 on the previous day. However, the data in the partition that corresponds to 23:00 on the current day is analyzed. What do I do?: describes how to configure scheduling parameters to obtain time data across days.

Obtain the last day of the previous month.

The following table describes how to configure a scheduling parameter to obtain the last day of the previous month.

Parameter configuration	Scheduled time for testing	Returned result
• Assignment: `last_month=$[yyyy-mm]` • Syntax to process the return value: `SELECT REPLACE(DATEADD(date'${last_month}-01',-1,'dd'),'-','');` • Format of the returned result: `yyyymmdd`	`20210926000000`	`20210831`

Obtain the quarter to which the scheduled time belongs

The following table describes how to configure a scheduling parameter to obtain the quarter to which the scheduled time belongs.

Parameter configuration	Scheduled time for testing	Returned result
• Assignment: `month=$[mm]` • Syntax to process the return value: `SELECT CEIL(INT('${month}')/3);` • Type of the returned result: positive integer	`20211025001700`	4

Obtain the year, month, day, hour, and minute of the time that is 15 minutes before the scheduled time

The following table describes how to configure scheduling parameters to obtain the year, month, day, hour, and minute of the time that is 15 minutes before the scheduled time.

Parameter configuration	Scheduled time for testing	Returned result

Parameter configuration	Scheduled time for testing	Returned result

• Assignment:
  ◦ `year=$[yyyy-15/24/60]`
  ◦ `month=$[yyyymm-15/24/60]`
  ◦ `day=$[yyyymmdd-15/24/60]`
  ◦ `hour=$[hh24-15/24/60]`
• Syntax to process the return value: `select 'year=${year} month=${month} day=${day} hour=${hour} mi=${mi}';`
• Format of the returned result:
  ◦ Year: `yyyy`
  ◦ Month: `yyyymm`
  ◦ Day: `yyyymmdd`
  ◦ Hour: `hh`
  ◦ Minute: `mm`
	`20210727000500`	• `year=2021`
		• `month=202107`
		• `day=20210726`
		• `hour=23`
		• `mi=50`

Obtain the interval for data synchronization with the interval for periodic scheduling being one day

Obtain the interval for data synchronization between `00:00:00` of the previous day and `00:00:00` of the current day. The interval for periodic scheduling is one day and accurate to seconds, in the format of `yyyymmddhh24miss`.

> **Note** If you create a batch synchronization node to synchronize data generated in a Kafka or LogHub data source in a specified interval and you need to configure the scheduling parameters, you must set the data timestamp in the format of `yyyymmddhh24miss`, and the interval must be left-closed and right-open. For more information, see Kafka Reader and LogHub (SLS) Reader. For more information about incremental synchronization in different scenarios, see Synchronize incremental data.

The following table describes how to configure scheduling parameters to obtain the interval for data synchronization.

Parameter configuration	Scheduled time for testing	Returned result

• Assignment:
  ◦ `beginDateTime=$[yyyymmdd-1]`
  ◦ `endDateTime=$[yyyymmdd]`
• Syntax to process the return value: `select '${beginDateTime}000000 ${endDateTime}000000';`
• Format of the returned result: `yyyymmddhh24miss`
	`20220117002300`	• `20220116000000`
		• `20220117000000`

Obtain the interval for data synchronization with the interval for periodic scheduling being one hour

Obtain the interval for data synchronization between `00:00:00` of the previous day and `00:00:00` of the current day. The interval for periodic scheduling is one hour and accurate to seconds, in the format of `yyyymmddhh24miss`.

> **Note** If you create a batch synchronization node to synchronize data generated in a Kafka or LogHub data source in a specified interval and you need to configure the scheduling parameters, you must set the data timestamp in the format of `yyyymmddhh24miss`, and the interval must be left-closed and right-open. For more information, see Kafka Reader and LogHub (SLS) Reader. For more information about incremental synchronization in different scenarios, see Synchronize incremental data.

The following table describes how to configure scheduling parameters to obtain the interval for data synchronization.

Parameter configuration	Scheduled time for testing	Returned result

• Assignment:
  ◦ `beginDateTime=$[yyyymmddhh24-1/24]`
  ◦ `endDateTime=$[yyyymmddhh24]`
• Syntax to process the return value: `select '${beginDateTime}0000 ${endDateTime}0000';`
• Format of the returned result: `yyyymmddhh24miss`
	`20220117002300`	• `20220116230000`
		• `20220117000000`

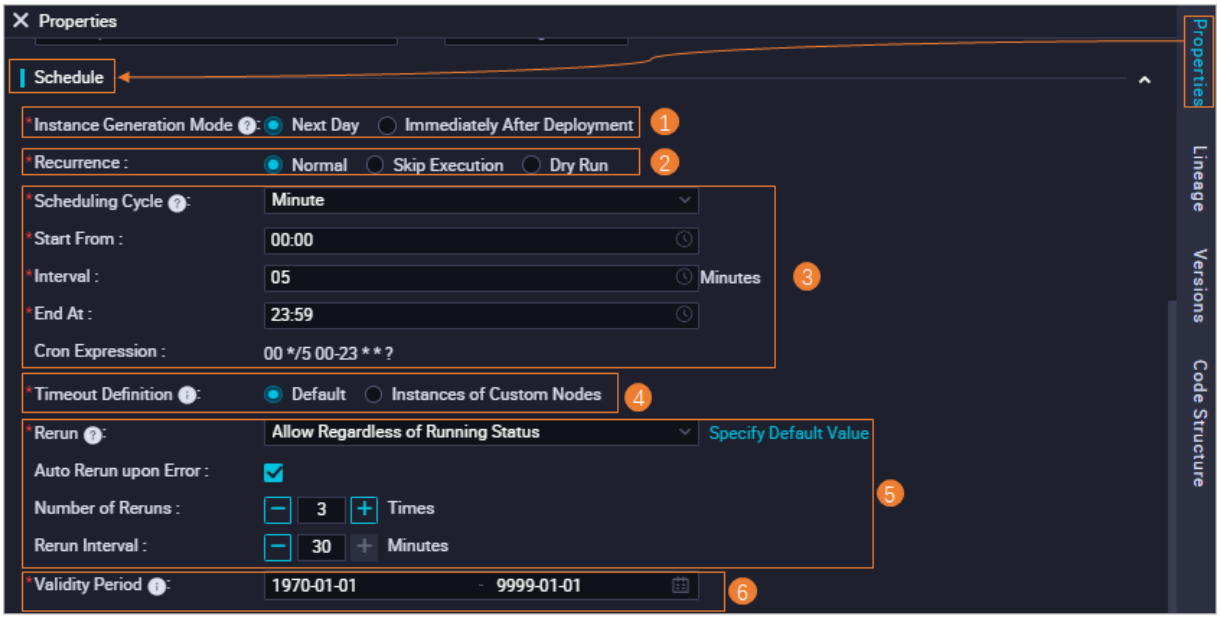
8.3. Configure time properties

8.3.1. Configure time properties

After a node is committed to and deployed in the production environment, you can configure time properties for the node, such as the instance generation mode, recurrence and scheduled time, rerun properties, and timeout period for the node. This topic describes how to configure time properties for a node.

Context

Go to the DataStudio page, double-click a node in the Business Flow section in the left-side navigation pane, and then click the **Properties** tab in the right-side navigation pane. In the Properties panel, configure time properties for the node in the Schedule section.



The following table describes the configuration items required for configuring time properties for a node.

Configuration item	Description
Instance generation modes	The mode in which instances are generated for the node in the production environment.
Scheduling types	The mode in which the node is run in the production environment.
Recurrence	The recurrence of the node. The recurrence determines the number of instances to be generated for this node and the time when the instances are run in the production environment.
Timeout period	The timeout period for the node. If the period of time for which the node is run exceeds the specified timeout period, the node is automatically terminated.
Rerun properties	Specifies whether to rerun the node and the conditions in which the node can be rerun. When you set this parameter, make sure that the data idempotence of the node is not affected.
Validity Period	The period of time during which the node is automatically run as scheduled. After the period of time expires, the node is not automatically run.

Instance generation modes

After a node is committed to and deployed in the production environment, auto triggered node instances are automatically generated and scheduled based on the instance generation mode that you specify. Valid values of the **Instance Generation Mode** parameter:

- **Next Day**: generates auto triggered node instances on the next day. After the node is committed and deployed, auto triggered node instances are generated and automatically scheduled on the next day.
- **Immediately After Deployment**: generates auto triggered node instances immediately after the node is committed and deployed. The auto triggered node instances are run only if the scheduled time is 10 or more minutes later than the time you commit and deploy the node. If the scheduled time is in the past or less than 10 minutes later than the time when you commit and deploy the node, the auto triggered node instances perform dry runs. For more information, see Configure immediate instance generation for a node.

> **Note**
>
> 1. After an auto triggered node is committed and deployed, you can view the status of the node on the Cycle Task page of Operation Center. The scheduling system generates instances every day for the auto triggered node that is deployed in the production environment. The instance generation mode determines when the auto triggered node instances take effect. For more information about how to view the auto triggered node in Operation Center, see View auto triggered nodes.
> 2. If the node is committed and deployed between 23:30 and 00:00, the instance generation mode that you specify does not take effect on the day.
>   - If Instance Generation Mode is set to Next Day, and the node is committed and deployed before 23:30, the node instance is generated and takes effect on the next day.
>   - If the node is committed and deployed between 23:30 and 00:00, instances are generated and take effect the day after the next day.
> 3. If you select Immediately After Deployment, instances are generated and run only if the scheduled time of the node is 10 or more minutes later than the time you commit and deploy the node.
> 4. If you select Immediately After Deployment, the dependencies of nodes whose recurrence is modified may be affected on this day. Instances that have been run are not deleted, whereas instances whose scheduled time has not arrived are replaced by immediately generated instances. For more information, see Configure immediate instance generation for a node.

Scheduling types

The following table describes the valid values of the Recurrence parameter.

Value	Description	Scenario

Value	Description	Scenario
Normal	If you set the Recurrence parameter to Normal, the node is run and generates data based on the setting of the Scheduling Cycle parameter. After the node is run as expected, the descendant nodes are also triggered and run. By default, the Recurrence parameter is set to Normal.	You want a node and the instances that are generated for this node to be run as expected.
Skip Execution	If you set the Recurrence parameter to Skip Execution, the node is scheduled based on the recurrence and the scheduled time that you specify. However, the status of the node becomes Freeze and no data is generated for this node. When this node is scheduled, the status of the node becomes Run failed and the descendant nodes cannot be run. **Note** The following icon is displayed next to the name of a node that is suspended for scheduling:	You want to suspend a node and the instances generated for this node. In this case, the current node and its descendant nodes cannot be run. You can suspend the root node of a workflow in a period of time based on your business requirements. You can also unfreeze the root node to resume the workflow as needed. For more information, see Node freezing and unfreezing.
Dry Run	If you set the Recurrence parameter to Dry Run, the node is run based on the setting of the Scheduling Cycle parameter. However, the node performs a dry run and no data is generated. When the node is scheduled, the scheduling system returns a success response. However, the duration is `0` seconds, and no operational logs are generated. This dry-run node does not affect the execution of its descendant nodes and occupies no resources.	You want to suspend a node in a period of time and require that its descendant nodes be run as expected.

Recurrence

Note Scheduling settings take effect only if you turn on **Periodic scheduling** on the **Scheduling Settings** tab. For more information, see Configure scheduling settings.

- **Background information**

The Scheduling Cycle parameter determines the frequency at which the code of a node is run after the node is committed to the production environment. After a node is committed and deployed, the scheduling system generates auto triggered node instances every day from the next day. Then, the auto triggered node instances are run based on the output results of their ancestor instances and the scheduled time specified for this node. If a node is committed and deployed between 23:30 and 00:00, instances are generated and automatically scheduled for the node from the day after the next day.

- **Valid values of the Scheduling Cycle parameter**
  - Minute: The node is automatically run once every N minutes within a specific period every day. For more information, see Schedule a node by minute.
  - Hour: The node is automatically run once every N hours within a specific period every day. For more information, see Schedule a node by hour.
  - Day: The node is automatically run once every day. For more information, see Schedule a node by day.
  - Week: The node is automatically run at a specified time on specific days every week. For more information, see Schedule a node by week.
  - Month: The node is automatically run at a specified time on specific days every month. For more information, see Schedule a node by month.
  - Year: The node is automatically run at a specified time on specific days every year. For more information, see Schedule a node by year.
- **Usage notes**
  - If a node is committed and deployed between 23:30 and 00:00, the scheduling system generates instances for the node from the day after the next day.
  - The dependencies of an auto triggered node take priority over its time properties. The scheduling system does not immediately run the instances of a node at the scheduled time. The system first checks whether all the ancestor instances of the node are successful.

    > **Note**
    > - If not all its ancestor instances are successful, the node instance is in the Pending (Ancestor) state at the scheduled time.
    > - If all its ancestor instances are successful, the node instance enters the Pending (Schedule) state before the scheduled time arrives.
    > - If all its ancestor instances are successful, the node instance enters the Pending (Resources) state at the scheduled time.
    >
    > For information about how to troubleshoot a node that fails to run at the scheduled time, see Why is an auto triggered instance not run after its scheduling time arrives?.

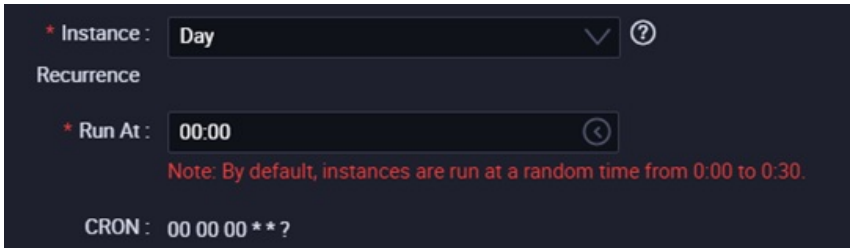
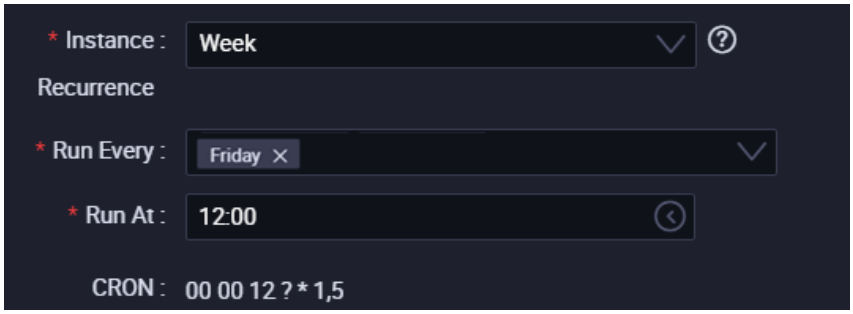
  - The time when a node is scheduled may be inconsistent with the scheduled time that you specify for the node due to unexpected issues. For example, the node needs to wait for resources.

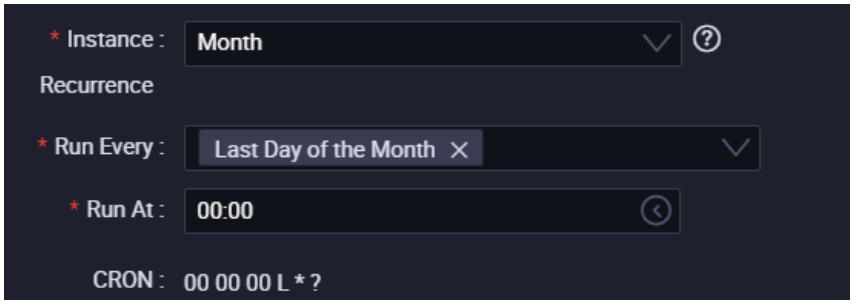
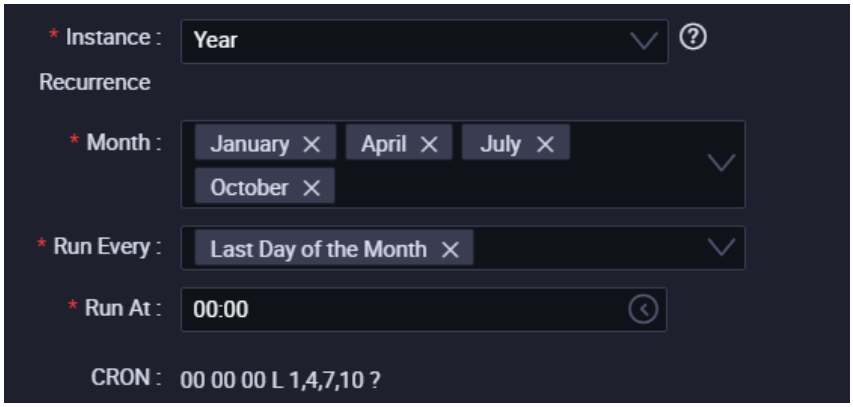
- For a node that is scheduled by week, month, or year, the scheduling system runs the node at the scheduled time every day. On the days that are not specified to run the node, the node performs a dry run and no data is generated for this node. The following examples are effects of a dry run:
  - The scheduling system returns a success response, and the duration is `0` seconds.
  - No operational logs are generated.
  - The execution of descendant nodes is not affected.
  - The dry-run node occupies no resources.

  > **Note**
  >
  > If you schedule a node to run every Monday, the node is run and generates data only on Mondays. On the other days of the same week, the node performs a dry run and the scheduling system returns a success response. If you test the node or backfill data for the node, you must set the data timestamp to Sunday. The data timestamp is one day earlier than the scheduled time of the node. This way, the node can be tested or data can be backfilled when the node is run and generates data on Monday.

- **Examples**
  - If a node that is scheduled by day is set to depend on a node that is scheduled every Monday, the ancestor node performs a dry run on the days except Monday, and the descendant node is automatically scheduled and generates data every day.
  - If you want the descendant node to be scheduled every Monday, you can configure time properties for the descendant node to be the same as the ancestor node. On the days except Monday, the descendant node also performs a dry run.

- **Scenarios and related configurations**

Scenario	Configuration
Schedule a node to run at 00:00 every day	
Schedule a node to run at 12:00 every Friday	

Scenario	Configuration
Schedule a node to run on the last day of every month	
Schedule a node to run on the last day of the first month in every quarter	

Timeout period

You can use the Timeout Definition parameter to specify a timeout period for a node. If the period of time for which the node is run exceeds the specified timeout period, the node is automatically terminated. Take note of the following items when you use this parameter:

- The timeout period applies to auto triggered node instances, data backfill instances, and test node instances.
- The default timeout period ranges from 72 hours to 168 hours. The system automatically adjusts the default timeout period for a node based on system loads.
- You can customize a timeout period, but it cannot exceed 168 hours.

> **Note** For exclusive resource groups for scheduling that you purchased before January 7, 2021, submit a ticket to contact the technical support staff to update the resource groups. The Timeout Definition parameter is available only after you update the resource groups.

Rerun properties

In the Schedule section, you can configure the conditions, interval, and number of times for rerunning a node.

> **Note**
>
> - When you configure rerun properties for a node, make sure that the data idempotence of the node is not affected based on your business requirements. This helps prevent data quality issues after a failed node is rerun. For example, when you create and develop an ODPS SQL node, you can replace the `INSERT INTO` statement with the `INSERT OVERWRITE` statement.
> - You can go to the **Scheduling Settings** tab to configure default scheduling settings for nodes to be created. For more information, see Configure scheduling settings.

- **Rerun**

  The following table describes the valid values of the Rerun parameter.

  > **Note** You can click **Specify Default Value** next to the Rerun parameter to go to the Scheduling Settings tab.

Value	Scenario
Allow Regardless of Running Status	If the data idempotence of a node is not affected after the node is rerun multiple times, you can set the Rerun parameter to this value.
Allow upon Failure Only	If the rerun of a failed node does not affect the data idempotence but the rerun of a successful node does, you can set the Rerun parameter to this value.
Disallow Regardless of Running Status	If the data idempotence of a node cannot be ensured after the node is rerun, you can set the Rerun parameter to this value. **Note** ◦ If you set the Rerun parameter to Disallow Regardless of Running Status, the system does not automatically rerun the node after the system recovers from an exception. ◦ The **Auto Rerun upon Error** parameter is not displayed if you set the Rerun parameter to Disallow Regardless of Running Status.

- **Auto Rerun upon Error**

  The following table describes the parameters you must set if you allow automatic reruns after an error occurs.

Parameter	Description

Parameter	Description
Number of Reruns	The default number of times that an auto triggered node is rerun after it fails to be run as scheduled. Valid values: 1 to 10. A value of 1 indicates that nodes are not rerun after it fails to be run as scheduled. A value of 10 indicates that nodes are rerun nine times after it fails to be run as scheduled. You can set this parameter based on your business requirements.
Rerun Interval	The interval at which a node is rerun after it fails to be run as scheduled. You can set this parameter based on your requirements. Valid values: 1 to 30. Default value: 30. Unit: minutes.

Note

- The **Auto Rerun upon Error** parameter is not displayed if you set the Rerun parameter to **Disallow Regardless of Running Status**. In this case, the node is not allowed to rerun after it fails to be run as scheduled.
- You can set the default number of reruns and default rerun interval for the nodes in the workspace on the **Scheduling Settings** tab. For more information, see Configure scheduling settings.
- The automatic rerun feature does not take effect if a node is automatically terminated because the timeout period is exceeded.

Validity Period

You can specify a validity period during which a node is automatically run as scheduled. After the period expires, the node is not automatically run. Nodes whose validity period expires are expired nodes. You can view the number of expired nodes on the Overview page of Operation Center and unpublish the nodes based on your requirements. For more information about the Overview page of Operation Center, see View the statistics on the Overview page.

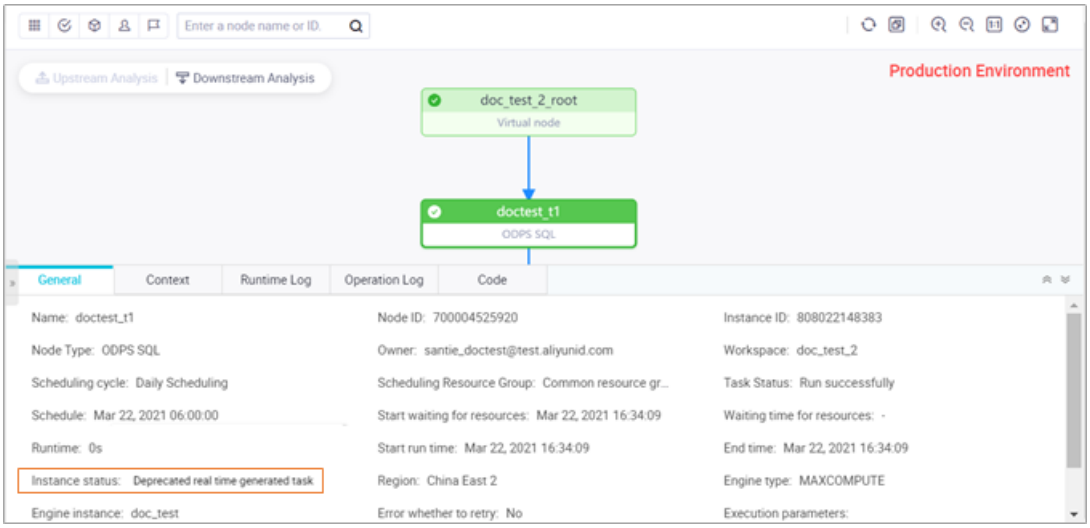
8.3.2. Configure immediate instance generation for a node

After you commit and deploy a node to the scheduling system, the system automatically generates an instance for the node and runs the node. You can set the Instance Generation Mode to Next Day or Immediately After Deployment when you configure scheduling properties for a node. If you set the Instance Generation Mode to Immediately After Deployment, the system immediately generates an instance for your node after the node is deployed. Then, you can view the status of the instance in Operation Center. This topic describes the rules for immediate instance generation and how to configure the immediate instance generation feature.

Rules for immediate instance generation

- If you commit and deploy a node before 23:30, the system immediately generates an instance for the node. During the period from 23:30 to 24:00, the immediate instance generation feature does not take effect.

- If the scheduled time for running your node is 10 or more minutes later than the time you commit and deploy your node, the system generates an instance for the node and runs the node. Then, you can view the instance in the instance list. For example, you commit and deploy your node at 18:00, and your node is scheduled to run at 18:30. In this case, the preceding situation occurs.
- If the scheduled time for running your node is less than 10 minutes from the time you commit and deploy your node, the system generates an instance whose running is complete for the node. The instance is an **expired instance that is generated in real time**. For example, you commit and deploy your node at 18:00, and your node is scheduled to run at 18:05. In this case, the preceding situation occurs. The expired instance is not actually run.



- If you commit and deploy a node after 23:30, the immediate instance generation feature does not take effect. In this case, you cannot find the instance generated for the node in the instance list. You can find the instance two days after the node is committed and deployed.
- If a node is an isolated node, no instances can be generated for the node. An isolated node is a node that has no dependent ancestor nodes.

  For example, you set the **Instance Generation Mode** parameter to **Next Day** for Node A and **Immediately After Deployment** for Node B. Node B is a descendant node of Node A. Then, the system immediately generates an instance for Node B but does not immediately generate an instance for Node A. In this case, Node B cannot find the instance of Node A based on the dependencies between Node A and Node B. As a result, Node B becomes an isolated node and cannot be scheduled to run.
- If you change the instance generation mode of a node from **Next Day** to **Immediately After Deployment**, instance generation for the node on the current day is affected.
  - If the node is scheduled to run 10 or more minutes after you change the time to deploy the node, instances generated for the node are deleted and replaced with the instances that are immediately generated.
  - If the node is scheduled to run within 10 minutes after you change the time to deploy the node, instances generated for the node are retained.
- You cannot set the instance generation modes of node groups to **Immediately After Deployment** because node groups do not support the immediate instance generation feature. Node groups include PAI nodes, do-while nodes, and for-each nodes that contain inner nodes.

Instance generation mode

Two instance generation modes are available for a node:

- **Next Day**

  If the node is committed and deployed before 23:30, instances are generated the next day. If the node is committed and deployed between 23:30 and 00:00, instances are generated the day after tomorrow.

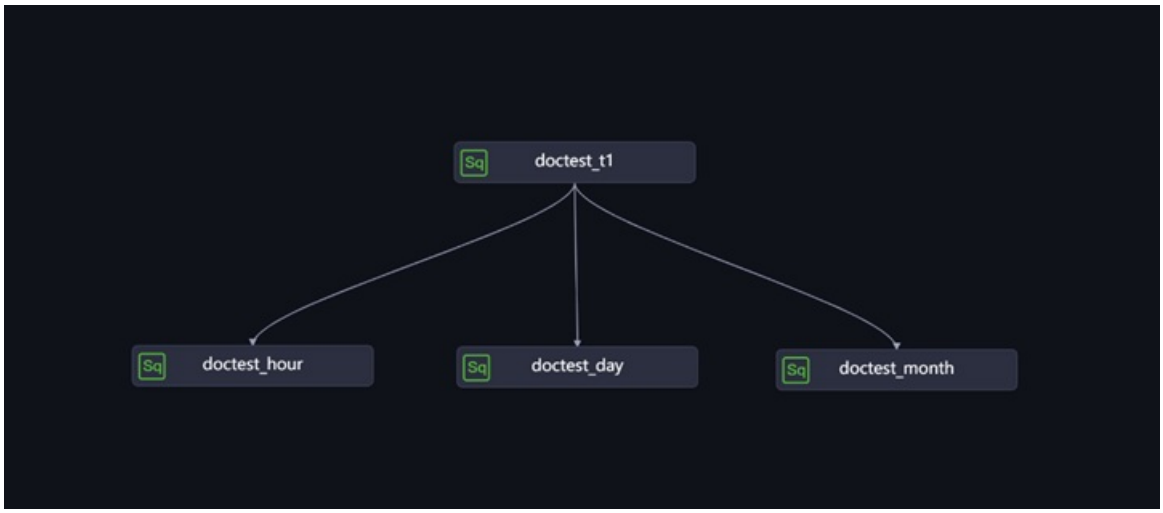
- **Immediately After Deployment**

  Instances are immediately generated after the node is committed and deployed. To generate instances for the node, the node must meet the rules for immediate instance generation. For more information, see Rules for immediate instance generation.

  > **Note** A node that is scheduled to run only 10 or more minutes after it is deployed can be actually run and generate data.

Scenario: The instance generation mode of Node A is Next Day, whereas that of the descendant nodes of Node A is Immediately After Deployment.

The immediate instance generation feature is typically used in the following scenario: The instance generation mode of Node A is Next Day, whereas that of the descendant nodes of Node A is Immediately After Deployment. The following figure shows the dependencies between Node A and its descendant nodes.



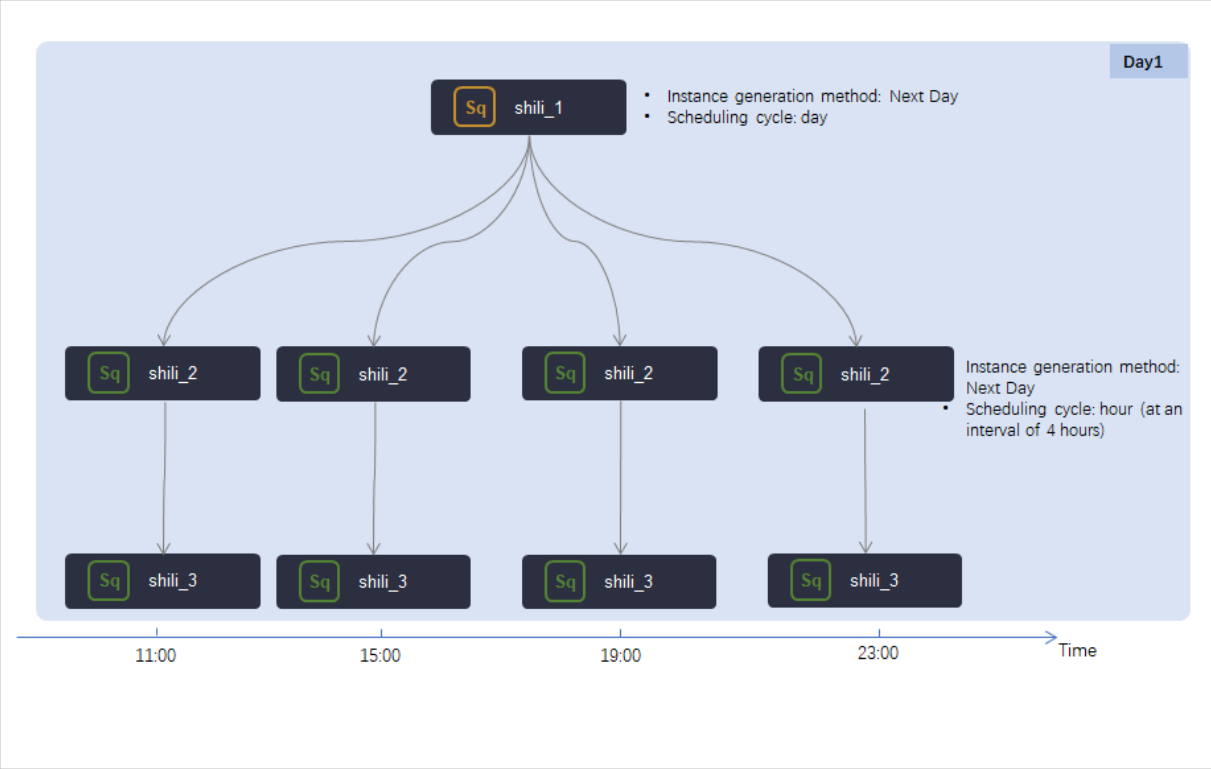
Based on the instance generation modes configured for Node A and its descendant nodes, the system immediately generates instances for the descendant nodes but generates an instance for Node A the next day. After the descendant nodes are committed and deployed, the instances immediately generated for the descendant nodes can run only if an instance is generated for Node A on the current day and the instance is successfully run. The following table describes the detailed situations involved in the preceding scenario and the impacts on the scheduling of the descendant nodes.

Situation	Impact on the scheduling of the descendant nodes	Summary
Both Node A and its descendant nodes are newly created on the current day. Instances are not generated for Node A when the descendant nodes are committed and deployed.	• The system immediately generates instances for the descendant nodes but generates an instance for Node A the next day. As a result, the instances of the descendant nodes become isolated after both Node A and its descendant nodes are committed and deployed. • If you select **Previous Cycle** and set **Depend On** to **Instances of Current Node** in the **Dependencies** section of the Properties tab for a descendant node of Node A, the descendant node becomes an isolated node and cannot be scheduled to run even if an instance is generated for Node A the next day. This is because the descendant node depends on an isolated instance that was generated on the previous day.	We recommend that you change the instance generation mode of Node A to **Immediately After Deployment**. This way, the system can automatically generate an instance for Node A after Node A is committed and deployed, and the descendant node of Node A can be scheduled to run.

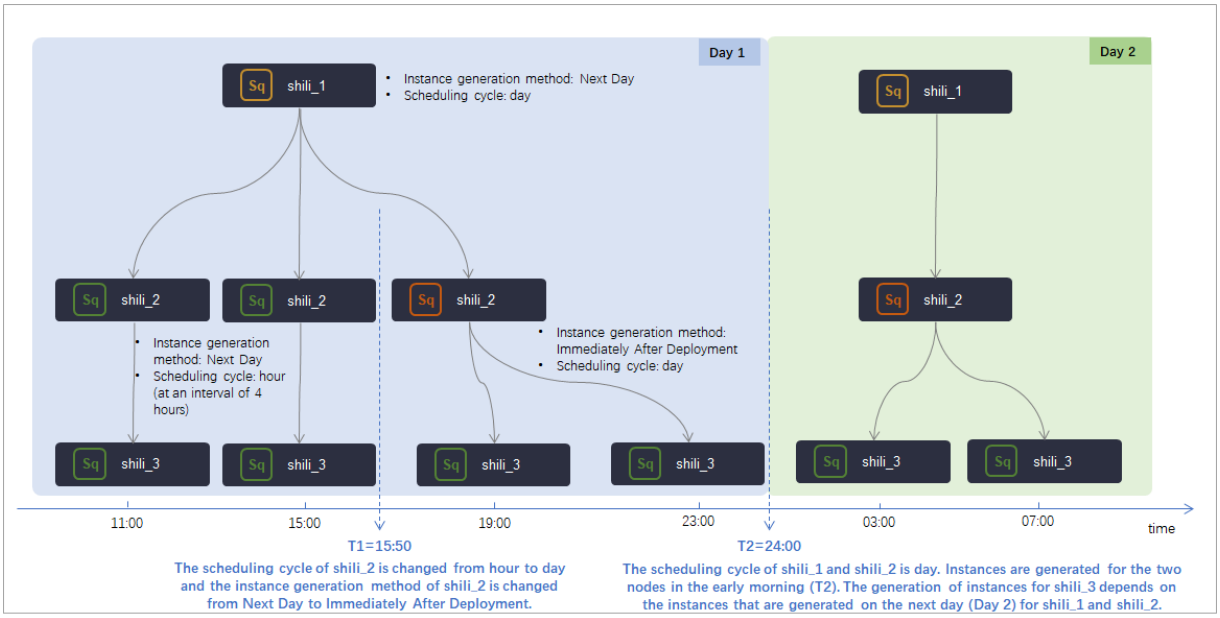
An instance is generated for Node A, and a descendant node that uses the immediate instance generation mode is created on the current day.
An instance has been generated for Node A when the descendant node is committed and deployed.
	• The system immediately generates an instance for the descendant node after the descendant node is committed and deployed. The generated instance depends on the instance of Node A.
	• If the **scheduling properties** of the descendant node contain the following settings, different situations occur:
	  ◦ The descendant node is scheduled by **hour** or **minute**. In this case, the instance generated for the descendant node depends on the instance that has been generated for Node A.
	  ◦ **Previous Cycle** is selected and **Depend On** is set to **Instances of Current Node** for the descendant node. In this case, the first instance generated for the descendant node has no dependent ancestor instances, but subsequent instances generated for the descendant node can depend on their previous-cycle instances. The descendant node can be normally scheduled to run.
		To run a node that is configured to depend on its previous-cycle instances, you must make sure that the instances of the node are normally run on the previous day.

Scenario: The instance generation mode of a node is changed from Next Day to Immediately After Deployment, and the scheduling cycle of the node is changed from hour to day.

In a workflow that is committed and deployed, the instance generation mode of Node A is Next Day, and Node A is scheduled by day, such as shili_1 in the following figure. The instance generation mode of the descendant nodes of Node A is Next Day, and these descendant nodes are scheduled by hour, such as shili_2 and shili_3 in the following figure.



Instances are generated and run for Node A and its descendant nodes the next day. After the instances are run for a period of time, the scheduling cycle of shili_2 is changed from hour to day, and the instance generation mode of shili_2 is changed to Immediately After Deployment at the T1 time point. The following figure shows the changes.



After the changes, the following situations occur:

- After you change the instance generation mode and scheduling cycle of shili_2 at T1 and commit

shili_2, the instances that are generated for shili_2 before T1 and used to run shili_2 after T1 are deleted. In addition, a new instance is generated for shili_2 whose scheduling cycle is changed to day.

- After shili_2 is committed and deployed at T1, shili_3, the descendant node of shili_2, depends on shili_2 whose scheduling cycle is changed to day.
- If **Previous Cycle** is selected and **Depend On** is set to **Instances of Current Node** for shili_2, the first instance generated after the change depends on the instances generated for shili_2 whose scheduling cycle is hour in the previous cycle.

8.3.3. Schedule a node by minute

If a node is scheduled by minute, the node is automatically run once `every N minutes` within a specific period every day.

Limits

The minimum time interval for a node that is scheduled by minute is *5* minutes.

Configuration example

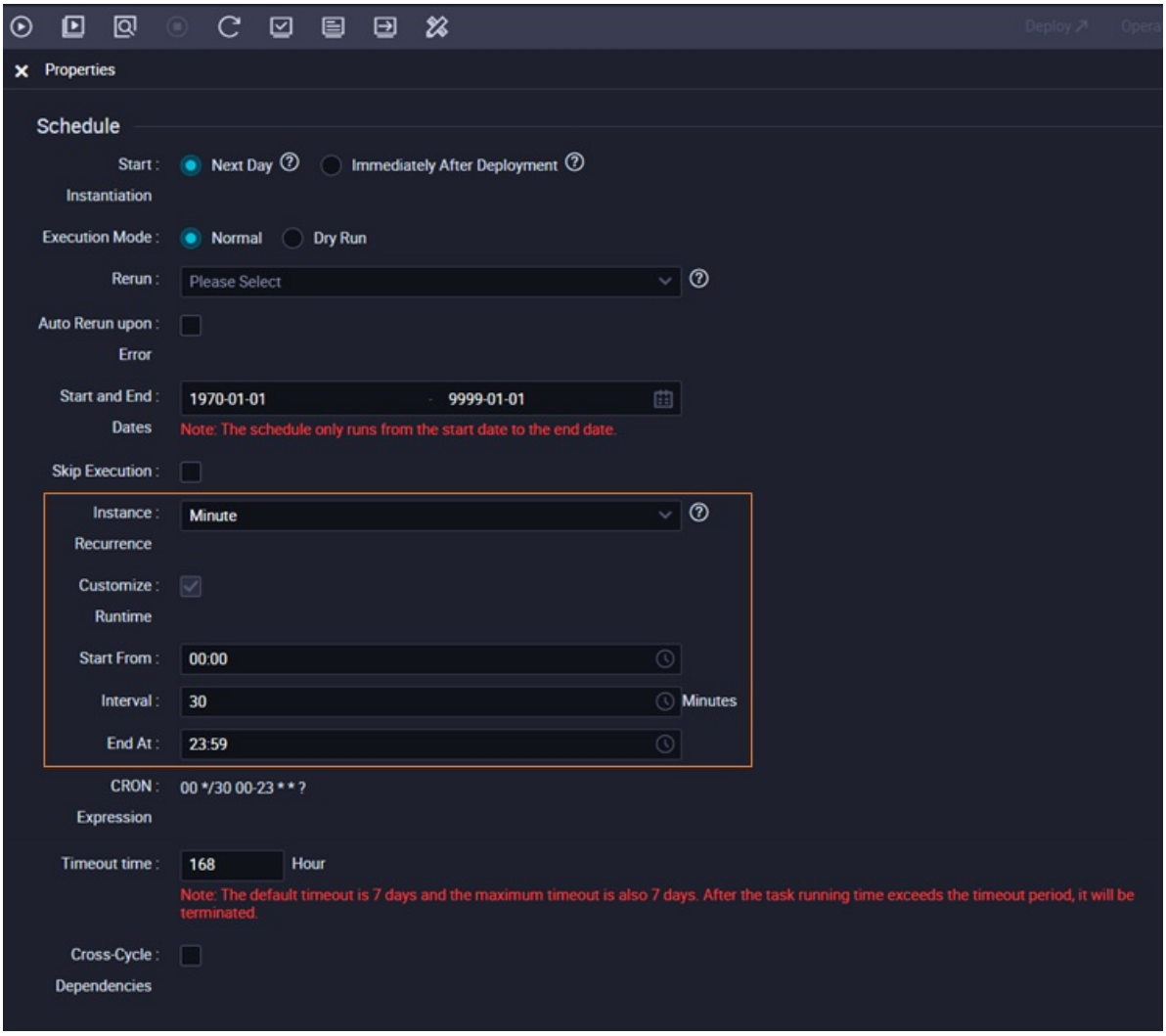
- **Configuration method**

  On the DataStudio page, create a node and go to the configuration tab of the node. Click the **Properties** tab in the right-side navigation pane. In the **General** section of the **Properties** tab, configure the scheduling period for the node.

- **Scenario**

  The following figure shows how to configure a node that is scheduled to run once every 5 minutes in the time period from `00:00` to `23:59` every day.

  > **Note** The corn expression is automatically generated based on the time you select and cannot be changed.



8.3.4. Schedule a node by hour

If a node is scheduled by hour, the node is automatically run once `every N hours` within a specific period every day. For example, a node is run once every hour from 00:00 to 03:00 every day.

Limits

You can schedule a node to run by hour only on the hour during the specified period. For example, you cannot schedule a node to run once every hour during the period from 00:05 to 23:59.

Precautions

- The period is a left-closed, right-closed interval. For example, if a node is scheduled to run once every hour during the period from 00:00 to 03:00, the scheduling system generates four instances for the node every day, and the four instances are run at 00:00, 01:00, 02:00, and 03:00 in sequence.
- You can schedule a node to run at specified intervals within a specific period every day. You can also schedule the node to run at specified points in time on the hour every day.
- The time at which a node is actually run may be different from the configured time due to reasons such as insufficient resources.

Configuration example

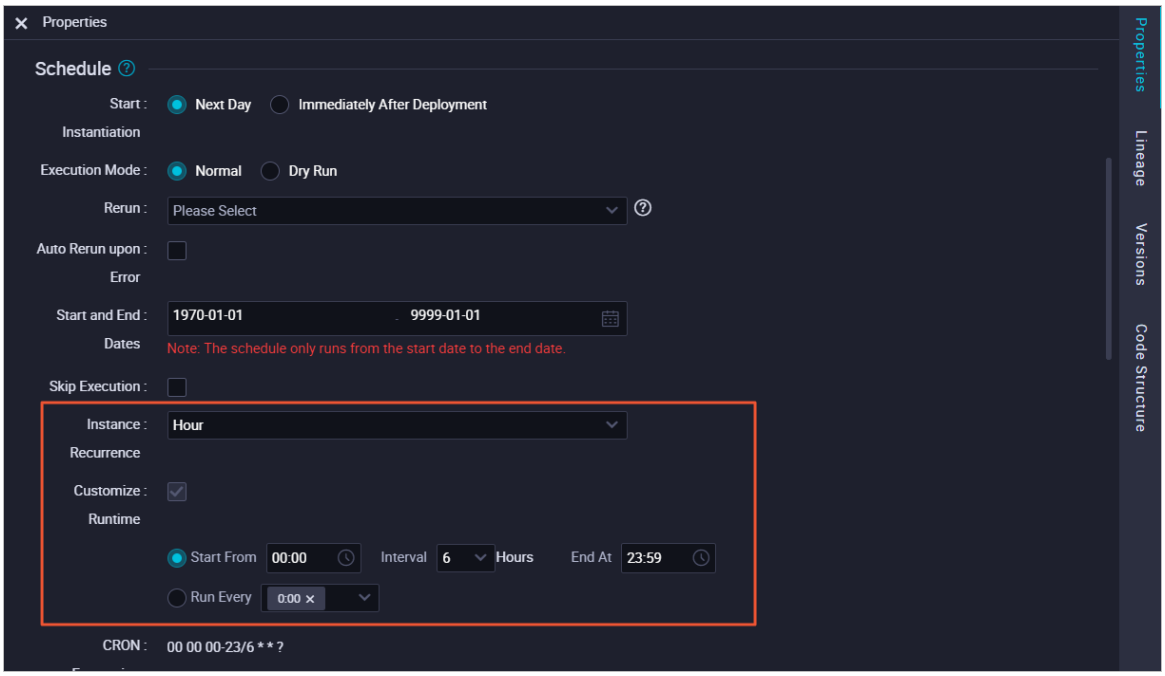
- **Configuration method**

  On the DataStudio page, create a node and go to the configuration tab of the node. Click the **Properties** tab in the right-side navigation pane. In the **General** section of the **Properties** tab, configure the scheduling period for the node.

- **Scenario**
  - **Configuration details**

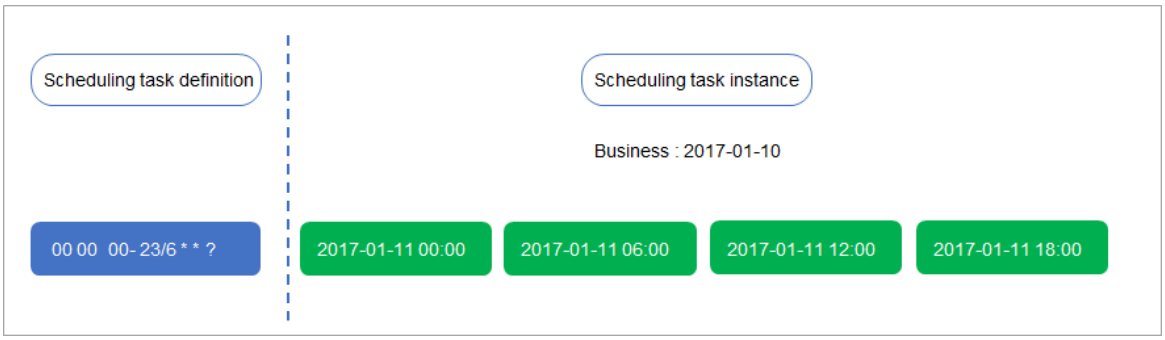
    The following figure shows how to configure a node that is scheduled to run every 6 hours in the time period from `00:00` to `23:59` every day.

    > **Note** The corn expression is automatically generated based on the time you select and cannot be changed.

  - **Scheduling details**

    The following figure shows that the scheduling system generates four instances every day and runs the instances at 00:00, 06:00, 12:00, and 18:00 in sequence.

8.3.5. Schedule a node by day

If a node is scheduled by day, the node is automatically run once at a specified point in time every day. If you create an auto triggered node that is scheduled by day, the node is scheduled to run at 00:00 every day by default. You can change the time based on your business requirements. For example, you can set the time to 13:00.

Configuration example

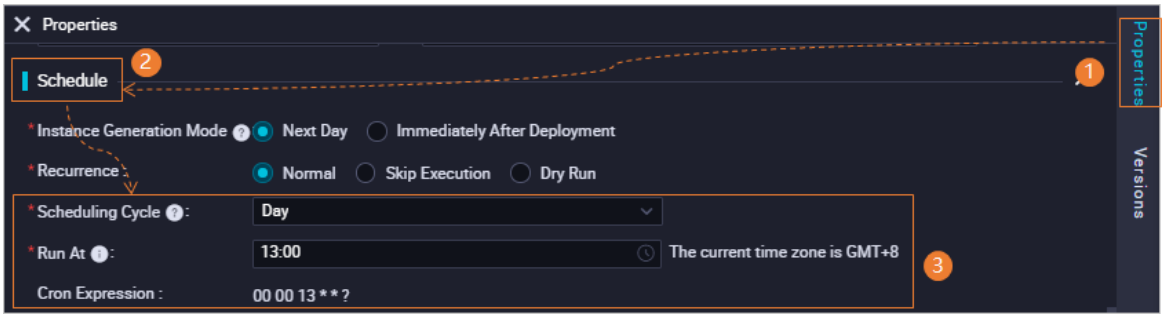
- **Configuration method**

  On the DataStudio page, create a node and go to the configuration tab of the node. Click the **Properties** tab in the right-side navigation pane. In the **General** section of the **Properties** tab, configure the scheduling period for the node.

- **Scenario**
  - **Configuration details**
    - You create an import node, an analytics node, and an export node,
    - and you set the point in time at which the nodes are run every day to `13:00` .
    - The analytics node depends on the import node, and the export node depends on the analytics node.

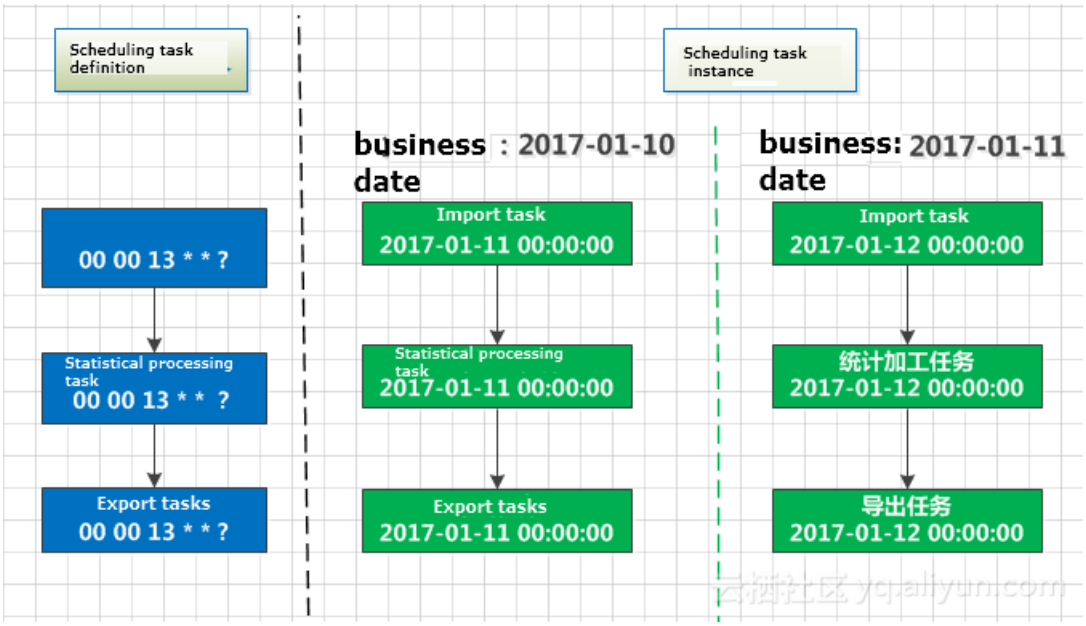
    The following figure shows how to configure a node that is scheduled to run at 13:00 every day.

    > **Note** The corn expression is automatically generated based on the time you select and cannot be changed.



- **Scheduling details**

  The following figure shows that the scheduling system automatically generates and runs instances for the nodes.



> **Note**
>
> - The following prerequisites must be met before a node is run:
>   - The ancestor node of the node is successfully run.
>   - The point in time at which the node is scheduled to run has arrived.
>
>   Both prerequisites must be met and they have no specific chronological order.
> - The default point in time at which the node is scheduled to run is a random time from `00:00` to `00:30`.

8.3.6. Schedule a node by week

If a node is scheduled by week, the node is automatically run at a specified time on specific days every week.

Precautions

To ensure that the descendant instances of a node that is scheduled by week can be run as expected during the time period that is not the scheduled time of the node, the system generates dry-run auto triggered instances for the node. That means the scheduling system does not actually run the node but directly returns a success response when the scheduled time of the node arrives. This way, the descendant nodes of the node can be run as expected and resources are not occupied by the node.

Configuration example

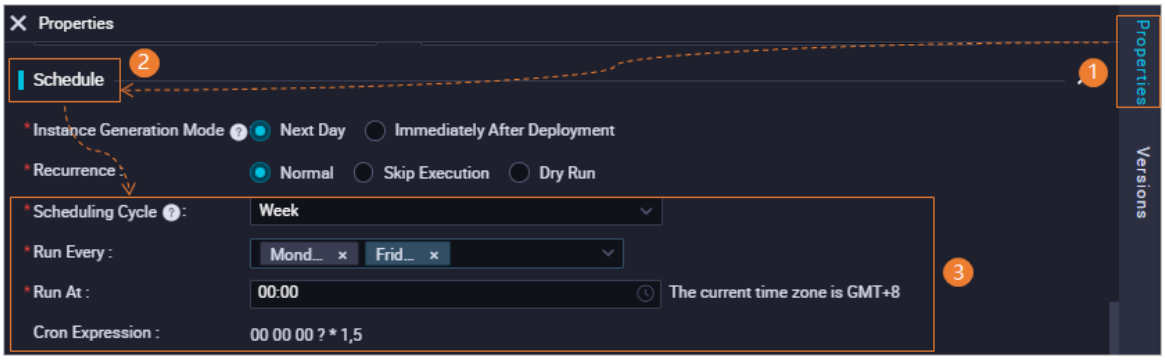
- **Configuration method**

On the DataStudio page, create a node and go to the configuration tab of the node. Click the **Properties** tab in the right-side navigation pane. In the **General** section of the **Properties** tab, configure the scheduling period for the node.

- Scenario
  - Configuration details

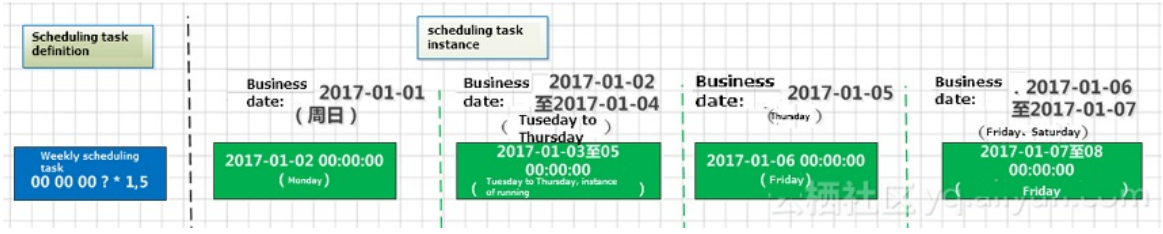
    If the node is scheduled to run at a specified time every Monday and Friday, the scheduling system runs the instances that are generated every Monday and Friday. The scheduling system does not run the instances that are generated every Tuesday, Wednesday, Thursday, Saturday, or Sunday and directly returns a success response for the instances. The following figure shows the details about the configurations.

    > **Note** The corn expression is automatically generated based on the time you select and cannot be changed.



  - Scheduling details

    The following figure shows that the scheduling system automatically generates and runs instances for the nodes.



8.3.7. Schedule a node by month

If a node is scheduled by month, the node is automatically run at a specified time on specific days every month.

Precautions

- To ensure that the descendant instances of a node that is scheduled by month can be run as expected during the time period that is not the scheduled time of the node, the system generates dry-run auto triggered instances for the node. That means the scheduling system does not actually run the node but directly returns a success response when the scheduled time of the node arrives. This way, the descendant nodes of the node can be run as expected and resources are not occupied by the node.
- You can set the **Run Every** parameter to **Last Day of the Month**. This way, the node is run on the

last day of every month.

Configuration example

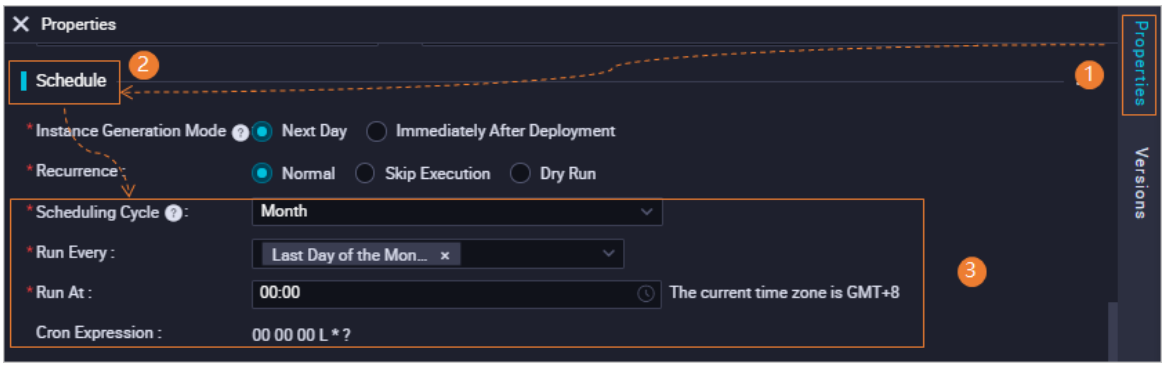
- **Configuration method**

  On the DataStudio page, create a node and go to the configuration tab of the node. Click the **Properties** tab in the right-side navigation pane. In the **General** section of the **Properties** tab, configure the scheduling period for the node.

- **Scenario**
  - **Configuration details**

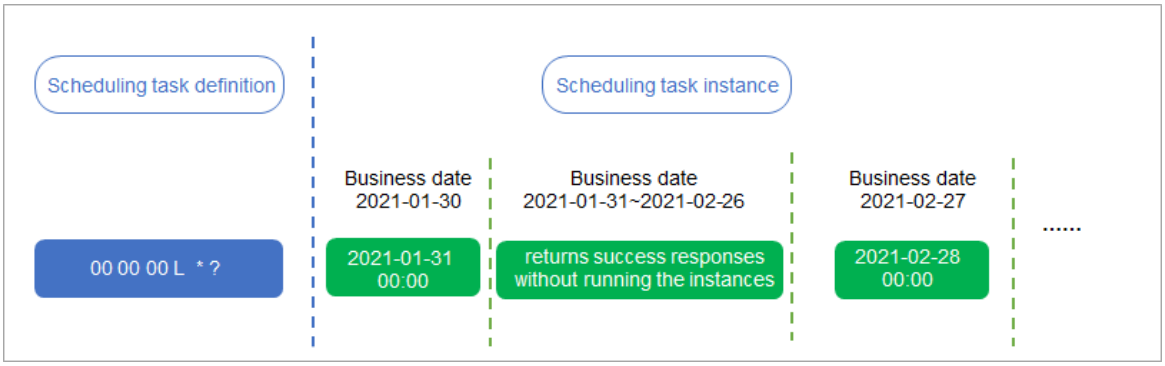
    If the node is scheduled to run on the last day of each month, the scheduling system runs the instances that are generated on the last day of each month. The scheduling system generates dry-run instances for the node on each of the rest days every month and directly returns a success response for the instances. The following figure shows the details about the configurations.

    > **Note** The corn expression is automatically generated based on the time you select and cannot be changed.

  - **Scheduling details**

    The following figure shows that the scheduling system automatically generates and runs instances for the nodes.

8.3.8. Schedule a node by year

If a node is scheduled by year, the node is automatically run at a specified point in time on specific days every year.

Precautions

To ensure that the descendant instances of a node that is scheduled by year can be run as expected during the time period that is not the scheduled time of the node, the system generates dry-run auto triggered instances for the node. That means the scheduling system does not actually run the node but directly returns a success response when the scheduled time of the node arrives. This way, the descendant nodes of the node can be run as expected and resources are not occupied by the node.

Configuration example

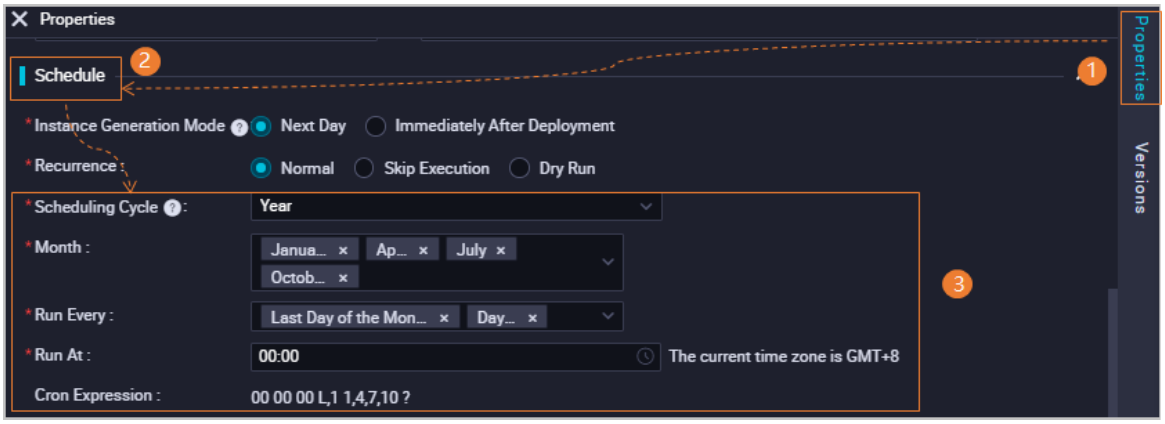
- **Configuration method**

  On the DataStudio page, create a node and go to the configuration tab of the node. Click the **Properties** tab in the right-side navigation pane. In the **General** section of the **Properties** tab, configure the scheduling period for the node.

- **Scenario**
  - **Configuration details**

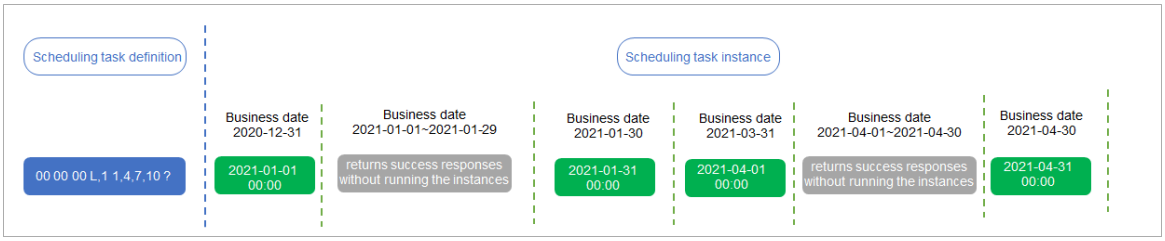
    If the node is scheduled to run at a specified point in time on the first and last days of January, April, July, and October each year, the scheduling system runs the instances that are generated on these days each year. The scheduling system generates dry-run instances for the node on each of the rest days in the year and directly returns a success response for the instances. The following figure shows the details about the configurations.

    > **Note** The corn expression is automatically generated based on the time you select and cannot be changed.

  - **Scheduling details**

    The following figure shows that the scheduling system automatically generates and runs instances for the nodes.

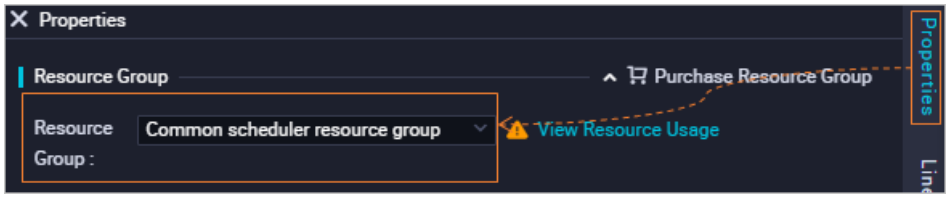
    

8.4. Configure a resource group

To run an auto triggered node, you must configure a resource group for scheduling. You can select a resource group in the Resource Group section of the Properties panel for the required node.

Configure a resource group

Go to the configuration tab of a node. Click the **Properties** tab in the right-side navigation pane. In the Properties panel, configure the resource group for the node in the **Resource Group** section.



1. Select the resource group you want to use from the **Resource Group** drop-down list. By default, **Common scheduler resource group** is selected.

   > **Note** You can go to the **Scheduling Settings** tab of the Settings page to change the default resource group for node scheduling. For more information, see Configure scheduling settings.

2. Click **View Resource Usage** next to the resource group to view its usage in different periods on the previous day. You can select an appropriate resource group for node scheduling based on the resource group usage.

> **Note**
>
> - If you select Common scheduler resource group, nodes may wait for resources in peak hours. We recommend that you change the scheduled time or use an exclusive resource group.
> - If the existing resource groups do not meet your requirements, click **Purchase Resource Group** to purchase an exclusive resource group. For more information, see Create and use an exclusive resource group for scheduling.

8.5. Configure scheduling dependencies

8.5.1. Logic of same-cycle scheduling dependencies

Nodes in a workflow in DataWorks are run in sequence based on the scheduling dependencies configured for each node. This ensures that business data is generated in an effective and timely manner. This topic describes the principles for configuring same-cycle scheduling dependencies and how scheduling dependencies work.

Reasons for configuring scheduling dependencies

Scheduling dependencies define the relationships between nodes. After you configure scheduling dependencies for a node, the node is run only after its ancestor node is run as expected.

After the execution of a node is successful, the node generates the output data. Then, its descendant node extracts that data from the node. This mechanism ensures that a node can obtain valid data from its ancestor node. This also prevents the node from extracting output data from its ancestor node before the execution of the ancestor node is successful.

We recommend that you plan and configure scheduling dependencies for nodes based on the lineage of the table data of each node. Make sure that the following principles are met:

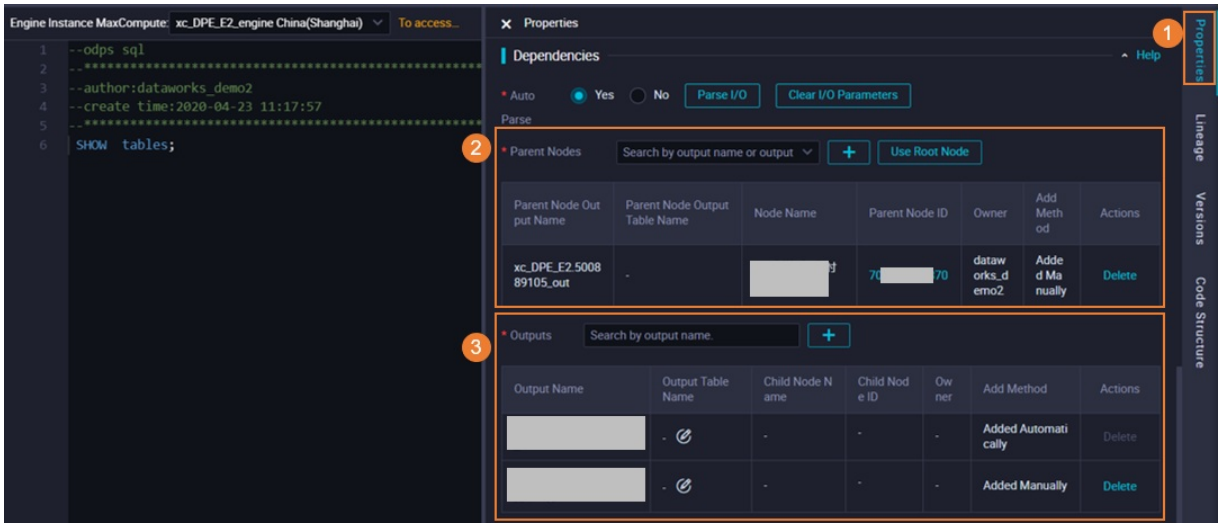
- A table is generated by only one node, and the table must be configured as the output of the node.

  > **Note**
  > - The system automatically adds the table generated by an SQL node to the output of the SQL node based on the automatic parsing feature.
  > - You must manually add the table generated by a batch sync node to the output of the node. The name of the table is in the projectname.tablename format. This way, the output table of the node is configured as the input table of its descendant node based on the automatic parsing feature when the descendant node is run.

- The output of a node must be configured as the input of its descendant node, which forms dependencies between nodes.

> **Note** For a node whose table data has no lineage, you can plan and configure scheduling dependencies for the node based on the upstream and downstream relationships between nodes in a workflow. The configuration principles and results must comply with those of a node whose table data has a lineage.

The scheduling dependencies of a node are configured in the **Properties** panel of the node. You must set the **Parent Nodes** and **Output** parameters for the node.



Scheduling dependencies can be automatically or manually configured for a node.

- In most cases, the system can automatically identify input and output commands such as `SELECT` and `INSERT` based on the standard code that you developed for a node. The system can also identify the lineage of table data based on the code. Then, the system automatically configures scheduling dependencies for the node based on the automatic parsing feature and the identified lineage.

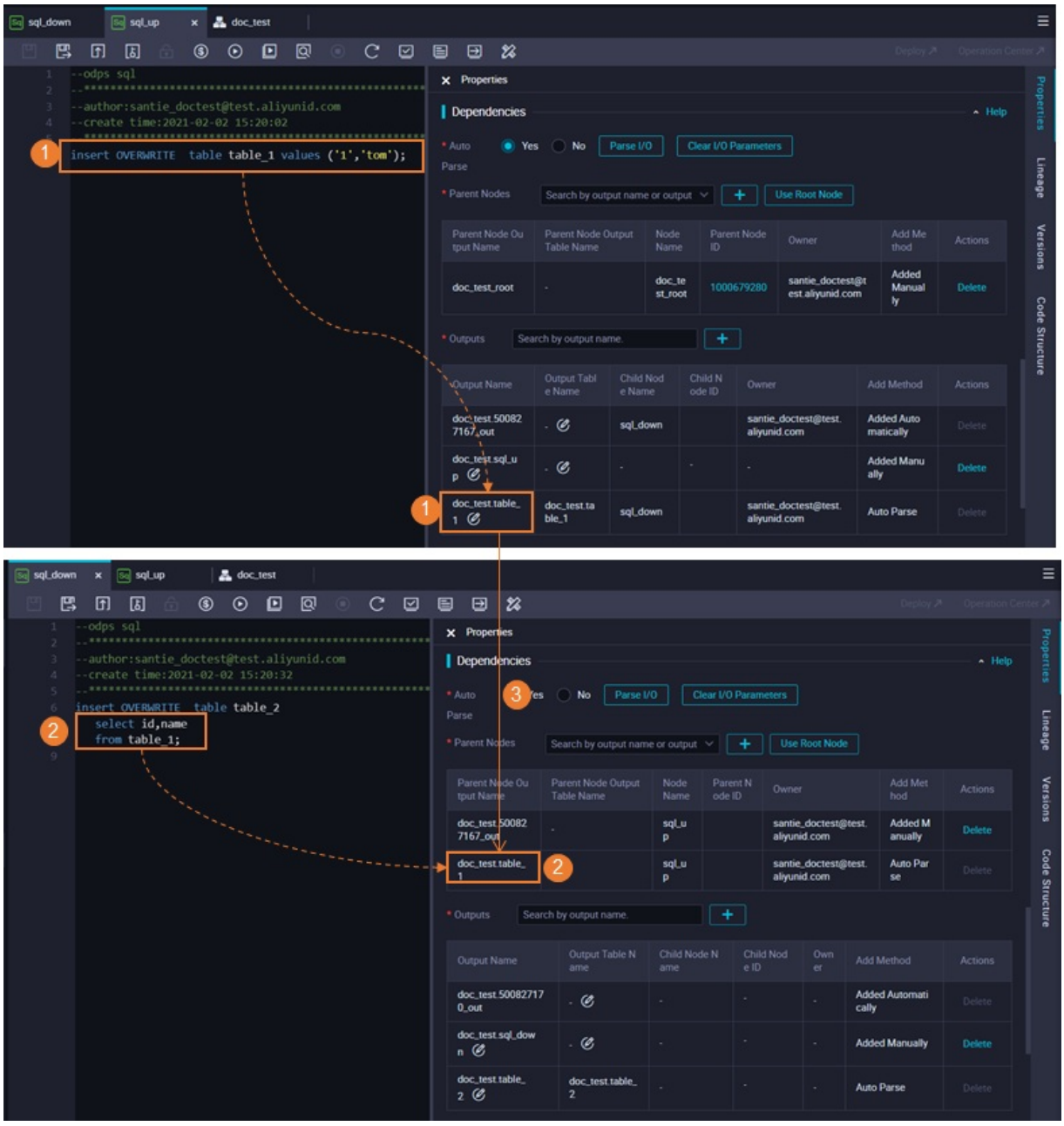
- In special cases, you can manually configure scheduling dependencies for a node. For example, the scheduling dependencies configured for a node contain a table that is not generated by an auto triggered node, such as a table uploaded from your on-premises machine. In this case, you can manually modify the scheduling dependencies.

When you commit a node, the system checks whether the scheduling dependencies configured for the node are consistent with the data lineage in the code developed for the node. If they are inconsistent, the system displays an error message. In this case, you can determine whether to modify the scheduling dependencies based on actual situations.

Automatic parsing

For an SQL node, the system can automatically determine the scheduling dependencies of the node based on the code developed for the node. Then, the system automatically sets the **Output** or **Parent Nodes** parameters for the scheduling dependencies of the node.

Principles for adding scheduling dependencies based on automatic parsing



- If the code developed for a node contains output commands such as `INSERT` and `CREATE`, the system automatically parses the commands and adds the table generated by the node to **Output** for the node.
- If the code developed for a node contains input commands such as `SELECT`, the system automatically parses the commands and adds the input table to **Parent Nodes** for the node.
- The output of a node is configured as the input of its descendant node. This way, a scheduling dependency is established between the nodes based on their data lineages.

In normal cases, the results of automatic parsing are consistent with data lineages. If you commit a node, the system checks whether the scheduling dependencies configured for the node are consistent with data lineages. If they are inconsistent, the system displays an error message. In this case, you can use one of the following methods to resolve the issue:

- If the scheduling dependencies of a node do not contain a table that is not generated by an auto

triggered node, you must check whether the scheduling dependencies are correctly configured for the node.

- If the scheduling dependencies of a node contain a table that is not generated by an auto triggered node, you must manually delete the dependency for the node.

Requirements and principles for code development

Automatic parsing enables the system to automatically identify the scheduling dependencies of a node based on the code that you developed for the node. Therefore, we recommend that you strictly comply with the following requirements when you develop data:

- Requirements for code development: One node generates only one table, and a table is generated by only one node.
- Requirements for node creation: The name of a node must be consistent with that of the table that is to be generated by the node.
- Requirements for scheduling configurations: The table generated by a node must be added to Output for the node.

Manually configure scheduling dependencies between nodes

DataWorks allows you to manually modify the **Parent Nodes** and **Output** parameters for a node during the code development for the node. If the scheduling dependencies automatically generated by the system for your node do not meet your business requirements, you can manually modify the dependencies.

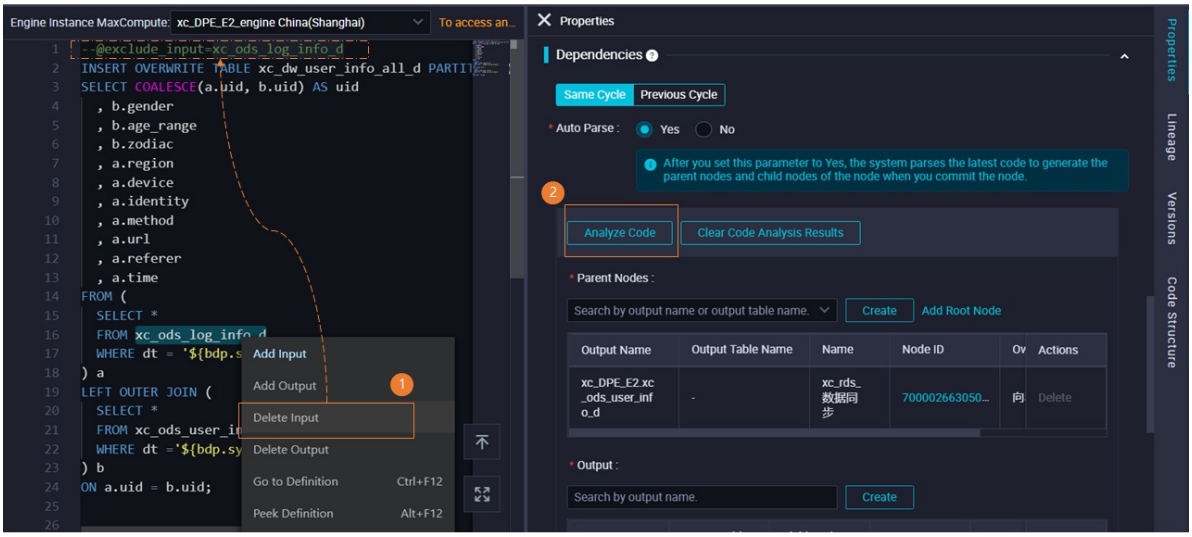
Scenarios

Scheduling dependencies ensure that a node can successfully obtain the table data generated by its ancestor node that is scheduled to run. However, if the ancestor node of a node is not scheduled to run, the system cannot monitor whether the ancestor node has generated the latest table data. If the table specified in the SELECT statement of the code for a node is not generated by an auto triggered node and the table name is automatically added to Parent Nodes for the node, you must manually delete the dependency for the node. Tables that are not generated by auto triggered nodes include the following types:

- Tables uploaded from on-premises machines to DataWorks
- Dimension tables
- Tables that are not generated by nodes scheduled by DataWorks
- Tables generated by manually triggered nodes

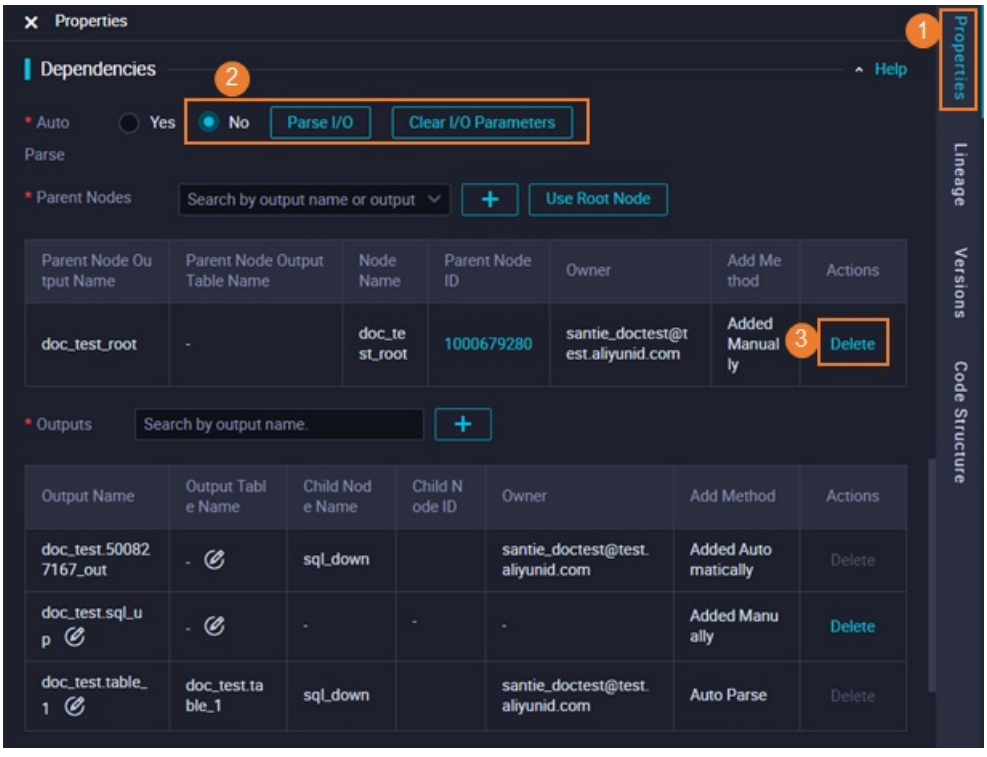
Configuration methods

- Delete a scheduling dependency in the code editor of a node



If the SELECT statement in the code of a node specifies a table that is not generated by an auto triggered node, you can delete the dependency for the node. Specifically, you can go to the code editor of the node, right-click the name of the table that you want to remove from the input, and then click Delete Input. The preceding figure shows the process. You can also add a rule as a comment at the top of the code. This way, the system does not automatically parse the dependency based on the rule.

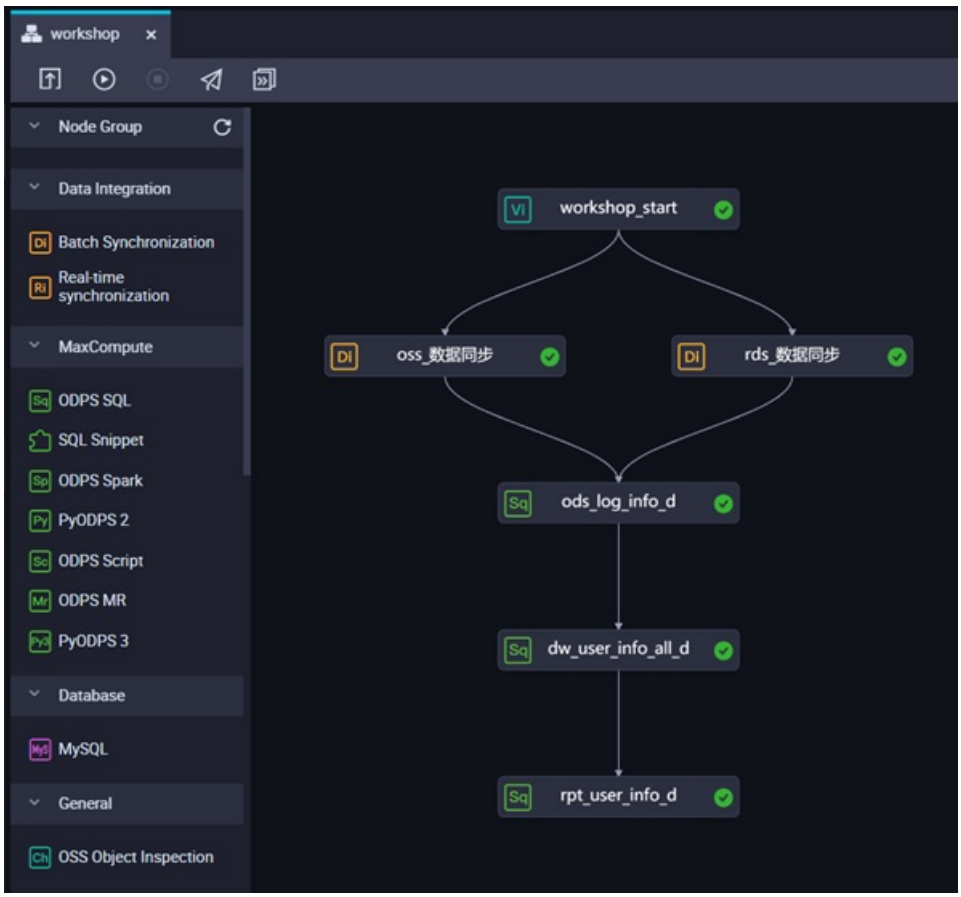
- Delete a scheduling dependency in the Properties panel of a node



If the SELECT statement in the code of a node specifies a table that is not generated by an auto triggered node, you can manually remove the table from **Parent Nodes** for the node. Specifically, you can go to the Properties panel of the node, set Auto Parse to No, and then manually perform the operation.

Configuration by drawing lines to connect nodes

DataWorks allows you to specify the relationships between nodes by drawing lines to connect nodes on the editing pages of workflows. After the nodes are connected, the system automatically adds scheduling dependencies for each node based on the connections.



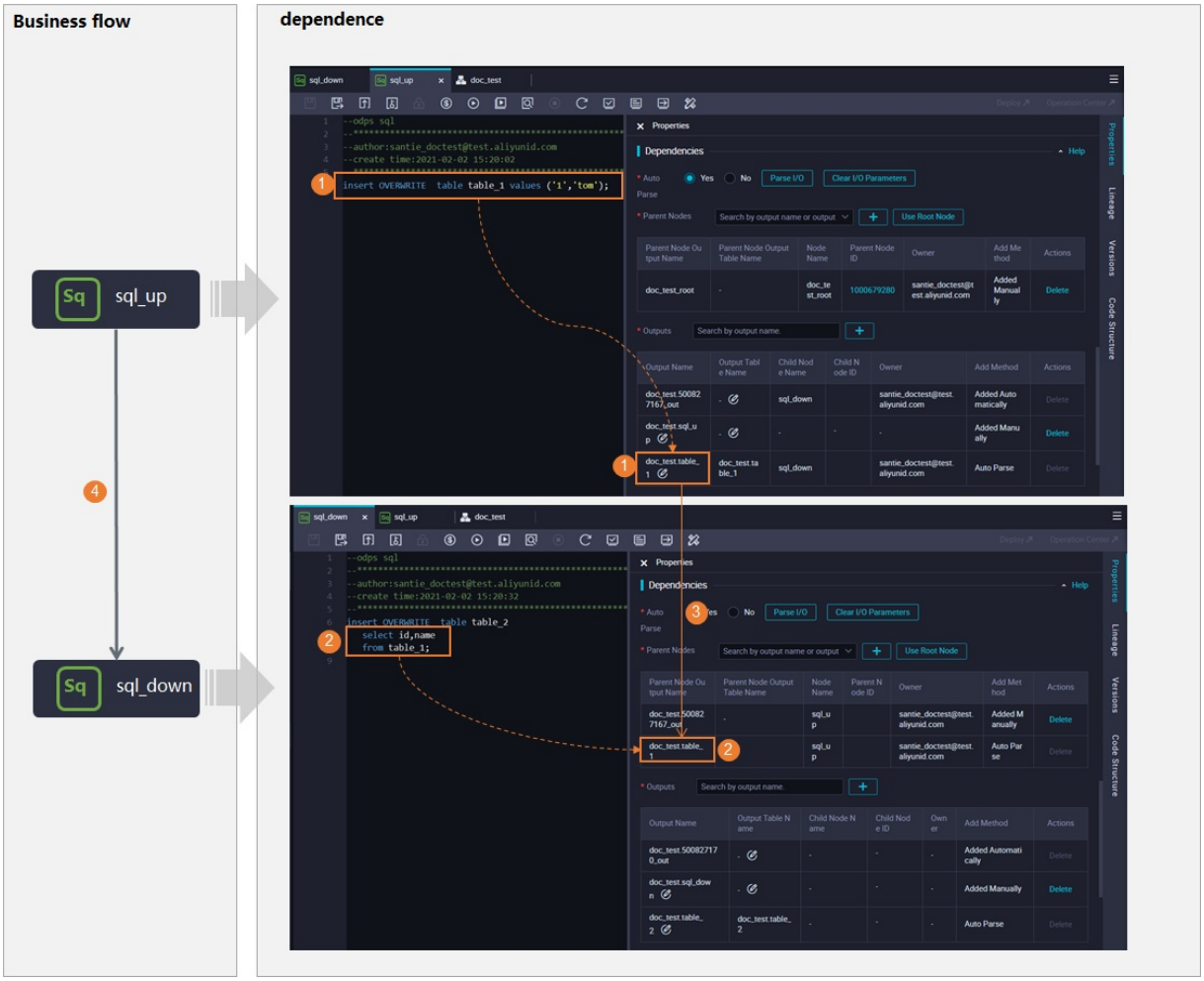
After all nodes are created, the system automatically adds an **output** whose name is suffixed with **_out** for each node. When you connect nodes by drawing lines, the system adds an output whose name is suffixed with **_out** to the input of each descendant node.

Scenario

After you create a workflow, you can connect nodes by drawing lines on the configuration tab of the workflow to configure scheduling dependencies for each node based on your business requirements. During subsequent code development, you can add or modify scheduling dependencies for each node manually or by using the automatic parsing feature. This way, all nodes in the workflow can be configured with correct scheduling dependencies.

Case study

In this section, an example is provided to demonstrate how scheduling dependencies work.



The preceding figure shows the following information:

- The table generated by a node must be added to **Output** for the node. If the code developed for the node contains output commands such as `INSERT`, the system automatically parses the commands and adds the table to **Output** for the node.
- The input of a node must be added to **Parent Nodes** for the node. If the code developed for the node contains input commands such as `SELECT`, the system automatically parses the commands and adds the table to **Parent Nodes** for the node.
- The output of a node is configured as the input of its descendant node. This way, a scheduling dependency is established between the nodes based on their data lineages.

Then, the system runs the ancestor node first based on the established scheduling dependency. After the execution of the ancestor node is successful, the system begins to run its descendant node.

The preceding process indicates that the following principles must be met when you configure scheduling dependencies for each node:

- For a node that has upstream and downstream relationships, the **Output** of the ancestor node of the node must be configured as the **Parent Nodes** of the node. This helps establish scheduling dependencies between nodes.
- The value of the **Output Table Name** and **Node ID** parameters of **Parent Nodes** configured for a descendant node must be unique. This also indicates that the output names of all nodes must be unique. Otherwise, the descendant node cannot find its ancestor node based on the two pieces of

information and obtain the data generated by the ancestor node.

Instructions on configuring scheduling dependencies

For more information about how to configure scheduling dependencies in common scenarios, see Configure same-cycle scheduling dependencies.

For more information about how to configure scheduling dependencies in other typical scenarios, see the following topics:

- Scenario 1: Configure scheduling dependencies for batch synchronization nodes in a workflow
- Scenario 2: Configure scheduling dependencies for a node that depends on last-cycle instances
- Scenario 3: Configure scheduling dependencies for nodes across workflows or workspaces

FAQ

- When I commit Node A, the system reports an error that the output name of the dependent ancestor node of Node A does not exist. What do I do?
- When I commit a node, the system reports an error that the input and output of the node are not consistent with the data lineage in the code developed for the node. What do I do?
- Scheduling dependencies

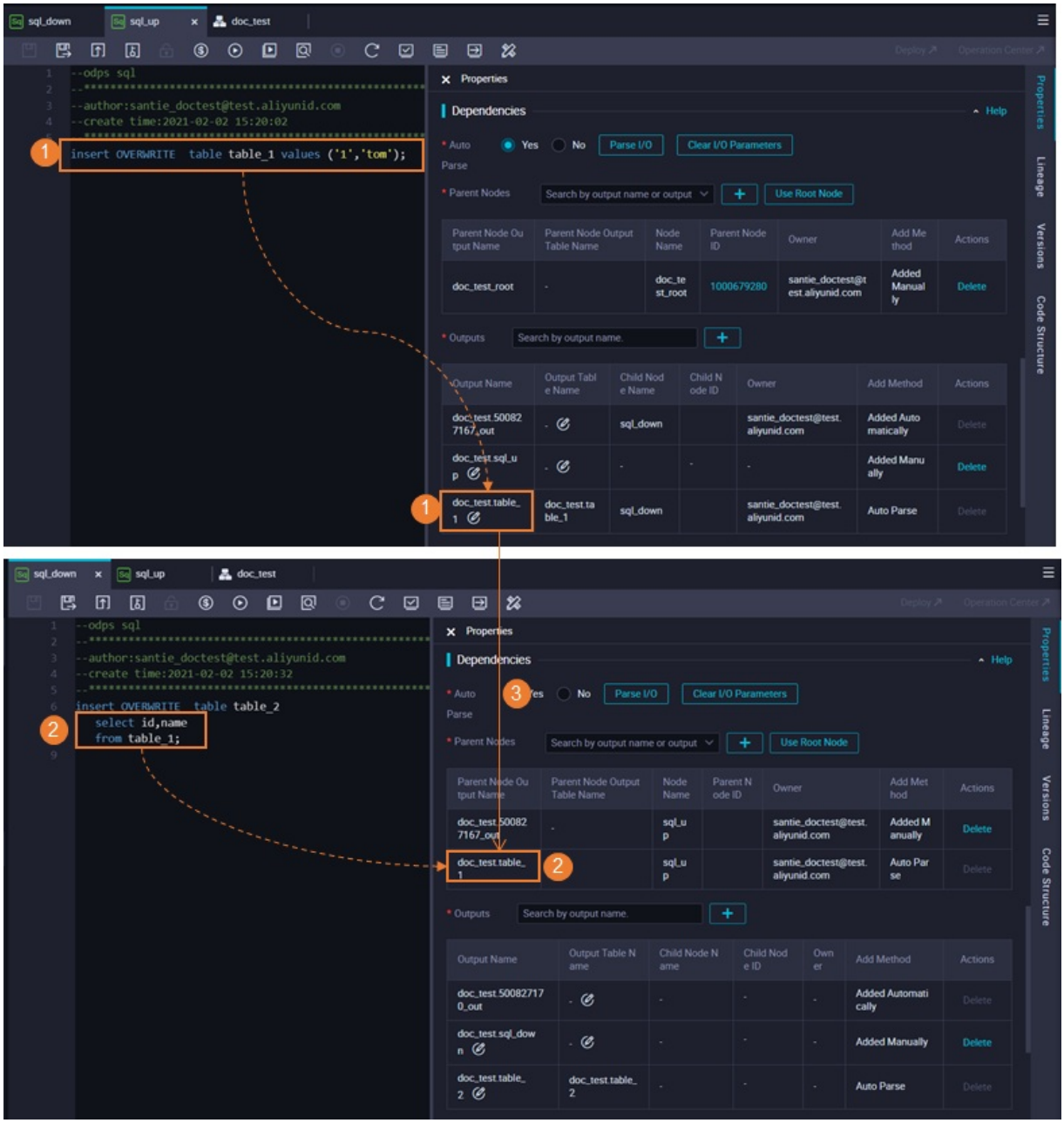
8.5.2. Configure same-cycle scheduling dependencies

Scheduling dependencies are fundamental to the establishment of orderly workflows. You must configure correct dependencies between nodes to ensure that business data is produced effectively and in time. This topic provides instructions to configure scheduling dependencies.

Background information

The configurations of scheduling dependencies include **Parent Nodes** and **Output**.

- **Parent Nodes**: the ancestor node of the current node. The current node can start to run only after its ancestor node is successfully run.
- **Output**: the output of the current node. You can search for the current node by its output name and configure the current node as an ancestor node of another node.

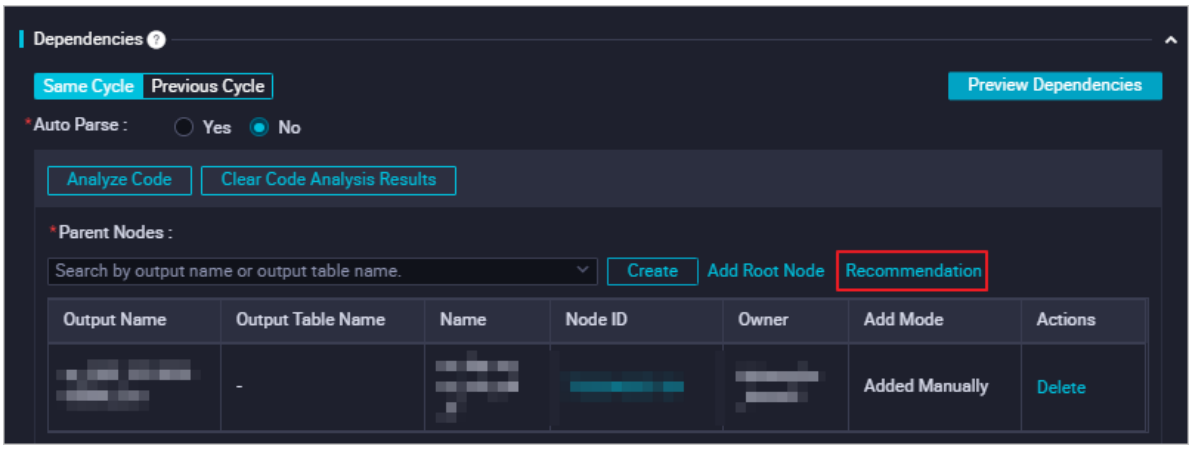


You can use one of the following methods to configure scheduling dependencies for an auto triggered node in DataStudio:

- On the configuration tab of the workflow to which the auto triggered node belongs, draw lines to connect the nodes in the workflow. For more information, see Configuration by drawing lines to connect nodes.
- In the Dependencies section of the Properties tab on the configuration tab of the auto triggered node, click the Same Cycle tab, and enter the name of a node or the name of the output table of the node in the Parent Nodes field to add the node as an ancestor node of the auto triggered node. You can click **Analyze Code** and modify the analysis results based on your business requirements. For more information, see Manual configuration.
- In the Dependencies section of the Properties tab on the configuration tab of the auto triggered node, select Yes for Auto Parse to enable the automatic parsing feature. When you commit the node, the system parses the code of the node to obtain the dependencies of the node and configures dependencies for the node in the Parent Nodes section. For more information, see

Configuration based on the automatic parsing feature.

- In the Dependencies section of the Properties tab on the configuration tab of the auto triggered node, select No for Auto Parse and click Recommendation. In the Recommended Parent Nodes dialog box, the system provides the recommended ancestor nodes for the auto triggered node based on the data lineage in the code of the auto triggered node.



> **Note** The data lineage is updated `one day later than` the scheduling time of the auto triggered node. As a result, the recommendation of ancestor nodes may be delayed.

After the configuration is complete, you can preview the scheduling dependencies of the node. For more information, see Preview scheduling dependencies.

For more information about the configurations in common scenarios, see Instructions to configure scheduling dependencies in common business scenarios.

After scheduling dependencies are configured for a node, the system checks whether the scheduling dependencies are consistent with the data lineage in the code developed for the node when you commit the node. For more information, see Operations performed by the system after scheduling dependencies are configured.

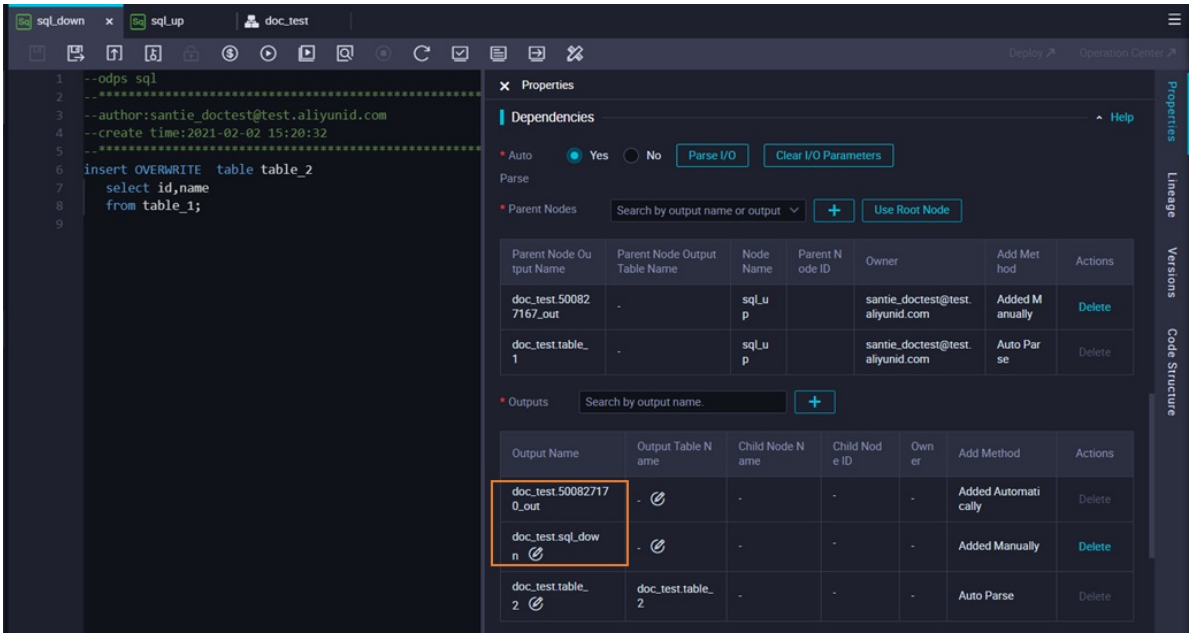
General principles for configuring scheduling dependencies

This section describes the general principles for configuring **scheduling dependencies** and **configuration items**:

- **Output**: the output of the current node. The node name must be unique within your Alibaba Cloud account.

  After you create nodes, the system automatically generates the following two **outputs** for each node:

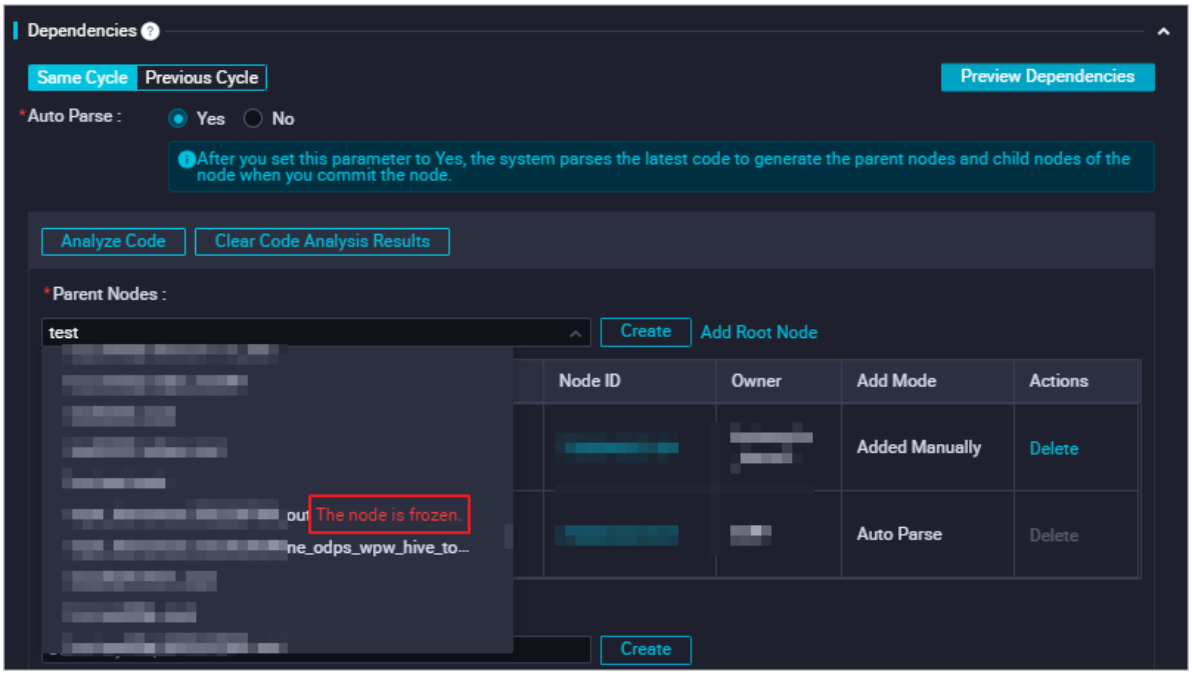
  - One **output** whose name is suffixed with ********_out**
  - The other **output** named in the `projectname.nodename` format



- **Parent Nodes**: the ancestor node on which the current node depends. After you configure this parameter, the system can find the ancestor node based on the configured output name or output table name of the ancestor node.

  If you enter an output name to search for the ancestor node, the system searches for the output name among the output names of the nodes that have been committed to the scheduling system.

  Fuzzy match is supported. After you enter a keyword, all nodes whose names contain the keyword are displayed. If the message **The node is frozen** is displayed on the right side of a node, do not use this node as an ancestor node of the current node. If you use a frozen node as an ancestor node of the current node, the current node may not run as expected.



Regardless of the method that is used to configure scheduling dependencies, the scheduling dependencies must comply with the following principles:

- A table is generated by only one node, and the table must be configured as the output of the node.

> **Note**
>
> - The system automatically adds the table generated by an SQL node to the output of the SQL node based on the automatic parsing feature.
> - You must manually add the table generated by a batch sync node to the output of the node. The name of the table is in the projectname.tablename format. This way, the output table of the node is configured as the input table of its descendant node based on the automatic parsing feature when the descendant node is run.

- The output of a node must be configured as the input of its descendant node, which forms dependencies between nodes.

For more information about the logic of scheduling dependencies, see Logic of same-cycle scheduling dependencies. The following sections describe the principles and configuration methods for scheduling dependencies in detail.

> **Note** If a workspace is created before January 10, 2019, some of the data generated in the workspace may be invalid. In this case, you must submit a ticket to resolve the issue. If a workspace is created after January 10, 2019, this issue does not occur.

Configuration based on the automatic parsing feature

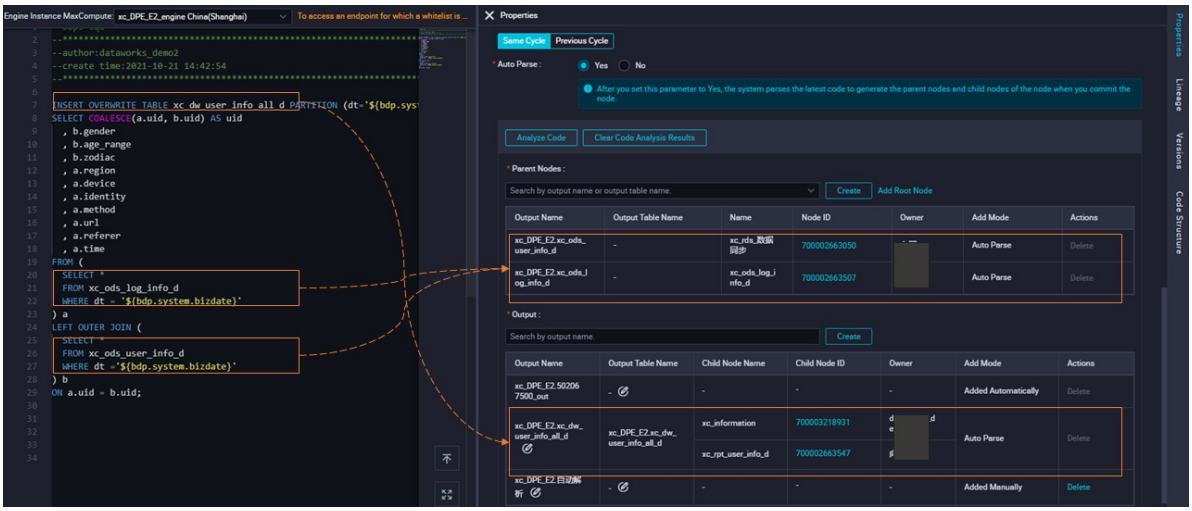
- Scenarios

  The system can automatically parse SQL statements in the code developed for a node to obtain the data lineage. Then, the system adds an **output** or an **ancestor node** for the node based on the data lineage and automatic parsing feature. The automatic parsing feature is convenient, efficient, and suitable for most scenarios.

  > **Note** The automatic parsing feature cannot be used to configure scheduling dependencies for batch synchronization nodes. After a table is generated in a synchronization node, you must manually add the table as the output of the node. The table name is in the project_name.table_name format. This way, the scheduling dependencies of the synchronization node can be quickly configured by using the automatic parsing feature when data of the generated table is cleansed by the descendant nodes of the synchronization node.

- How it works

The following figure shows the principles of automatic parsing for dependencies.



- If a table is specified in the SELECT statement of the code for a node, the system adds the table name to Parent Nodes for the node based on the automatic parsing feature.
- If a table is specified in the INSERT statement of the code for a node, the system adds the table name to Output for the node based on the automatic parsing feature.

If multiple INSERT and SELECT statements are used, multiple output and input names are parsed based on the automatic parsing feature.

- Configuration principles

The automatic parsing feature enables the system to automatically identify relationships between nodes and configure scheduling dependencies for nodes. The following table describes the principles of automatic parsing.

Node type	SQL statement	Configuration based on automatic parsing	Configuration principle
ODPS node	- CREATE - INSERT	If the code developed for the node contains such SQL statements, the system automatically adds an **output** for the node.	The **output** that is automatically added by the system is named in the *odps_project_name.table_name* format. In the preceding format: *odps_project_name*: the DataWorks workspace to which the node belongs. *table_name*: the name of the generated table.
	SELECT	If the code developed for the node contains such an SQL statement, the system automatically adds an **ancestor node** for the node.	The **ancestor node** that is automatically added by the system is named in the *project_name.table_name* format. In the preceding format: *project_name*: the name of the workspace to which the node that generates the table belongs. *table_name*: the name of the generated table.

<table>
<tr><th>Node type</th><th>SQL statement</th><th>Configuration based on automatic parsing</th><th>Configuration principle</th></tr>
<tr><td rowspan="2">SQL node other than the ODPS node</td><td>ALTER
CREATE
UPDATE
INSERT</td><td>If the code developed for the node contains such SQL statements, the system automatically adds an output for the node.</td><td>The outputs that are automatically added by the system for different nodes are named in the following formats:
E-MapReduce (EMR): workspace_name.db_name.table_name
AnalyticDB for PostgreSQL: workspace_name.db_name.schema_name.table_name
AnalyticDB for MySQL: workspace_name.db_name.schema_name.table_name
Hologres: workspace_name.db_name.schema_name.table_name
In the preceding formats:
workspace_name: the name of the DataWorks workspace to which the node belongs.
db_name: the name of the database to which the data is written.
schema_name: the name of the schema of the node.
table_name: the name of the generated table.</td></tr>
<tr><td>SELECT</td><td>If the code developed for the node contains such an SQL statement, the system automatically adds an ancestor node for the node.</td><td>The ancestor node that is automatically added by the system is named in the project_name.table_name format.
In the preceding format:
project_name: the name of the workspace to which the node that generates the table belongs.
table_name: the name of the generated table.</td></tr>
<tr><td>Batch synchronization node</td><td colspan="3">Batch synchronization nodes do not support the automatic parsing feature. You must manually configure scheduling dependencies for such nodes.
Note After a table is generated in a synchronization node, you must manually add the table as the output of the node. The table name is in the project_name.table_name format. This way, the scheduling dependencies of the synchronization node can be quickly configured by using the automatic parsing feature when data of the generated table is cleansed by the descendant nodes of the synchronization node.</td></tr>
</table>

- Precautions

- Requirements for node development

  Automatic parsing enables the system to automatically identify the scheduling dependencies of a node based on the code that you developed for the node. Therefore, we recommend that you strictly comply with the following requirements when you develop data:

  - Requirements for code development: One node generates only one table, and a table is generated by only one node.
  - Requirements for node creation: The name of a node must be consistent with that of the table that is to be generated by the node.
  - Requirements for scheduling configurations: The table generated by a node must be added to Output for the node.

- Nodes that do not support the automatic parsing feature
  - Batch synchronization nodes, AnalyticDB for PostgreSQL nodes, AnalyticDB for MySQL nodes, and EMR nodes do not support configuration of scheduling dependencies based on the automatic parsing feature. The tables generated by these nodes must be manually added to the outputs of these nodes.
  - Temporary tables created by executing SQL statements do not support the automatic parsing feature. For example, in the Workspace settings topic, the tables whose names are prefixed with t_ are specified as temporary tables. Such tables cannot be automatically added to Output or Parent Nodes for nodes.

- Logic for handling non-standard configurations
  - If a table specified in SQL statements is used as both an output table and a referenced table, the table is parsed only as an output table.
  - If a table specified in SQL statements is used as an output table or a referenced table multiple times, only the latest scheduling dependency that is recommended by the system is used.

- Scheduling dependency inconsistencies

  After you enable automatic parsing for a node, the system automatically generates the input and output for the node based on the data lineage in the code of the node when you commit the node. This ensures that the node can successfully generate data. You can also modify the input and output of the node based on your business requirements.

  When you commit a node, if the scheduling dependencies of the node that are modified based on the automatic parsing result are different from those in the development or production environment, a message indicating the changes of the scheduling dependencies is displayed. The changes indicate that some inputs or outputs are added or deleted after you modify the scheduling dependencies in the **Dependencies** section of the **Properties** tab based on the automatic parsing result. You can choose whether to use the **current scheduling dependencies** and commit the current node that is scheduled based on the current scheduling dependencies. After the node is committed, the latest inputs and outputs are automatically added to Parent Nodes and Output in the **Dependencies** section of the **Properties** tab.

  > **Note** When you commit a node, if the system detects that the current scheduling dependencies of the node are different from those in the production or development environment, check whether the current scheduling dependencies of the node meet your business requirements. This can prevent inappropriate changes that you make on the scheduling dependencies from affecting data generation. If the current node has a large number of descendant nodes, inappropriate scheduling dependencies may have a great impact on the descendant nodes and data generation. Do not change the scheduling dependencies unless necessary.
  >
  > For example, if the system detects that the scheduling dependencies of a node do not include the input A that is included in the scheduling dependencies in the development or production environment when you commit the node, you must check whether the current scheduling dependencies are correct. A is the **output name of an ancestor node** of the current node. If Table A is specified in the SELECT statement in the code developed for the current node but the node that generates data of Table A is not configured as an ancestor node for the current node, the current node may start to run before the data of Table A is generated. In this case, the current node cannot be run and generate data.

- Other precautions

  If an SQL statement in the code of a node specifies a table that is not generated by an auto triggered node, the system adds the table to **Parent Nodes** for the current node based on the automatic parsing feature. The tables that are not generated by auto triggered nodes include dimension tables and tables uploaded from on-premises machines to DataWorks. However, the system cannot find the node that generates the table added to **Parent Nodes**, and an error occurs. In this case, you must manually delete the table.

Manual configuration

- Scenarios

  DataWorks allows you to manually modify the **Parent Nodes** and **Output** parameters for a node during the code development for the node. If the scheduling dependencies automatically generated by the system for your node do not meet your business requirements, you can manually modify the dependencies.

  You can manually configure scheduling dependencies in the following scenarios:

- You must delete a scheduling dependency configured by using a table that is not generated by an auto triggered node.

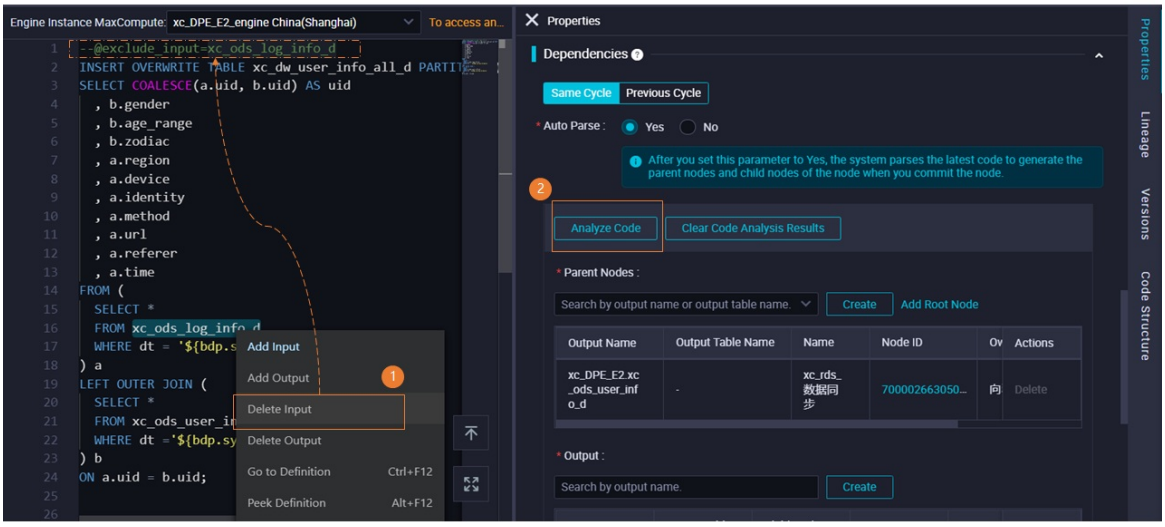
  Scheduling dependencies ensure that a node can successfully obtain the table data generated by its ancestor node that is scheduled to run. However, if the ancestor node of a node is not scheduled to run, the system cannot monitor whether the ancestor node has generated the latest table data. If the table specified in the SELECT statement of the code for a node is not generated by an auto triggered node and the table name is automatically added to Parent Nodes for the node, you must manually delete the dependency for the node. Tables that are not generated by auto triggered nodes include the following types:

  - Tables uploaded from on-premises machines to DataWorks
  - Dimension tables
  - Tables that are not generated by nodes scheduled by DataWorks
  - Tables generated by manually triggered nodes

- You must manually add the tables generated by nodes that do not support the automatic parsing feature to the outputs of these nodes.

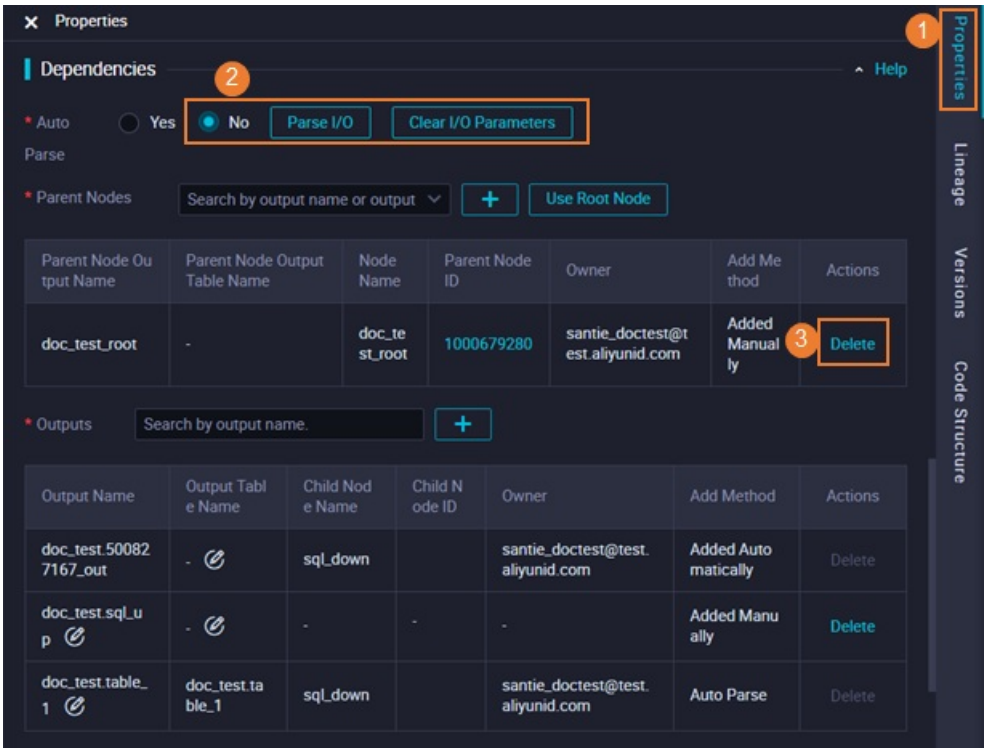
  Batch synchronization nodes, AnalyticDB for PostgreSQL nodes, AnalyticDB for MySQL nodes, and EMR nodes do not support configuration of scheduling dependencies based on the automatic parsing feature. The tables generated by these nodes must be manually added to the outputs of these nodes.

- Configuration principles
  - Delete a scheduling dependency in the code editor of a node



  If the SELECT statement in the code of a node specifies a table that is not generated by an auto triggered node, you can delete the dependency for the node. Specifically, you can go to the code editor of the node, right-click the name of the table that you want to remove from the input, and then click Delete Input. The preceding figure shows the process. You can also add a rule as a comment at the top of the code. This way, the system does not automatically parse the dependency based on the rule.

- Delete a scheduling dependency in the Properties panel of a node



If the SELECT statement in the code of a node specifies a table that is not generated by an auto triggered node, you can manually remove the table from **Parent Nodes** for the node. Specifically, you can go to the Properties panel of the node, set Auto Parse to No, and then manually perform the operation.

- Precautions
  - When you configure scheduling dependencies on the Properties tab of a node, we recommend that you click **Analyze Code** after you set Auto Parse to **No**. This way, the system automatically adds scheduling dependencies for the node by using a method that is consistent with automatic parsing. Then, you can perform operations on the scheduling dependencies based on your business requirements.
  - You can click **Clear Code Analysis Results** to delete the scheduling dependencies that are automatically added by the system. The scheduling dependencies that are manually added are not deleted.
  - If you click **Recommendation**, the system recommends all the other SQL nodes that generate the input table of the current node based on the SQL lineage of the current workspace. You can select your desired node from the recommended results and configure the node as the ancestor node on which the current node depends.

    > **Note** Only the nodes that are committed and deployed to the production environment and have generated data in the required table can be recommended by the system. Therefore, the recommended nodes may be parsed one day later than the scheduling time of the current code.
    >
    > The recommended nodes must be committed to the scheduling system on the previous day. This way, they can be identified by the automatic recommendation feature after data is generated on the current day.

Configuration by drawing lines to connect nodes

- Scenarios

  DataWorks allows you to specify the relationships between nodes by drawing lines to connect nodes on the editing page of a workflow. After the nodes are connected, the system automatically generates scheduling dependencies for each node based on the connections.

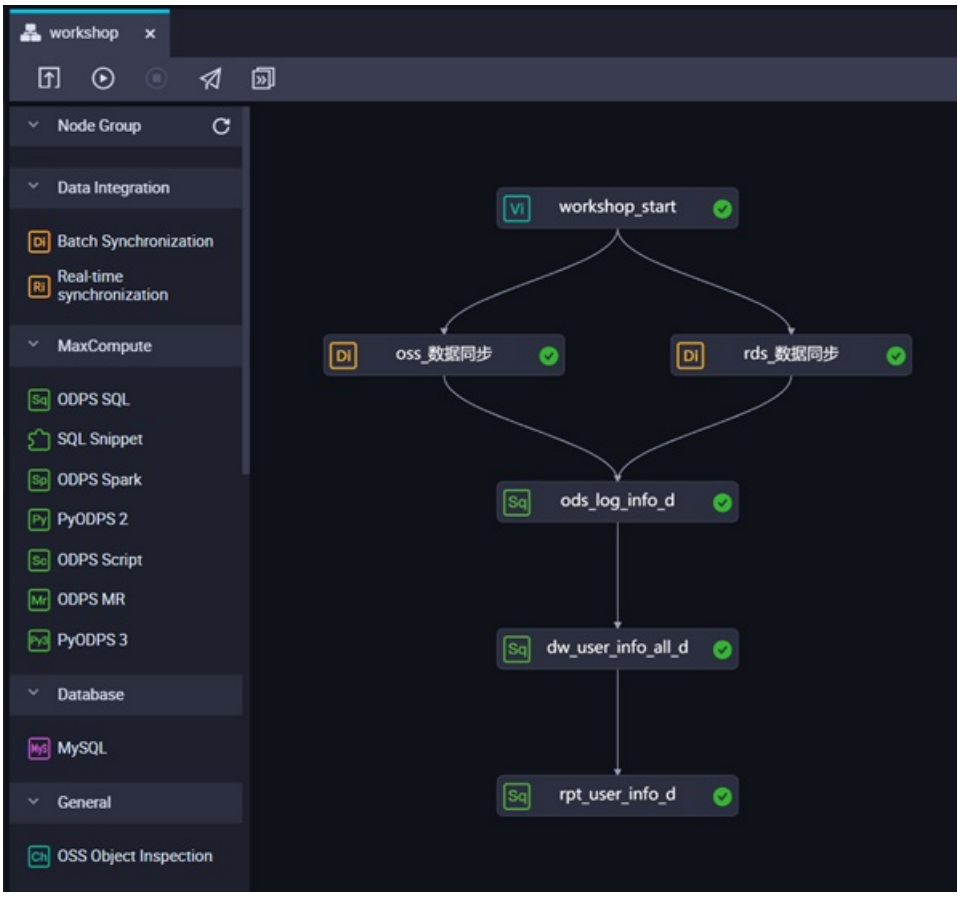
  After you create a workflow, you can connect nodes by drawing lines on the configuration tab of the workflow to configure scheduling dependencies for each node based on your business requirements. During subsequent code development, you can add or modify scheduling dependencies for each node manually or by using the automatic parsing feature. This way, all nodes in the workflow can be configured with correct scheduling dependencies.

- How it works

  When you connect nodes by drawing lines, the system automatically adds an output whose name is suffixed with ********_out** to the input of each descendant node.

- Configuration principles

  You can connect nodes by drawing lines on the editing page of a workflow.



Preview scheduling dependencies

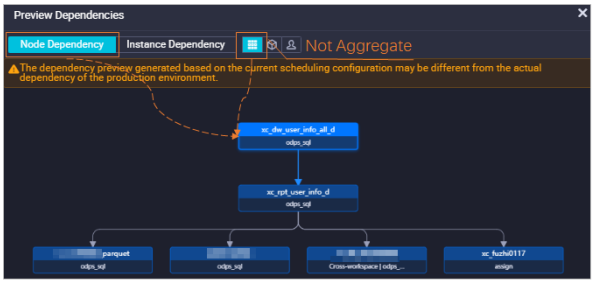
After you configure scheduling dependencies for a node, click **Preview Dependencies**. In the Preview Dependencies dialog box, you can preview the scheduling dependencies of the node on the **Node Dependency** and **Instance Dependency** tabs. You can modify the scheduling dependencies that do not meet your business requirements.

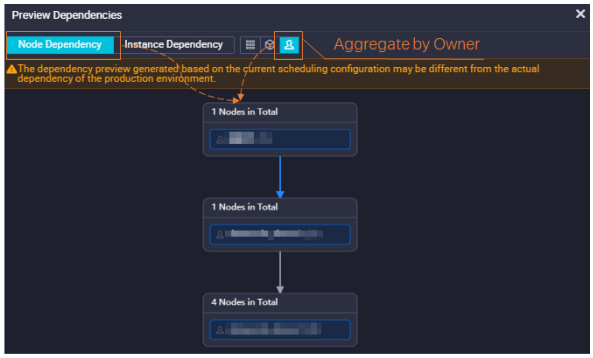
> **Note**
>
> - A directed acyclic graph (DAG) that is generated based on the scheduling dependencies is only for reference. The DAG that is generated may be different from the DAG in the production environment.
> - Only the following roles can be used to preview the scheduling dependencies of a node: **Development**, **O&M**, **Project Owner**, and **Workspace Manager**. If a user wants to preview the scheduling dependencies of a node, you must assign one of the preceding roles to the user. For more information, see Manage workspace-level roles and members.
> - You can preview only the ancestor nodes and descendant nodes that are at the nearest level of the current node.
> - If you do not save scheduling dependencies of a node before you click **Preview Dependencies**, click **Confirm** in the **Attention** dialog box. This way, you can view the latest scheduling dependencies of the node.
> - On the **Instance Dependency** tab, you can preview scheduling dependencies of an auto triggered node that generates multiple instances. For example, you can preview the scheduling dependencies of *an auto triggered node scheduled by hour*. The auto triggered node scheduled by hour depends on an auto triggered node scheduled by minute.
> - The scheduling dependencies of a node that you preview are as expected only after you save the ancestor nodes of the node.

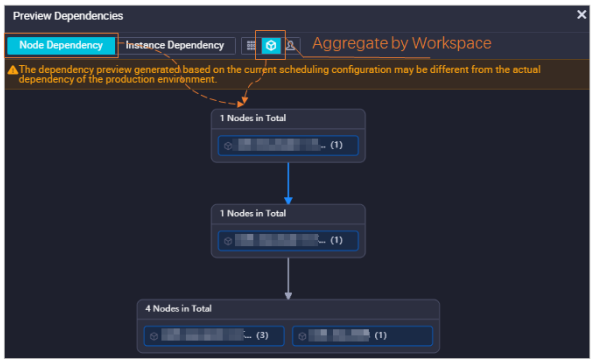
- Select a preview method.

  You can select **Not Aggregate**, **Aggregate by Workspace**, or **Aggregate by Owner** to preview the scheduling dependencies of a node. For more information about aggregation methods, see Manage instances in a DAG.

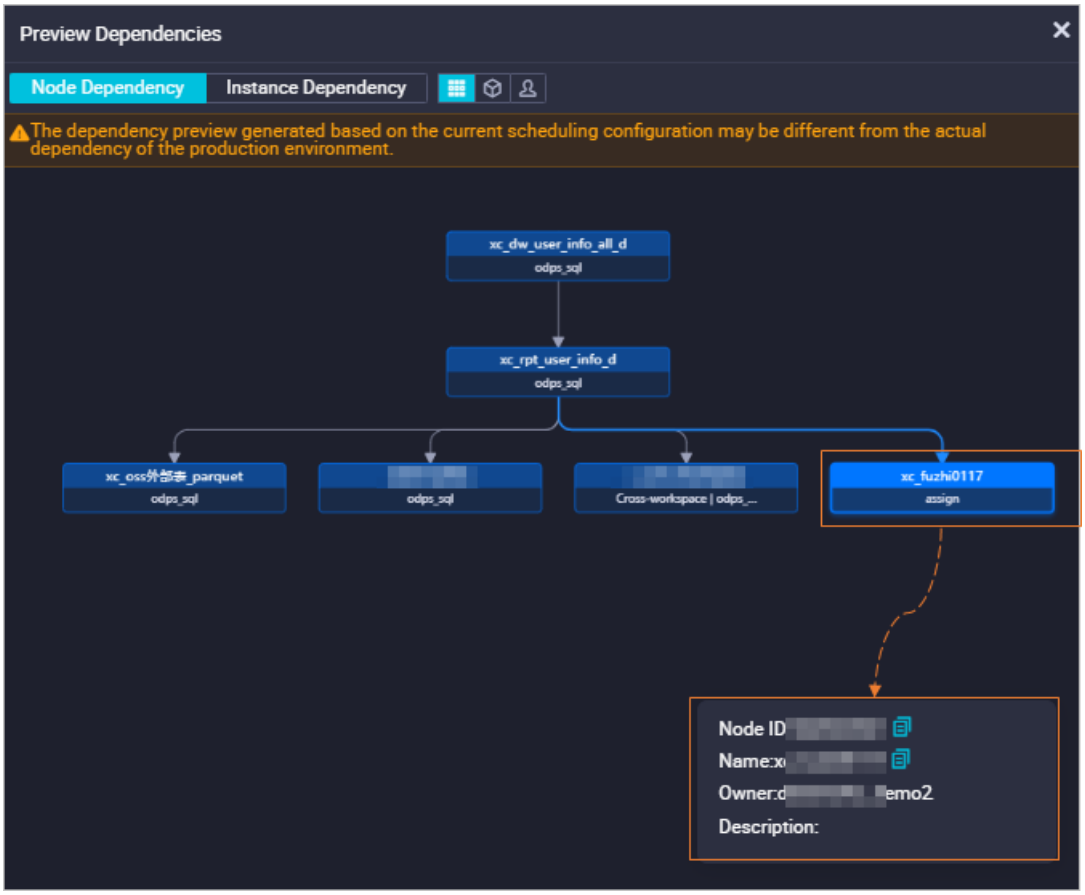
The following figures show preview effects by using different aggregation methods on the **Node Dependency** tab.





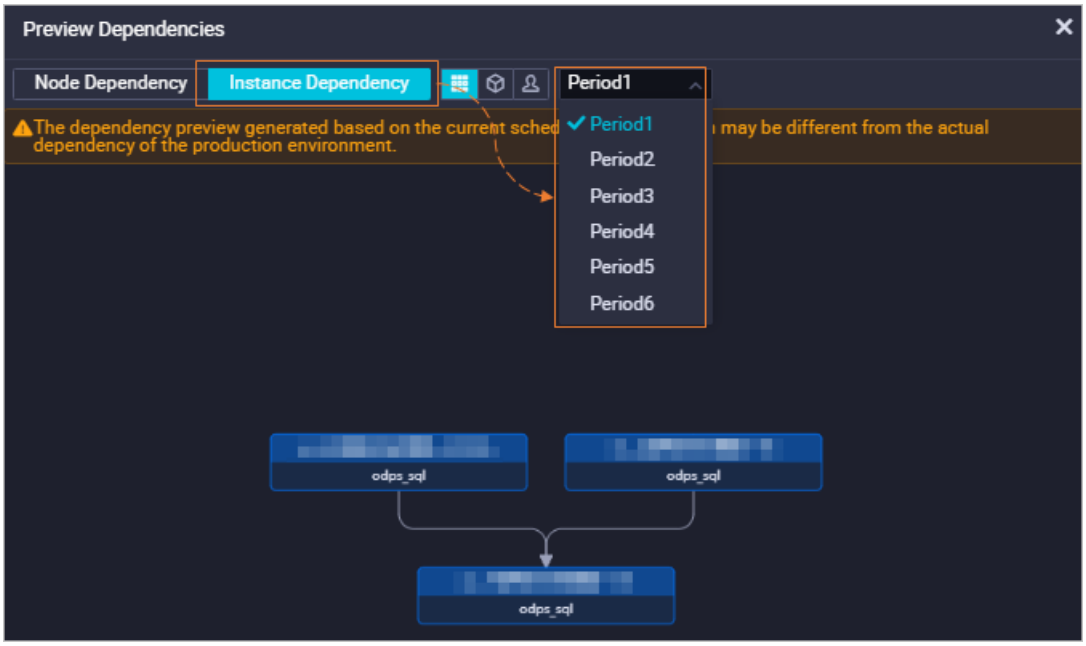


You can click a node to view the basic information about the node.



- Preview scheduling dependencies of an auto triggered node that generates multiple instances.

  For an auto triggered node that generates multiple instances, you can click the **Instance Dependency** tab and select a scheduling cycle to preview scheduling dependencies of the node.



Instructions to configure scheduling dependencies in common business scenarios

- Scenario 1: Configure scheduling dependencies for batch synchronization nodes in a workflow
- Scenario 2: Configure scheduling dependencies for a node that depends on last-cycle instances
- Scenario 3: Configure scheduling dependencies for nodes across workflows or workspaces

Operations performed by the system after scheduling dependencies are configured

After scheduling dependencies are configured for a node, the system checks whether the configured scheduling dependencies are consistent with the data lineage in the code developed for the node. If they are inconsistent, the system displays an error message. In this case, you must modify the scheduling dependencies based on actual situations. For more information, see When I commit a node, the system reports an error that the input and output of the node are not consistent with the data lineage in the code developed for the node. What do I do?.

FAQ

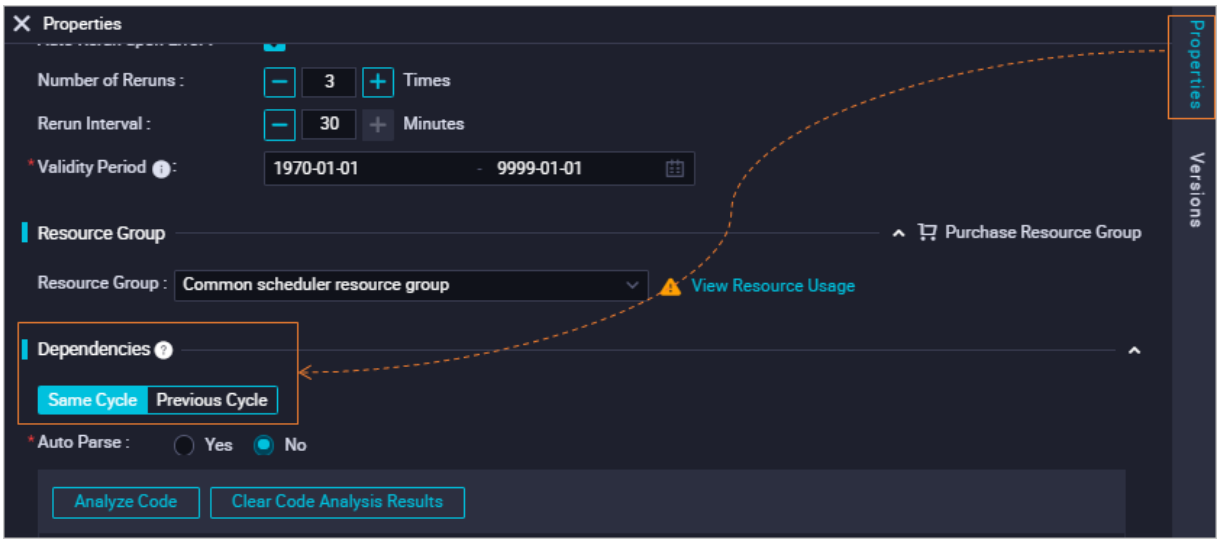
- When I commit Node A, the system reports an error that the output name of the dependent ancestor node of Node A does not exist. What do I do?
- When I commit a node, the system reports an error that the input and output of the node are not consistent with the data lineage in the code developed for the node. What do I do?
- Scheduling dependencies

8.5.3. Configure previous-cycle scheduling dependencies

DataWorks allows you to configure previous-cycle scheduling dependencies for auto triggered nodes. You can configure the instance generated for a node in the current cycle to depend on the instances generated for one or more specific nodes in the previous cycle. This way, the instance generated for the node in the current cycle can start to run only after the instances generated for one or more specific nodes on which the node depends are successfully run. This topic describes how to configure previous-cycle scheduling dependencies for a node and the types of previous-cycle dependencies.

Configuration scheduling dependencies

Create a node in DataStudio and go to the configuration tab of the node. Click the **Properties** tab in the right-side navigation pane. Then, configure scheduling dependencies for the node in the **Dependencies** section of the Properties tab. DataWorks allows you to configure same-cycle or previous-cycle scheduling dependencies for a node.



The following table describes the differences between the two types of scheduling dependencies.

Dependency type	Business logic and use scenario	Display of dependencies
Same-cycle scheduling dependency	The current node (Node A) needs to use the table data that is generated by another node on the current day. The table is specified in the `SELECT` statement for Node A. To ensure that Node A can successfully obtain the table data, you must configure **same-cycle scheduling dependencies** for Node A.	Same-cycle scheduling dependencies are presented as solid lines in Operation Center. For more information about how to go to Operation Center and view the scheduling dependencies of a node, see Overview.

Dependency type	Business logic and use scenario	Display of dependencies
Previous-cycle scheduling dependency	The current node (Node A) needs to use the table data that is generated by one or more specific nodes in the previous cycle. The table is specified in the `SELECT` statement for Node A. To ensure that Node A can successfully obtain the table data, you must configure **previous-cycle scheduling dependencies** for Node A.	Previous-cycle scheduling dependencies are presented as dashed lines in Operation Center. For more information about how to go to Operation Center and view the scheduling dependencies of a node, see Overview.

Types of previous-cycle scheduling dependencies

The following table describes the types of previous-cycle scheduling dependencies.

Type	Description	Scenario
Dependency on the instance generated for the current node in the previous cycle	The instance generated for a node in the current cycle starts to run only after the instance generated for the same node in the previous cycle is successfully run.	The instance generated for a node in the current cycle depends on the latest business data from the instance generated for the same node in the previous cycle.
Dependency on the instances generated for the descendant nodes of a node in the previous cycle	The instance generated for a node in the current cycle starts to run only after the instances generated for the descendant nodes of the current node in the previous cycle are successfully run. For example, Node A has three descendant nodes: Node B, Node C, and Node D. If you configure this type of scheduling dependency for Node A, the instance generated for Node A in the current cycle depends on the instances generated for Node B, Node C, and Node D in the previous cycle. The instance generated for Node A in the current cycle starts to run only after the instances generated for Node B, Node C, and Node D in the previous cycle are successfully run.	The instance generated for a node in the current cycle depends on whether the output table data of the current node in the previous cycle is cleansed by the instances generated for the descendant nodes of the current node in the previous cycle.

Type	Description	Scenario
Dependency on the instances generated for one or more specified nodes in the previous cycle	The instance generated for a node in the current cycle starts to run only after the instances generated for one or more specified nodes in the previous cycle are successfully run. **Note** If you configure this type of scheduling dependency for a node, you must search by node ID to add the nodes on which the node needs to depend in the Dependencies section of the Properties tab.	The instance generated for a node in the current cycle depends on the output table data from the instances generated for one or more other nodes in the previous cycle in the business logic but does not use the data in the code.

You can set **Follow the upstream air running attribute** to Yes when you set Depend On to **Instances of Current Node** or **Other Nodes**. For more information, see Pass the dry-run attribute of an ancestor node.

After you configure scheduling dependencies for your node, you can preview the scheduling dependencies. For more information, see Configure same-cycle scheduling dependencies.

For more information about typical scenarios of the dependency on instances in the previous cycle, see Examples.

Dependency on the instance generated for the current node in the previous cycle

- **Node dependency**

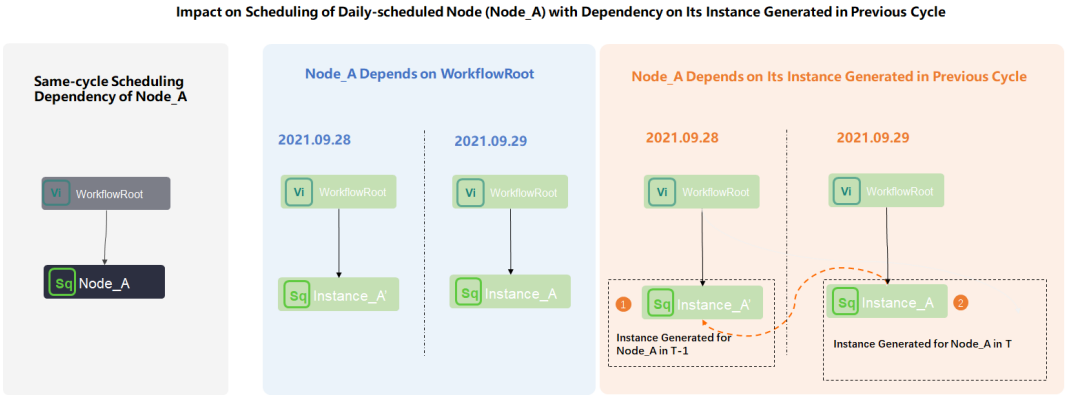
  The instance generated for a node in the current cycle starts to run only after the instance generated for the same node in the previous cycle is successfully run.

- **Scenario**

  The instance generated for a node in the current cycle depends on the latest business data from the instance generated for the same node in the previous cycle.

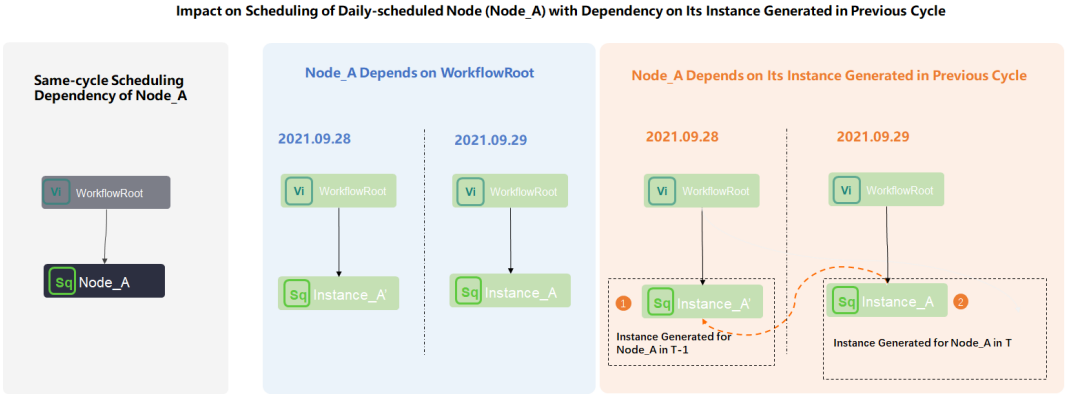
- **Impact on the scheduling of the current node whose instance generated in the current cycle is configured to depend on the instance generated for the same node in the previous cycle**

- Example 1: Configure the instance generated in the current cycle for a node scheduled by day to depend on the instance generated for the same node in the previous cycle
  - WorkflowRoot and Node_A are auto triggered nodes that are scheduled by day.
  - The following configuration is performed for Node_A: The instance generated in the current cycle depends on the instance generated in the previous cycle.
  - Node_A generates an instance named `Instance_A` in the current cycle (T).
  - Node_A generates an instance named `Instance_A'` in the previous cycle (T-1).



After you set **Depend On** to Instances of Current Node for Node_A in the Dependencies section of the Properties tab, `Instance_A` starts to run in the current cycle only after `Instance_A'` and the instance generated for `WorkflowRoot` in the current cycle are successfully run.

- Example 2: Configure the instance generated in the current cycle for a node scheduled by hour to depend on the instance generated for the same node in the previous cycle
  - WorkflowRoot is a zero load node that is scheduled by day. Node_A is an auto triggered node that is scheduled by hour. WorkflowRoot is the ancestor node of the Node_A.
  - Node_A is scheduled to run every 8 hours from `00:00` to `23:59`. The scheduled time of Node_A is `00:00 (T1)`, `08:00 (T2)`, and `16:00 (T3)`.
  - Node_A generates the following instances at T1, T2, and T3: `Instance_A1`, `Instance_A2`, and `Instance_A3`.
  - The following configuration is performed for Node_A: The instance generated in the current cycle depends on the instance generated in the previous cycle.

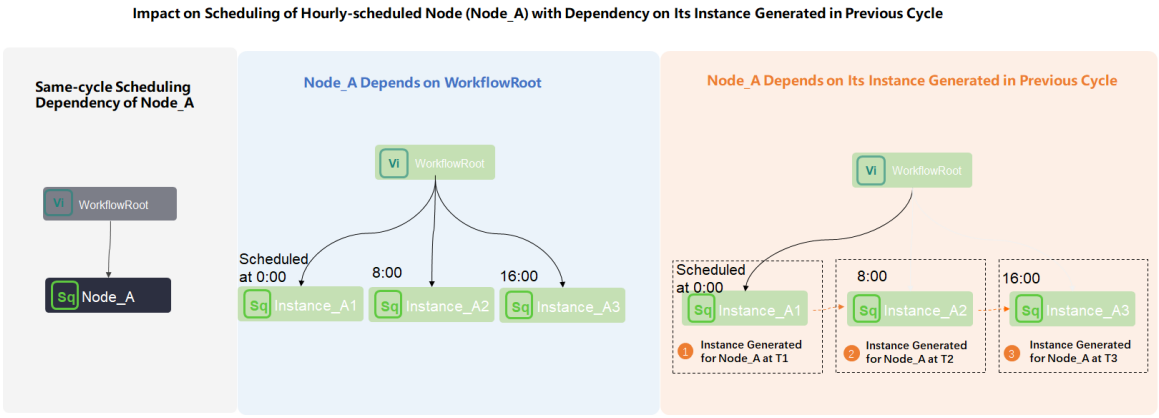


  - If you do not configure the instance generated for Node_A in the current cycle to depend on the instance generated for Node_A in the previous cycle, after WorkflowRoot is successfully run on the current day, `Instance_A1`, `Instance_A2`, and `Instance_A3` run based on their own scheduled time.
  - If you configure the instance generated for Node_A in the current cycle to depend on the instance generated for Node_A in the previous cycle, `Instance_A1` depends on the instance generated for WorkflowRoot, `Instance_A2` depends on `Instance_A1`, and `Instance_A3` depends on `Instance_A2`. In this case, an instance can run only after the instance on which it depends in the previous cycle is successfully run.

> **Note** This example uses an auto triggered node scheduled by hour to demonstrate the logic of the dependency on the instance generated for the same node in the previous cycle. The logic is similar for an auto triggered node scheduled by minute.

- **Impact on the scheduling of the descendant nodes when the instance generated in the current cycle for the current node scheduled by hour or minute is configured to depend on the instance generated for the same node in the previous cycle**
  - WorkflowRoot is a zero load node scheduled by day. Node_A is an auto triggered node scheduled by hour. WorkflowRoot is the ancestor node of Node_A. Node_B and Node_C are auto triggered nodes scheduled by day and are descendant nodes of Node_A.
  - Node_A is scheduled to run every 8 hours from `00:00` to `23:59`. The scheduled time of Node_A is `00:00 (T1)`, `08:00 (T2)`, and `16:00 (T3)`.
  - Node_B is scheduled to run at `00:00` every day. Node_C is scheduled to run at `08:00` every day.

- Node_A generates the following instances at T1, T2, and T3: `Instance_A1`, `Instance_A2`, and `Instance_A3`.
- Node_B and Node_C generate instances `Instance_B` and `Instance_C`.
- The following configuration is performed for Node_A: The instance generated in the current cycle depends on the instance generated in the previous cycle.



- If you do not configure the instance generated for Node_A in the current cycle to depend on the instance generated for Node_A in the previous cycle, the instances generated for Node_A, Node_B, and Node_C run in the following ways:
  - After WorkflowRoot is successfully run on the current day, `Instance_A1`, `Instance_A2`, and `Instance_A3` run based on their own scheduled time.
  - `Instance_B` and `Instance_C` depend on all the three instances generated for Node_A on the current day. This indicates that `Instance_B` and `Instance_C` start to run only after `Instance_A1`, `Instance_A2`, and `Instance_A3` are all successfully run on the current day.
- If you configure the instance generated for Node_A in the current cycle to depend on the instance generated for Node_A in the previous cycle, the instances generated for Node_A, Node_B, and Node_C run in the following ways:
  - `Instance_A1` depends on the instance generated for WorkflowRoot, `Instance_A2` depends on `Instance_T1`, and `Instance_A3` depends on `Instance_A2`. In this case, an instance can run only after the instance on which it depends in the previous cycle is successfully run.
  - `Instance_B` and `Instance_C` depend on the instances whose scheduled time is closest to their scheduled time.

    This indicates that `Instance_B` whose scheduled time is `00:00` starts to run after `Instance_A1` is successfully run. `Instance_C` does not run.

    `Instance_C` whose scheduled time is `08:00` starts to run after `Instance_A2` is successfully run.

Dependency on the instances generated for the descendant nodes of a node in the previous cycle

- **Node dependency**

  The instance generated for a node in the current cycle starts to run only after the instances generated for the descendant nodes of the current node in the previous cycle are successfully run.

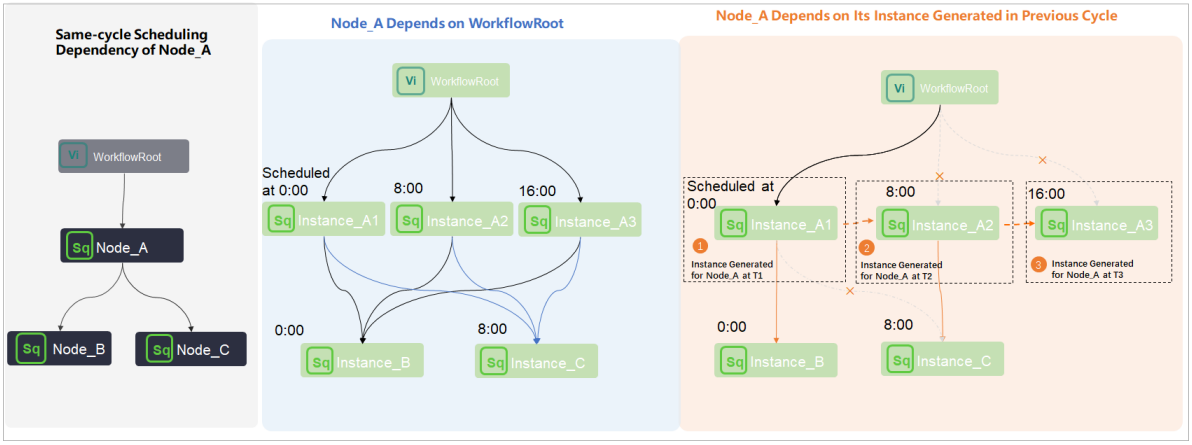
For example, Node A has three descendant nodes: Node B, Node C, and Node D. If you configure this type of scheduling dependency for Node A, the instance generated for Node A in the current cycle depends on the instances generated for Node B, Node C, and Node D in the previous cycle. The instance generated for Node A in the current cycle starts to run only after the instances generated for Node B, Node C, and Node D in the previous cycle are successfully run.

- Scenario

  The instance generated for a node in the current cycle depends on whether the output table data of the current node in the previous cycle is cleansed by the instances generated for the descendant nodes of the current node in the previous cycle.

- Example
  - WorkflowRoot, Node_A, Node_B, and Node_C are auto triggered nodes scheduled by day.
  - Node_B and Node_C are the descendant nodes of Node_A.
  - Node_A, Node_B, and Node_C generate the following instances in the current cycle (T): `Instance_A`, `Instance_B`, and `Instance_C`.
  - Node_A, Node_B, and Node_C generate the following instances in the previous cycle (T-1): `Instance_A'`, `Instance_B'`, and `Instance_C'`.

Impact on Scheduling of Descendant Nodes of Hourly-scheduled Node (Node_A) with Dependency on Its Instance Generated in Previous Cycle



After you set **Depend On** to Level-1 Child Node for Node_A, `Instance_A` depends on `Instance_B'`, `Instance_C'`, and the instance generated for `WorkflowRoot`. In this case, `Instance_A` starts to run only after Instance_B', Instance_C', and the instance generated for WorkflowRoot are successfully run.

Dependency on the instances generated for one or more specified nodes in the previous cycle

- Node dependency

  The instance generated for a node in the current cycle starts to run only after the instances generated for one or more specified nodes in the previous cycle are successfully run.

  > **Note** If you configure this type of scheduling dependency for a node, you must search by node ID to add the nodes on which the node needs to depend in the Dependencies section of the Properties tab.

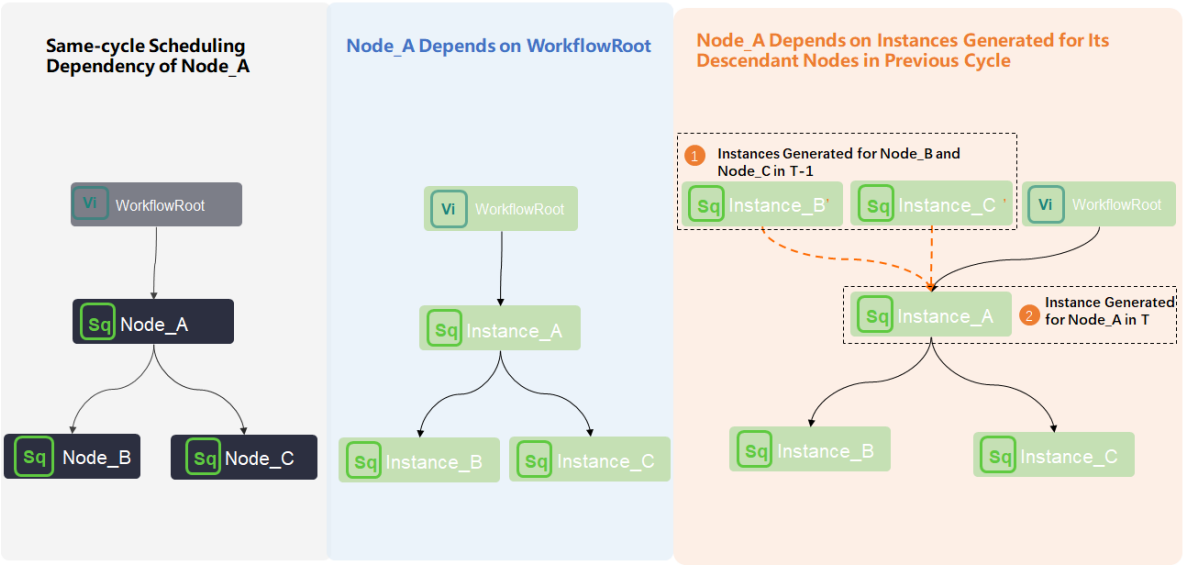
- Scenario

The instance generated for a node in the current cycle depends on the output table data from the instances generated for one or more other nodes in the previous cycle in the business logic but does not use the data in the code.

- Example
  - WorkflowRoot_1, WorkflowRoot_2, Node_A, Node_B, and Node_C are auto triggered nodes scheduled by day.
  - Node_A, Node_B, and Node_C belong to different workflows.

    Node_A and Node_B are the descendant nodes of WorkflowRoot_1. Node_C is the descendant node of WorkflowRoot_2.
  - The following configuration is performed for Node_A: The instance generated for Node_A in the current cycle depends on the instance generated for Node_C in the previous cycle.
  - Node_A, Node_B, and Node_C generate the following instances in the current cycle (T): `Instance_A`, `Instance_B`, and `Instance_C`.
  - Node_A, Node_B, and Node_C generate the following instances in the previous cycle (T-1): `Instance_A'`, `Instance_B'`, and `Instance_C'`.

Dependency on Instances of Descendant Nodes at Nearest Level of Daily-scheduled Node (Node_A)



After you set **Depend On** to Other Nodes for Node_A, `Instance_A` depends on `Instance_C'` and the instance generated for `WorkflowRoot_1`. In this case, `Instance_A` starts to run only after Instance_C' and the instance generated for WorkflowRoot_1 are successfully run.

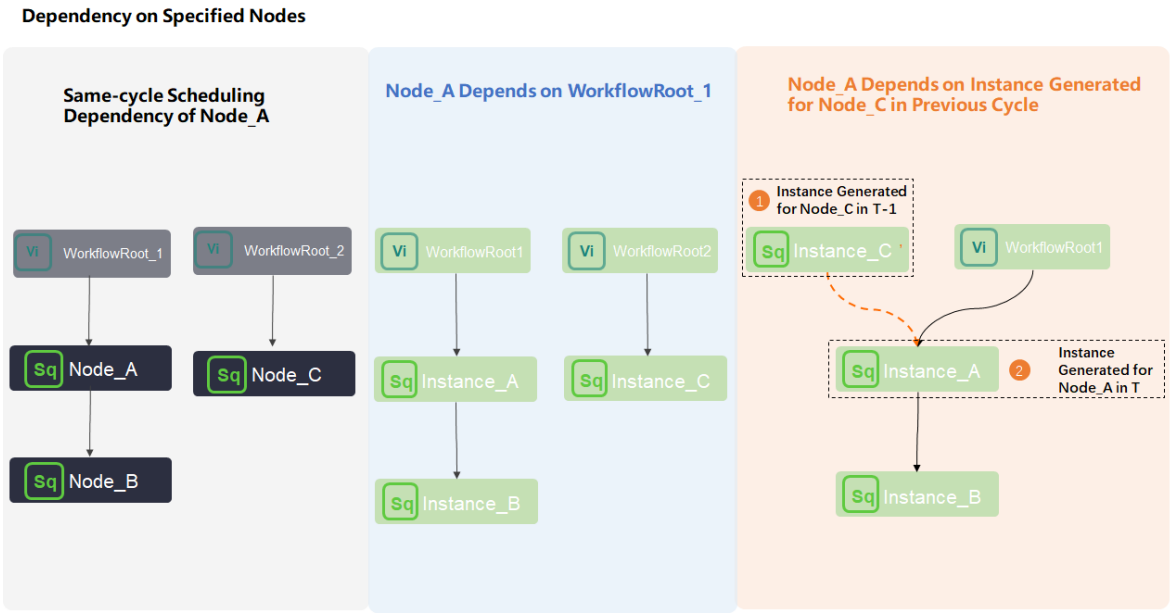
Pass the dry-run attribute of an ancestor node

- Scenario

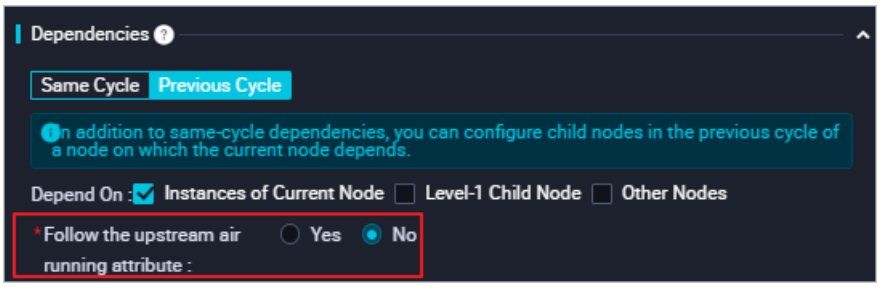
  If a branch node has two descendant nodes, one descendant node can normally run, and the other is dry-run. If you configure the instance generated for the dry-run descendant node in the current cycle to depend on the instance generated for the dry-run descendant node in the previous cycle, the dry-run attribute of the node is passed to the descendant nodes of the dry-run node. In this case, all the instances generated for the dry-run node and the descendant nodes of the dry-run node are dry-run. If you do not want the dry-run attribute to be passed, you can set **Follow the upstream air running attribute** to **No** for the dry-run descendant node in the Dependencies section of the Properties tab.

> **Note** The dry-run attribute of nodes other than branch nodes is not passed. When you use a **branch node** that has two or more descendant nodes, the dry-run attribute of one of the descendant nodes can be passed if you configure the instance generated for the descendant node in the current cycle to depend on the instance generated for the descendant node in the previous cycle.

- Example
  - Assign_Node is an assignment node. Branch_Node is a branch node. Shell_Node1 and Shell_Node2 are the descendant nodes of Branch_Node. All these nodes are scheduled by day.
  - Shell_Node1 is dry-run, and Shell_Node2 normally runs.
  - The following configuration is performed for Shell_Node1: The instance generated in the current cycle depends on the instance generated in the previous cycle.
  - Shell_Node1 generates an auto triggered node instance `Shell_Node1'` in the current cycle (T).
  - Shell_Node1 generates an auto triggered node instance `Shell_Node1` in the previous cycle (T-1).



`Shell_Node1'` depends on `Shell_Node1`. The dry-run attribute of Shell_Node1 is passed. Therefore, all the instances generated for Shell_Node1 and descendant nodes of Shell_Node1 are dry-run. You can set **Follow the upstream air running attribute** to **No** for Shell_Node1. This way, all the instances generated for Shell_Node1 and descendant nodes of Shell_Node1 can normally run.



Preview scheduling dependencies

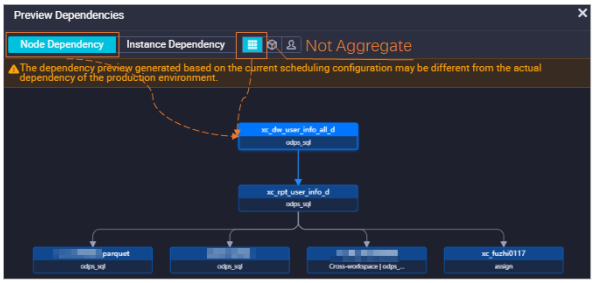
After you configure scheduling dependencies for a node, click **Preview Dependencies**. In the Preview Dependencies dialog box, you can preview the scheduling dependencies of the node on the **Node Dependency** and **Instance Dependency** tabs. You can modify the scheduling dependencies that do not meet your business requirements.

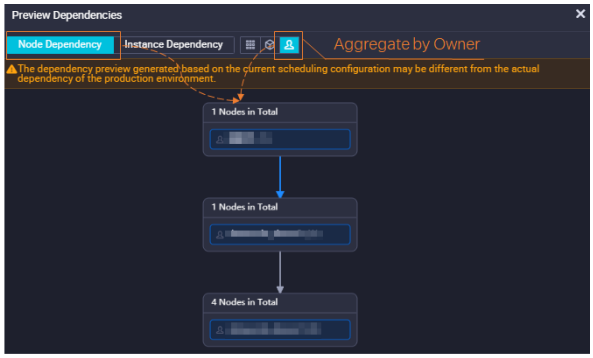
> **Note**
>
> - A directed acyclic graph (DAG) that is generated based on the scheduling dependencies is only for reference. The DAG that is generated may be different from the DAG in the production environment.
> - Only the following roles can be used to preview the scheduling dependencies of a node: **Development**, **O&M**, **Project Owner**, and **Workspace Manager**. If a user wants to preview the scheduling dependencies of a node, you must assign one of the preceding roles to the user. For more information, see Manage workspace-level roles and members.
> - You can preview only the ancestor nodes and descendant nodes that are at the nearest level of the current node.
> - If you do not save scheduling dependencies of a node before you click **Preview Dependencies**, click **Confirm** in the **Attention** dialog box. This way, you can view the latest scheduling dependencies of the node.
> - On the **Instance Dependency** tab, you can preview scheduling dependencies of an auto triggered node that generates multiple instances. For example, you can preview the scheduling dependencies of *an auto triggered node scheduled by hour*. The auto triggered node scheduled by hour depends on an auto triggered node scheduled by minute.
> - The scheduling dependencies of a node that you preview are as expected only after you save the ancestor nodes of the node.

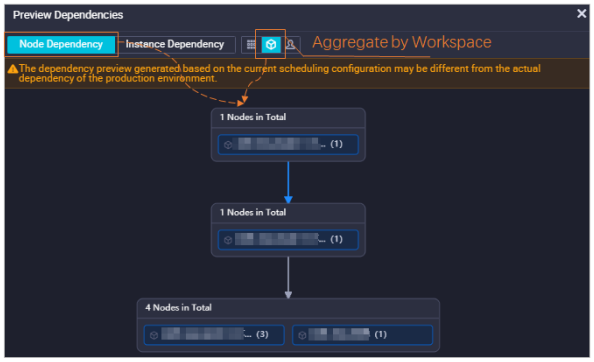
- Select a preview method.

  You can select **Not Aggregate**, **Aggregate by Workspace**, or **Aggregate by Owner** to preview the scheduling dependencies of a node. For more information about aggregation methods, see Manage instances in a DAG.

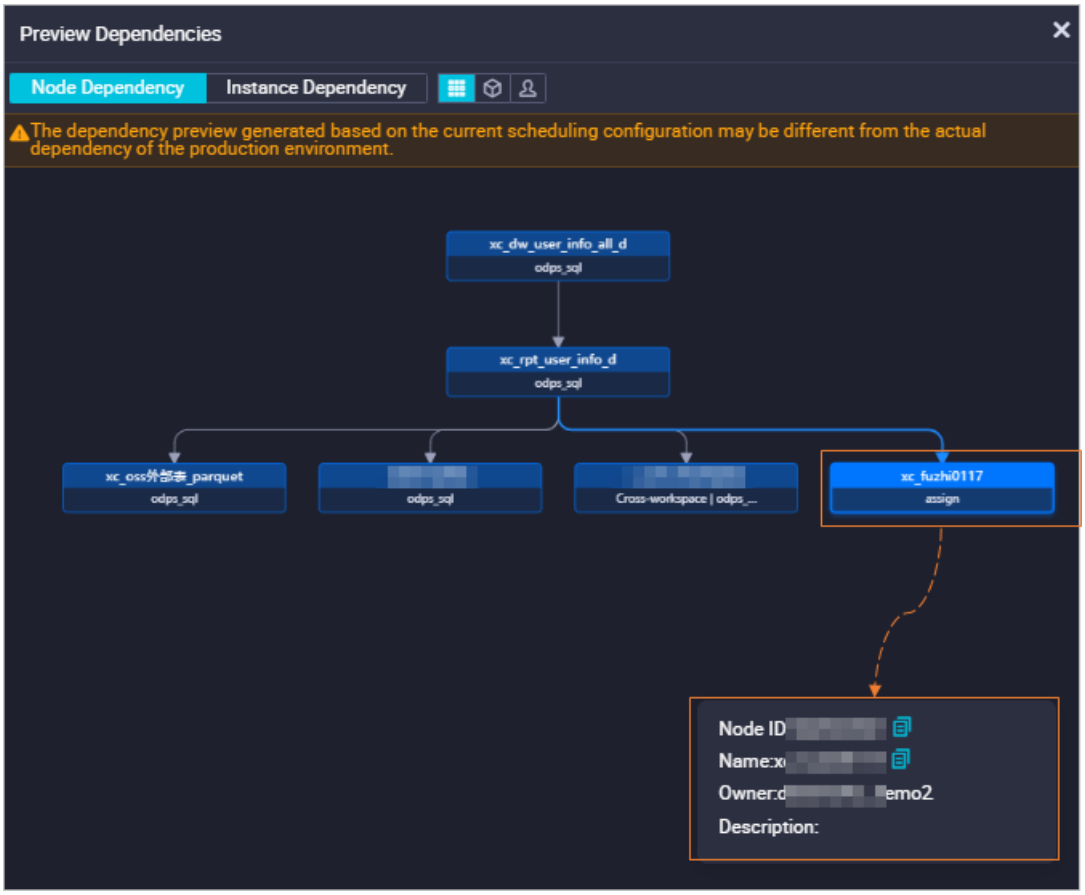
  The following figures show preview effects by using different aggregation methods on the **Node Dependency** tab.





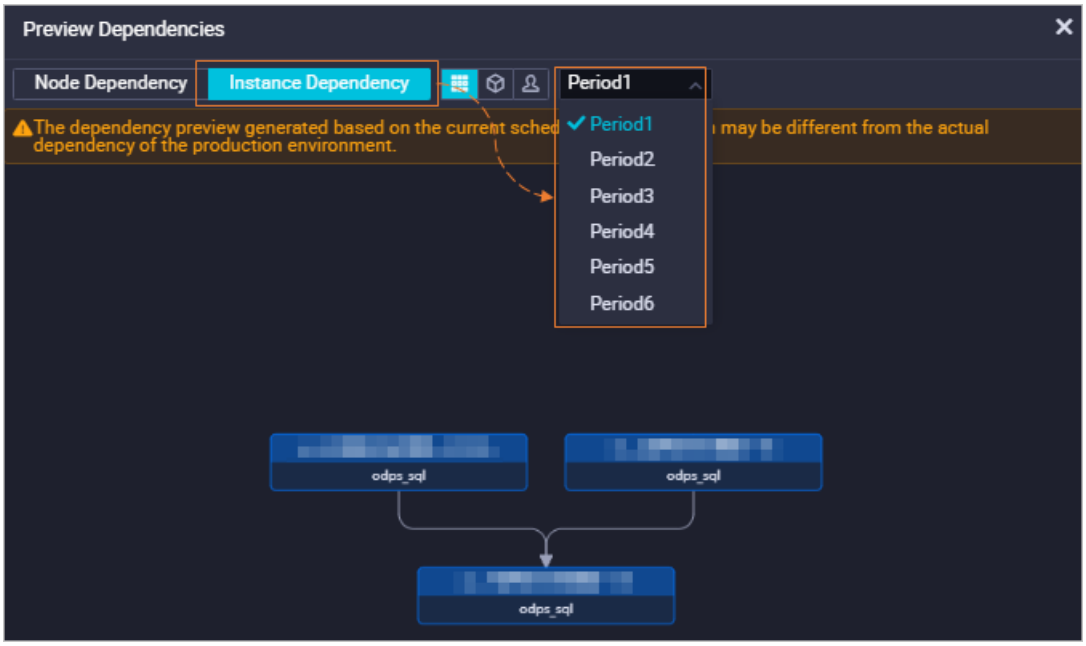


You can click a node to view the basic information about the node.



- Preview scheduling dependencies of an auto triggered node that generates multiple instances.

  For an auto triggered node that generates multiple instances, you can click the **Instance Dependency** tab and select a scheduling cycle to preview scheduling dependencies of the node.



Examples

For more information about typical scenarios of the dependency on instances in the previous cycle, see Scenario 2: Configure scheduling dependencies for a node that depends on last-cycle instances.

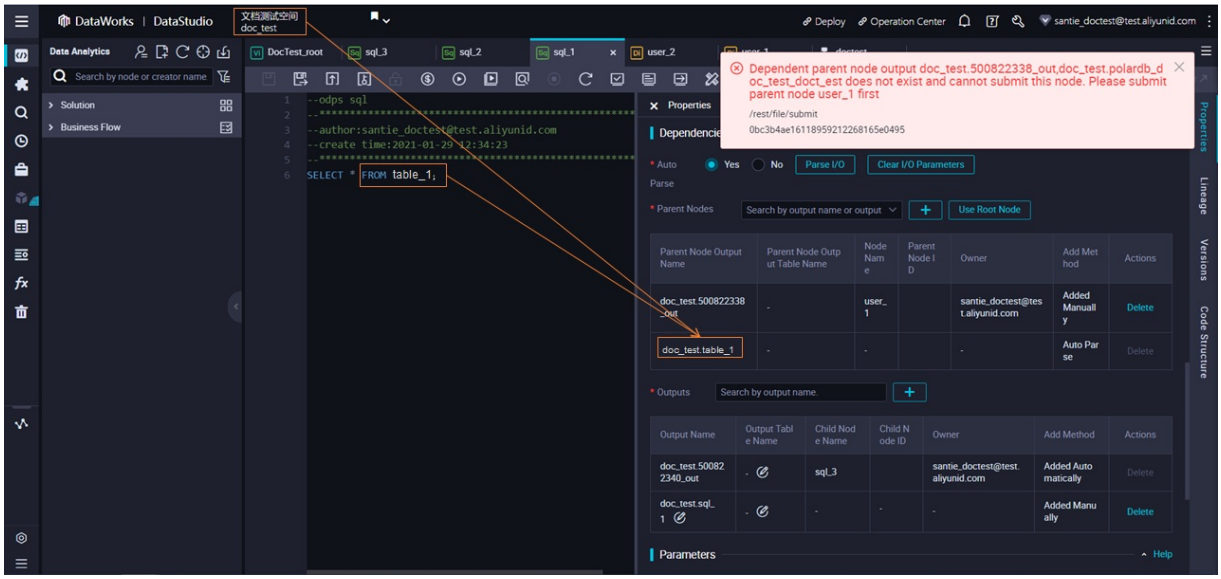
8.5.4. Typical application scenario cases

8.5.4.1. Scenario 1: Configure scheduling dependencies for batch synchronization nodes in a workflow

Scheduling dependencies cannot be automatically added to batch synchronization nodes in a workflow based on the automatic parsing feature. If a node depends on the table generated by its ancestor batch synchronization node, you must manually add the table to the output of the batch synchronization node. This way, when the node queries the table data, the automatic parsing feature can help quickly find the batch synchronization node.

Error message

If you do not manually add the table generated by a batch synchronization node to the output of the batch synchronization node, the system cannot use the automatic parsing feature to find the batch synchronization node. In this case, when you commit an SQL node that depends on the output of the batch synchronization node, an error message shown in the following figure appears.

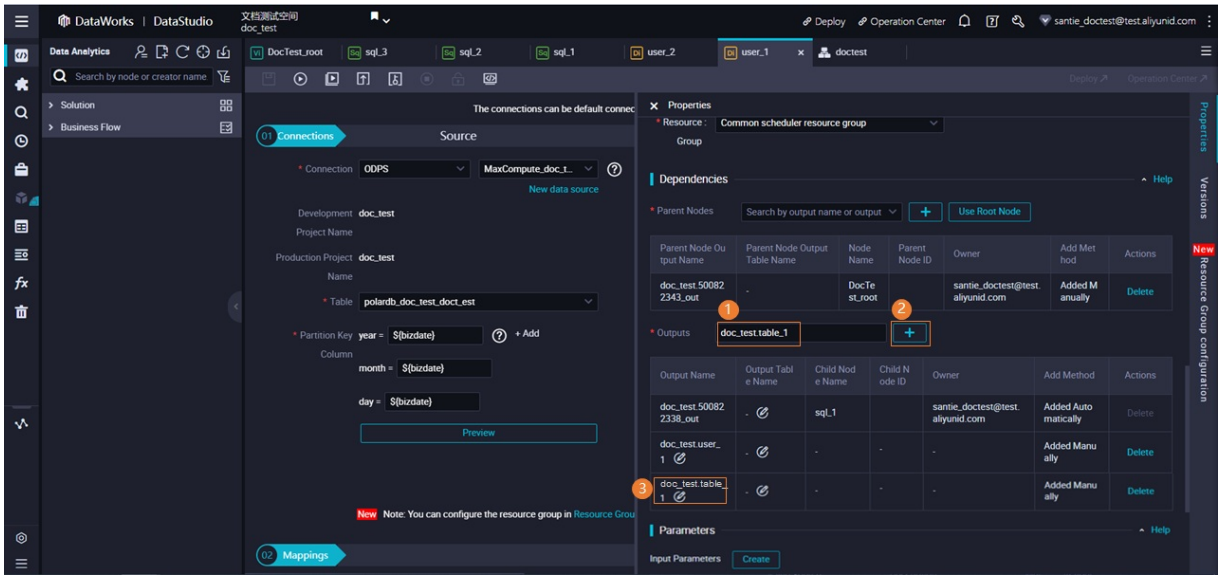


This error occurs because the system cannot find the batch synchronization node on which the SQL node depends based on the upstream dependency that is automatically parsed. For more information, see Error analysis. To prevent this error, we recommend that you use one of the following methods to configure scheduling dependencies for a batch synchronization node:

- Method 1: Manually add the table generated by a batch synchronization node to its output
- Method 2: Keep the name of a batch synchronization node the same as that of its generated table

Method 1: Manually add the table generated by a batch synchronization node to its output

To prevent the preceding error, you can manually add the upstream dependency that is automatically parsed to the output of the batch synchronization node. You can perform the operation on the Properties panel of the batch synchronization node. The following figure shows an example.



Method 2: Keep the name of a batch synchronization node the same as that of its generated table

Based on the preceding descriptions, you can obtain the following information:

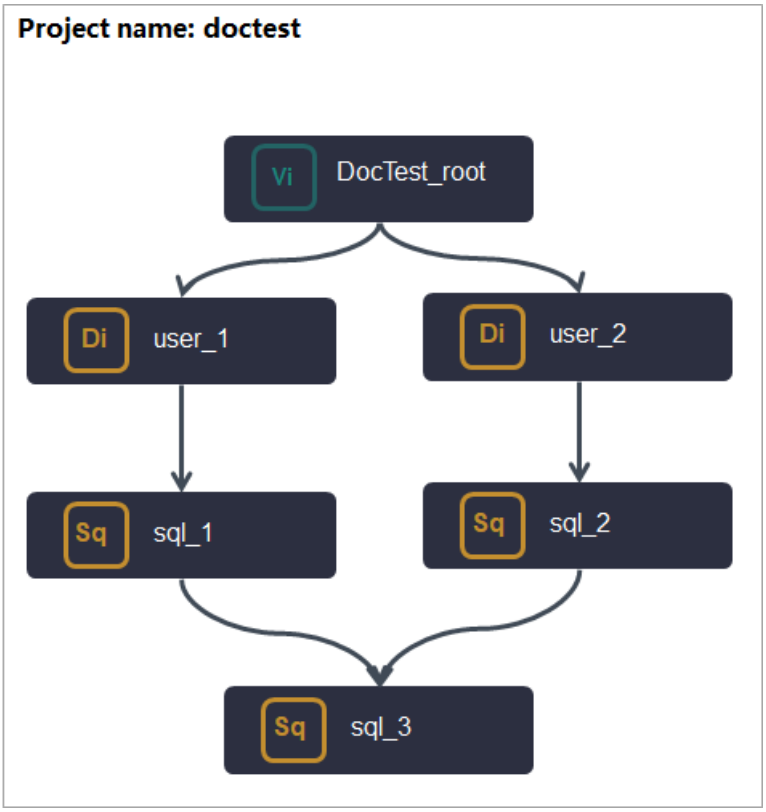
- When you create a batch synchronization node, the system automatically generates an **output** named in the `projectname.nodename` format for the node.
- When the SQL node uses the generated table of the batch synchronization node, the system automatically generates a **dependent ancestor node** named in the `projectname.tablename` format for the SQL node.
- To prevent errors, you must make sure that the name of the **dependent ancestor node** is the same as that of the **output** of the batch synchronization node.

Therefore, to prevent the preceding error when you commit the SQL node, you must keep the name of the batch synchronization node the same as that of the table generated by the batch synchronization node.

> **Note** When you create a node, the system automatically generates an **output** named in the `projectname.nodename` format for the node. If you change the name of the node after the node is created, the name of the **output** does not change. Therefore, this method can be used only when you create a batch synchronization node. If you change the name of a node or a table generated by a batch synchronization node in subsequent operations to ensure consistent names, this error persists.

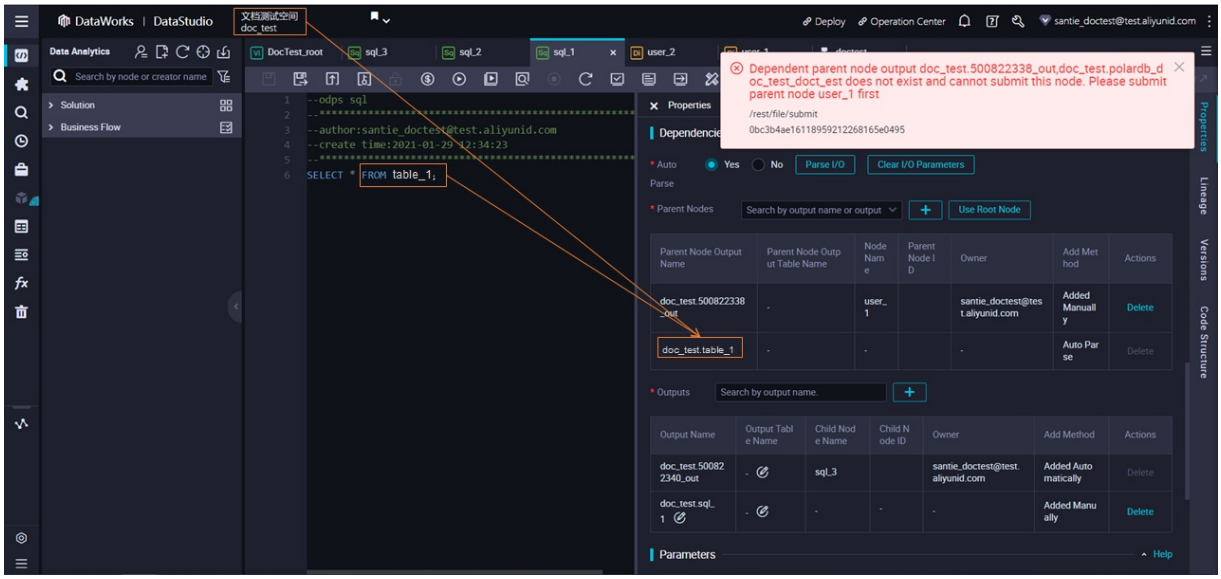
Error analysis

The following figure shows the nodes and scheduling dependencies configured in a workflow that contains batch synchronization nodes.



Step No.	Detailed step	Configured scheduling dependency
1	Create nodes in the workflow based on the planning of the workflow. In the preceding figure, virtual nodes, batch synchronization nodes, and MaxCompute nodes are created.	After the nodes are created in DataWorks, the system automatically generates two **outputs** for each node. One is named in the `projectname.nodename` format, and the name of the other is suffixed with **_out**. For example, the preceding figure shows that after the batch synchronization node user_1 is created, the system automatically generates the following outputs for the node: • One output named `*******_out` • The other output named `doctest.user_1`
2	Connect the nodes by drawing lines based on the planning of the workflow to determine the dependency relationships of the nodes.	After the nodes are connected, the system automatically adds dependency configurations for each ancestor node based on the connections. For example, after you connect the nodes, the MaxCompute node sql_1 in the preceding figure becomes a descendant node of the batch synchronization node user_1. In this case, the system automatically configures an output named `*******_out` of user_1 as a **dependent ancestor node** of sql_1.
3	Develop task code for each node.	When you develop task code for each node, the system automatically parses some input and output commands in the code and adds the **output** or **descendant ancestor node** for each node. For example, the MaxCompute node sql_1 needs to use the data in the `table_1` table generated by the batch synchronization node user_1, and the task code of sql_1 contains statements such as `select * from table_1`. In this case, the system automatically adds a **dependent ancestor node** for sql_1. The output of the automatically added ancestor node is named in the `projectname.tablename` format. In this example, the output name is `doctest.table_1`.

After you perform the preceding operations, if you fail to notice that the system does not automatically add the table generated by the batch synchronization node to the **output** of the batch synchronization node, an error is reported when you commit a node in the workflow. The error indicates that the output name of the dependent ancestor node does not exist.



This error is caused by the following reasons:

- The batch synchronization node user_1 does not support automatic parsing. Therefore, the table_1 table generated by user_1 is not automatically added to the **output** of user_1. This indicates that user_1 does not have an output named `doctest.table_1`.
- The system automatically adds a **dependent ancestor node** named in the `projectname.tablename` format for the descendant node sql_1. In this example, the name of the dependent ancestor node is `doctest.table_1`. However, the system does not add `doctest.table_1` to the output of user_1. Therefore, the system cannot find the ID of user_1.
- When you commit sql_1, the system detects that sql_1 has an upstream dependency `doctest.table_1`. However, the system cannot associate the upstream dependency with the ID of an ancestor node and reports an error indicating that the output name of the dependent ancestor node of sql_1 does not exist.

8.5.4.2. Scenario 2: Configure scheduling dependencies for a node that depends on last-cycle instances

A cross-cycle dependency indicates the dependency of a node on its last-cycle instance, last-cycle instances of its descendant nodes, or last-cycle instances of specified nodes. After you configure a cross-cycle dependency for a node, the node is run in the current cycle only after relevant last-cycle instances are successfully run.

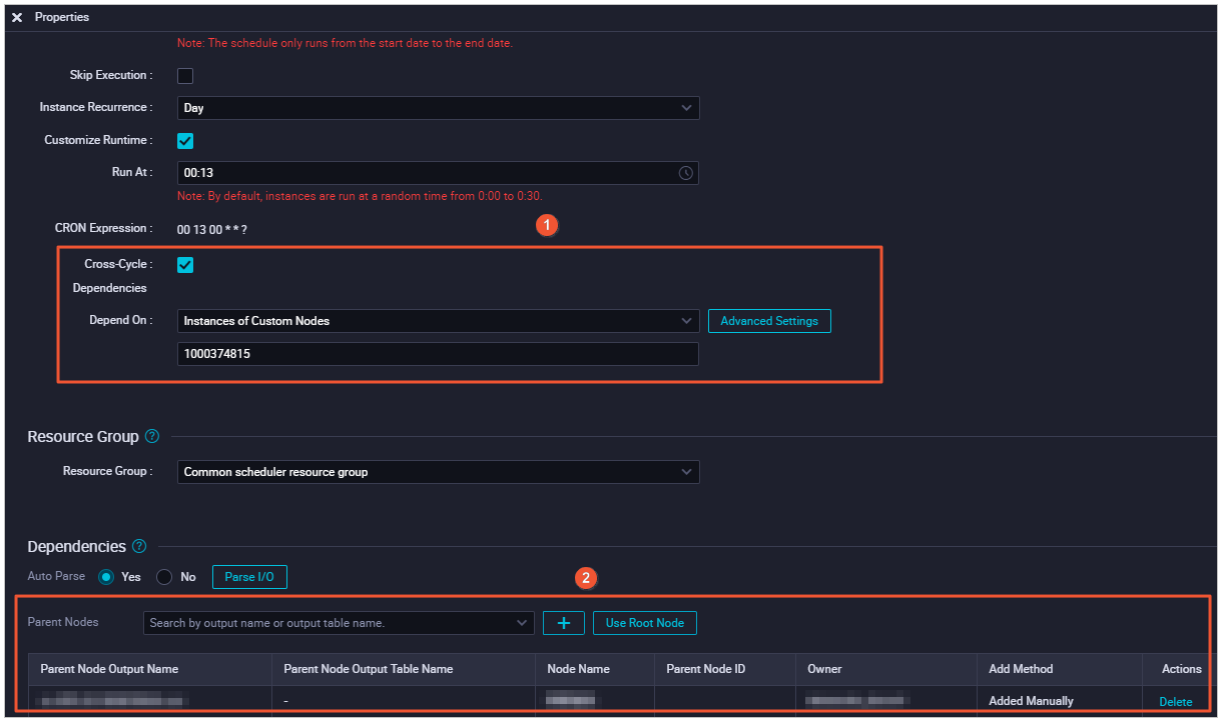
DataWorks supports the following types of cross-cycle dependencies:

- Dependency on the last-cycle instances of descendant nodes
  - Node dependency: A node depends on the last-cycle instances of its descendant nodes. For example, Node A has three descendant nodes: Node B, Node C, and Node D. If you configure this type of dependency for Node A, Node A is run in the current cycle only after all the last-cycle instances of Node B, Node C, and Node D are successfully run.
  - Business scenario: A node is run in the current cycle only after its descendant nodes successfully cleanse the data in the tables generated by the node in the last cycle. To check whether the descendant nodes have cleansed the data, you can configure data quality monitoring rules for the tables generated by the descendant nodes.

- Dependency on the last-cycle instance of the current node
  - Node dependency: A node depends on its last-cycle instance. The node is run in the current cycle only after the last-cycle instance of the node is successfully run.
  - Business scenario: The running of a node in the current cycle depends on the business data that is generated by the last-cycle instance of the node. To check whether the data of the node has been cleansed, you can configure data quality monitoring rules for the table generated by the node.
- Dependency on the last-cycle instances of specified nodes: To configure this type of dependency, you must manually enter the IDs of the nodes that you want a node to depend on. Separate the IDs with commas (,), such as 12345,23456.
  - Node dependency: A node depends on the last-cycle instances of specified nodes. The node is run in the current cycle only after the last-cycle instances of the specified nodes are successfully run.
  - Business scenario: In the business logic, a node depends on the business data that is generated by other nodes but is not processed by the node itself.

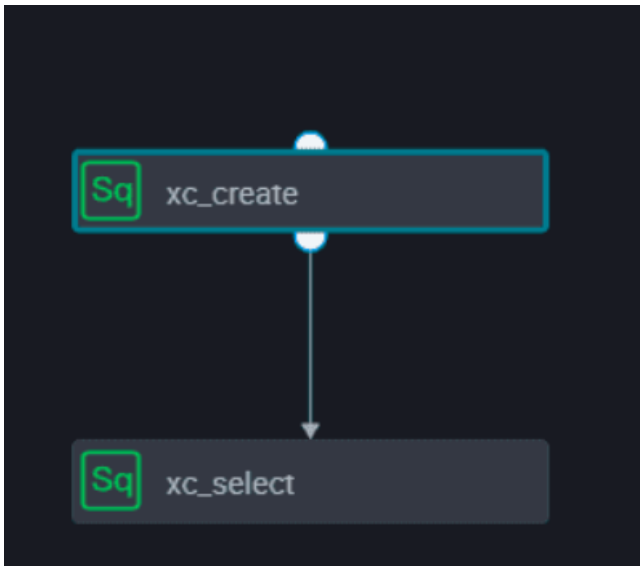
The difference between cross- and same-cycle dependencies is that cross-cycle dependencies appear as dotted lines in Operation Center.

Before you undeploy a node, you must delete the dependencies that are configured for the node, including the cross- and same-cycle dependencies. The following figure shows the Properties panel of a node. You must delete the cross-cycle dependencies for the node in the section marked as 1 and same-cycle dependencies in the section marked as 2.

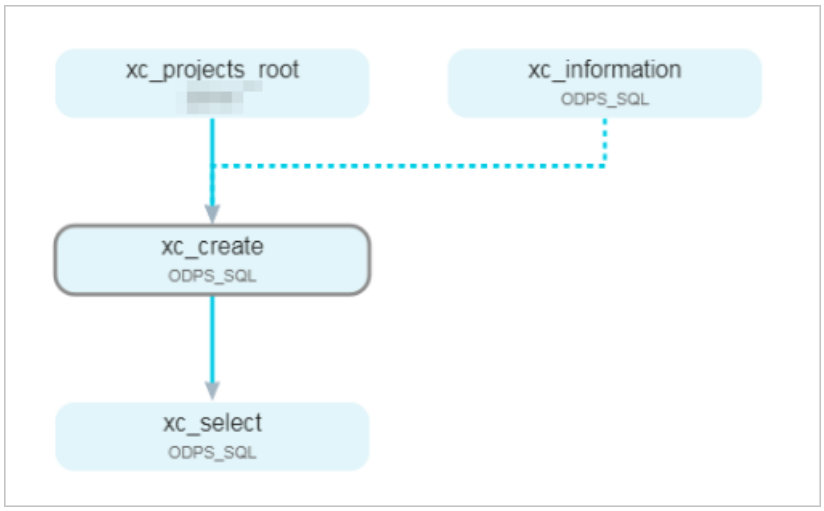


You can configure a cross- or same-cycle dependency on the ancestor node for a node based on your business requirements. In most cases, you can configure either a same- or cross-cycle dependency on the ancestor node for a node. If you enable the automatic parsing feature for a node, the scheduling system automatically configures a same-cycle dependency on the ancestor node for the node. If this configuration does not meet your requirements, you can delete the default same-cycle dependency and configure a cross-cycle dependency for the node. For more information, see Logic of scheduling dependencies.

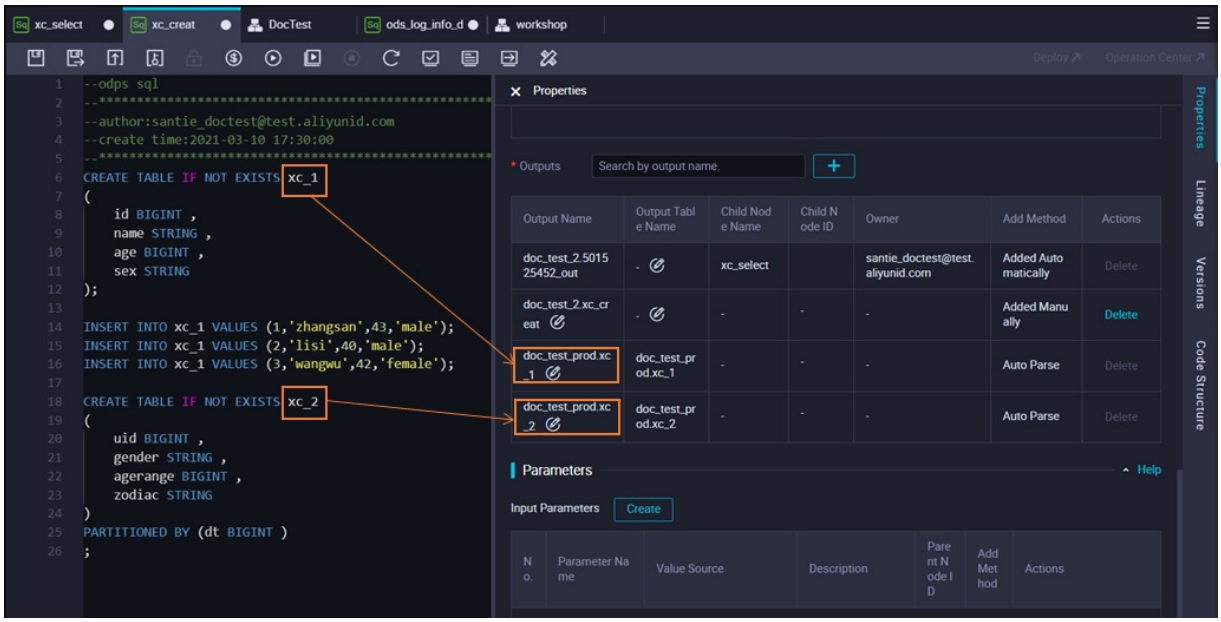
The following figure shows the dependency between the nodes in a workflow.



The following figure shows how the dependency appears in Operation Center.

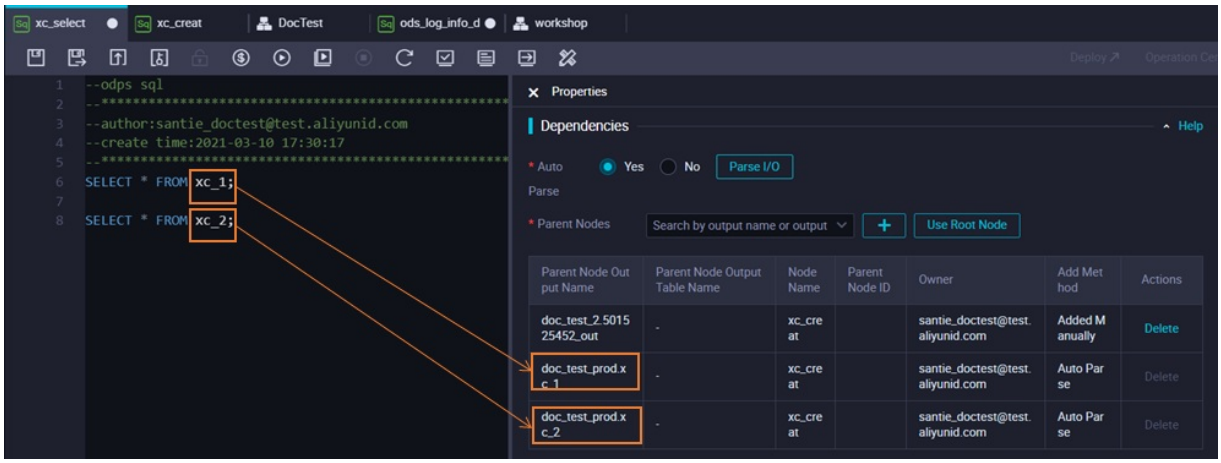


The following figure shows the code and configurations of the xc_create node.



In the preceding figure, the xc_create node creates the xc_1 and xc_2 tables and inserts data to the two tables. The xc_1 and xc_2 tables are the outputs of the xc_create node.

The following figure shows the code and configurations of the xc_select node.



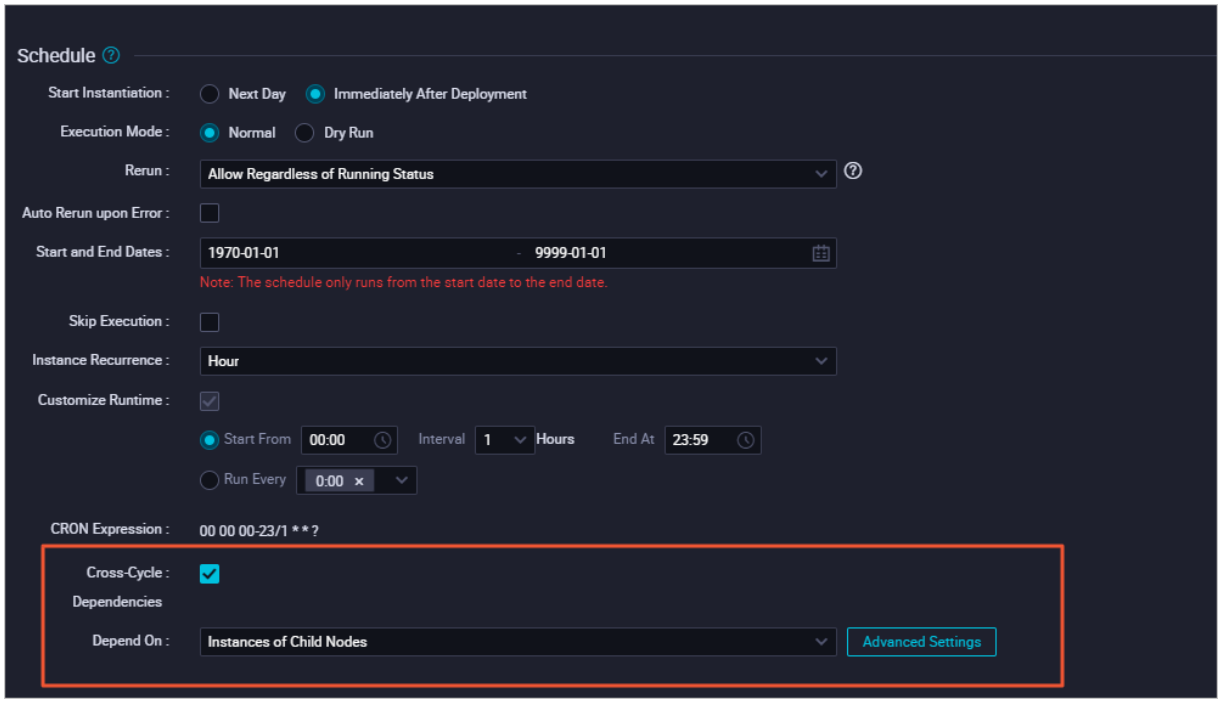
In the preceding figure, the xc_select node queries data in the xc_1 and xc_2 tables. Based on the automatic parsing feature, the same-cycle dependency on the xc_create node is automatically configured for the xc_select node.

Dependency on the last-cycle instances of descendant nodes

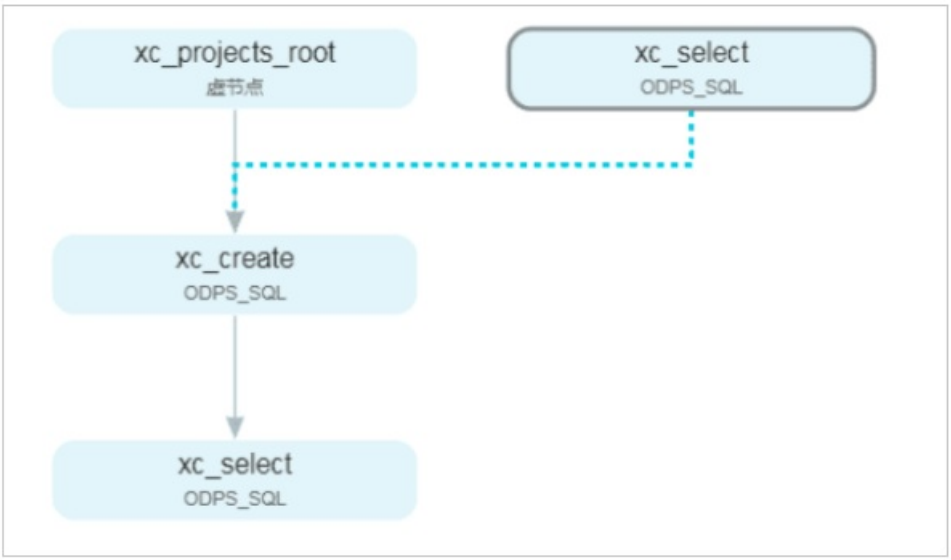
Node dependency: A node depends on the last-cycle instances of its descendant nodes. For example, Node A has three descendant nodes: Node B, Node C, and Node D. If you configure this type of dependency for Node A, Node A is run in the current cycle only after all the last-cycle instances of Node B, Node C, and Node D are successfully run.

Business scenario: A node is run only after its descendant nodes have successfully cleansed the data in the tables generated by the node in the last cycle. Otherwise, the node is not run in the current cycle.

When you configure dependencies for the xc_create node, select Cross-Cycle Dependencies and set Depend On to Instances of Child Nodes.



The following figure shows how the dependency appears in Operation Center.



Dependency on the last-cycle instance of the current node

Node dependency: A node depends on its last-cycle instance. The node is run in the current cycle only after the last-cycle instance of the node is successfully run.

Business scenario: The running of a node in the current cycle depends on the business data that is generated by the last-cycle instance of the node. In this example, the node is scheduled to run by week. This way, you can conveniently view the dependencies of the node in Operation Center.

To view the dependencies of the node, go to **Operation Center**. In the left-side navigation pane, choose **Cycle Task Maintenance > Cycle Instance**. Search for the node to view its dependencies.

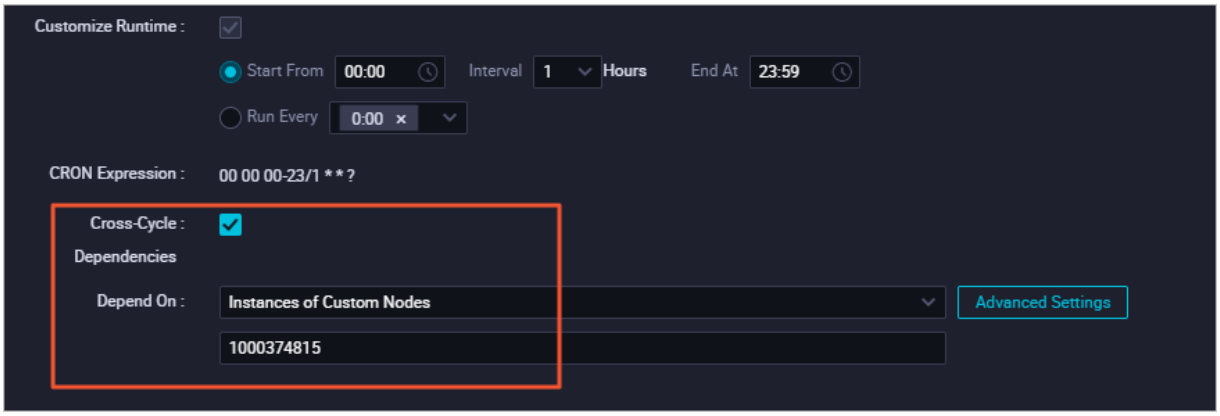
> **Note** For example, a node is scheduled to run by hour and depends on its last-cycle instance. If the instance that is generated for a specific hour is not successfully run, the system does not run the instance that is generated for the next hour.
>
> If the first instance that is generated on a day is not run or the running fails, the instances that are generated for the rest of the day cannot be run.

Dependency on the last-cycle instances of specified nodes

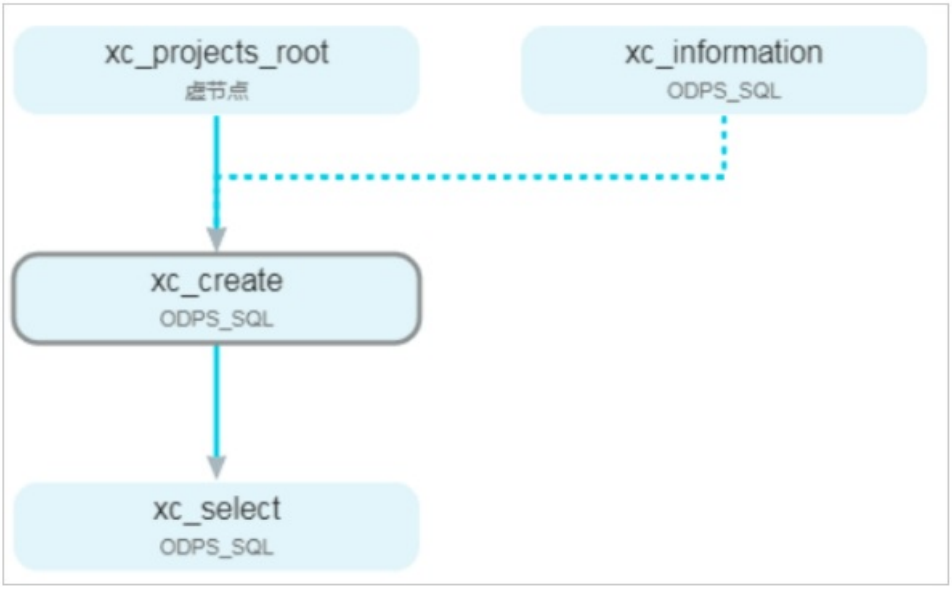
Node dependency: The tables generated by the xc_information node are not used in the code of the xc_create node. However, the xc_create node depends on the output data of the xc_information node in the last cycle, as configured in the business logic. Logically, the xc_create node depends on the last-cycle instance of the xc_information node.

Business scenario: Node A depends on the business data that is generated by Node B based on the business logic. However, the business data is not referenced in the code of Node A. This indicates that Node A performs no operations on the business data.

In this example, when you configure the xc_create node, select Cross-Cycle Dependencies, set Depend On to Instances of Custom Nodes, and then enter the ID of the xc_information node, which is 1000374815.



To view the dependencies of the node, go to **Operation Center**. In the left-side navigation pane, choose **Cycle Task Maintenance > Cycle Instance**. Search for the node to view its dependencies.

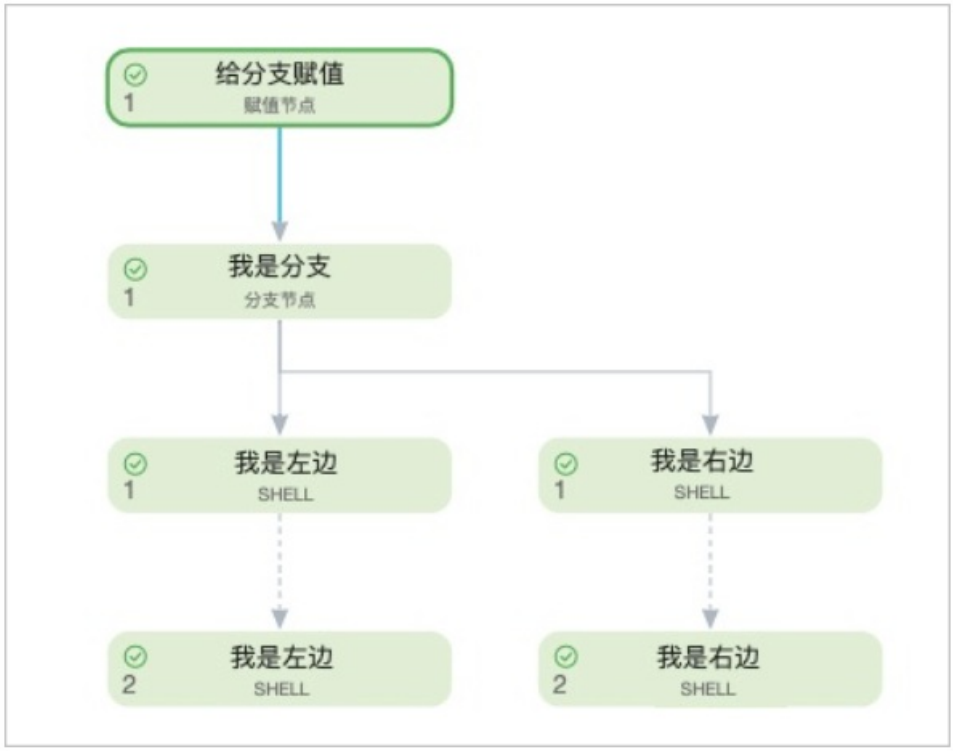


Advanced configuration for dependency on the last-cycle instances

A branch node has two descendant nodes. In most cases, only one of the descendant nodes is actually run. The scheduling system generates and runs an instance for one descendant node. For the other descendant node, the scheduling system generates an instance and directly returns a successful response without running the instance. In addition, the scheduling system also performs a dry run for the descendant node of this descendant node. If this does not meet your requirements, you can select **Upstream node air running attribute does not conduct cross-cycle** when you configure this descendant node of the branch node.

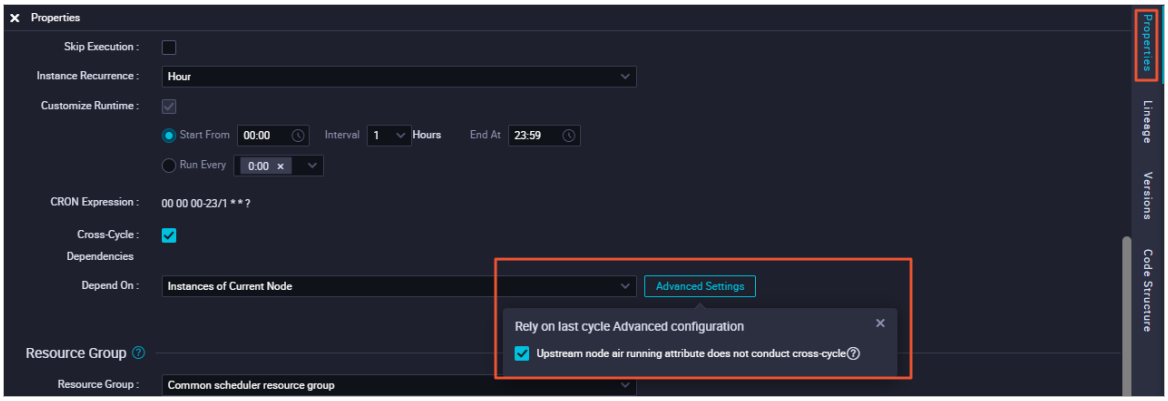
If a descendant node of a branch node depends on its own last-cycle instance and the last-cycle instance is dry-run, the descendant node is also dry-run in the current cycle. As a result, the descendant node becomes a dry-run node permanently.

The following figure shows an example. If the scheduling system performs a dry run for the descendant node on the left, the scheduling system also performs a dry run for the descendant node of this descendant node.



Your business may require that the running of the descendant node of a branch node depend only on the result of the branch node in the current cycle, and that the descendant node not be affected by its dry run in the last cycle. To meet this requirement, perform the following steps:

1. On the editing tab of this descendant node of the branch node, click **Properties** in the right-side navigation pane.
2. In the **Schedule** section of the Properties panel, select **Cross-Cycle Dependencies**.
3. Click **Advanced Settings**.
4. In the Rely on last cycle Advanced configuration message, select **Upstream node air running attribute does not conduct cross-cycle**. This way, the descendant node of the branch node is not affected by its dry run in the last cycle.



> **Note** This advanced configuration applies only to the descendant nodes of branch nodes. For other nodes, a dry run in the last cycle does not affect the running in the current cycle.

Typical scenarios of cross-cycle dependencies

- Scenario 1
  - Scenario description: Node A is scheduled to run by day. Node B is scheduled to run by hour. Node A depends on Node B. By default, Node A is run at the end of each day after Node B has been run for 24 times. However, you want Node A to be run at 12:00 every day.
  - Solution: When you configure Node B, select **Cross-Cycle Dependencies** and set Depend On to **Instances of Current Node**. When you configure Node A, set Run At to 12:00. Do not configure cross-cycle dependencies for Node A.

    This way, after an instance is generated and run for Node B at 12:00, the scheduling system runs Node A.
- Scenario 2
  - Scenario description: Node A is scheduled to run by day. Node B is scheduled to run by hour. Node A depends on the data that is generated by Node B on the previous day.
  - Solution: When you configure Node A, select **Cross-Cycle Dependencies**, set Depend On to **Instances of Custom Nodes**, and then enter the ID of Node B.
- Scenario 3
  - Scenario description: Node A is scheduled to run by hour. Node B is scheduled to run by day. Node A depends on Node B. After Node B is run on a day, Node A has gone through 24 cycles, and the scheduling system starts to generate and run 24 instances at the same time.
  - Solution: When you configure Node A, select **Cross-Cycle Dependencies** and set Depend On to **Instances of Current Node**.
- Scenario 4
  - Scenario description: A node depends on the data that is generated by the node in the last cycle. The time at which the data in the last cycle was generated needs to be determined.
  - Solution: When you configure the node, select **Cross-Cycle Dependencies** and set Depend On to **Instances of Current Node**.

8.5.4.3. Scenario 3: Configure scheduling dependencies for nodes across workflows or workspaces

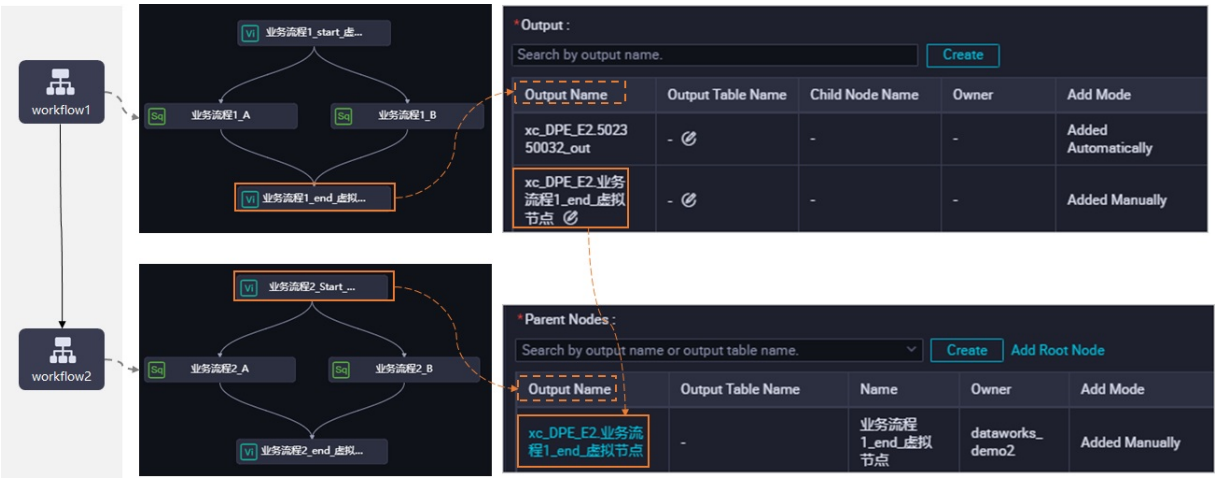
This topic describes how to configure scheduling dependencies for nodes across workflows or workspaces.

Configure scheduling dependencies for nodes across workflows

If you want to configure scheduling dependencies for branch nodes in a workflow on nodes in another workflow, you must use a zero load node to aggregate the outputs of the branch nodes in the current workflow, and configure the output of the zero load node as the input of the root node of the other workflow.

Note A zero load node is a control node that supports only dry-run scheduling and does not generate data. In most cases, a zero load node serves as the root node of a workflow or the output node of multiple branch nodes. If a workflow contains multiple branch nodes, you must create a zero load node and configure the zero load node as the descendant node of the branch nodes. For example, you can create a zero load node named in the format of `Workflow_end_Zero load node`. This way, the zero load node depends on the outputs of the branch nodes. After the `zero load node` is successfully run, the workflow is also successfully run.

The following example shows how to use zero load nodes to configure scheduling dependencies for branch nodes across workflows.



- Create two workflows: Workflow 1 and Workflow 2. Then, configure Workflow 1 as the ancestor workflow of Workflow 2.
- Create the following zero load nodes in Workflow 1:
  - `Workflow1_start_Zero load node`: the start node of the branch nodes in Workflow 1.
  - `Workflow1_end_Zero load node`: the output node that is used to aggregate the outputs of branch nodes in Workflow 1.
- Create the following zero load nodes in Workflow 2:
  - `Workflow2_start_Zero load node`: the start node of the branch nodes in Workflow 2.
  - `Workflow 2_end_Zero load node`: the output node that is used to aggregate the outputs of branch nodes in Workflow 2.
- Configure scheduling dependencies between Workflow 1 and Workflow 2: Configure the output of `Workflow1_end_Zero load node` as the input of `Workflow2_start_Zero load node`. This way, scheduling dependencies for branch nodes across workflows are configured.

> **Note** DataWorks allows you to configure the output of a node as the input of another node to establish dependencies between the nodes. You can use one of the following methods to configure scheduling dependencies: drag nodes in a directed acyclic graph (DAG) to configure dependencies between the nodes, enable the automatic parsing feature, and configure dependencies. In the preceding example, the scheduling dependencies are configured by configuring the output of `Workflow1_end_Zero load node` as the input of `Workflow2_start_Zero load node`.
>
> - For more information about how to create a workflow, see Create a workflow.
> - For more information about how to create a zero load node, see Create a zero-load node.
> - For more information about how to configure scheduling dependencies between nodes, see Configure same-cycle scheduling dependencies.

Configure scheduling dependencies for nodes across workspaces

DataWorks also allows you to configure scheduling dependencies for nodes across workspaces that reside in the same region. You can configure the output of a node in a workspace as the input of a node in another workspace to establish scheduling dependencies between the nodes. For example, you can configure the output of Node A in Workspace A as the input of Node B in Workspace B to establish scheduling dependencies between the nodes. The configuration method is the same as the method that is used to configure scheduling dependencies in common scenarios. For more information, see Configure same-cycle scheduling dependencies.

> **Note** Nodes in workspaces in standard mode cannot depend on nodes in workspaces in basic mode. To resolve this issue, submit a ticket.

8.6. Configure input and output parameters

Input and output parameters are used to transmit data between ancestor and descendant nodes. This topic describes how to define and use input and output parameters.

Context

After you define an output parameter and its value for a node, you can define an input parameter for the descendant node of the node and configure the descendant node to reference the value of the output parameter in the input parameter.

Precautions

Input and output parameters are used together with assignment nodes. For more information about assignment nodes, see Configure an assignment node. Some types of nodes support the assignment parameter. The assignment parameter functions the same as an assignment node. For more information about how to configure the assignment parameter, see the Assignment parameter section in this topic and the topic about how to configure an assignment node.

Configure input and output parameters

1. Log on to the DataWorks console.

2. In the left-side navigation pane, click **Workspaces**.
3. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **Data Development** in the Actions column.
4. In the **Scheduled Workflow** pane of the DataStudio page, double-click a desired node to go to the configuration tab of the node.
5. Click the **Properties** tab in the right-side navigation pane. On the Properties tab, configure the input and output parameters in the **Input and Output Parameters** section.

   For more information about how to configure scheduling parameters for different types of nodes, see SQL nodes and batch synchronization nodes.

Output parameters

You can define output parameters in the **Input and Output Parameters** section. The following value types are supported for output parameters: **Constant** and **Variable**.

After you define an output parameter for the current node and commit the current node, the output parameter can be referenced as an input parameter for a descendant node of the current node.

> **Note** You cannot write code for the current node to assign a value to the defined output parameter.

Field	Description	Remarks
No.	The serial number, which is automatically increased by 1 each time you define an output parameter.	N/A
Parameter Name	The name of the output parameter.	N/A
Type	The value type of the output parameter.	Valid values: **Constant** and **Variable**.
Value	The value of the output parameter.	The value type can be Constant or Variable. • If the Type field is set to Constant, the value is a fixed string. • If the Type field is set to Variable, the value can be a global variable, a built-in scheduling parameter, or a custom scheduling parameter in the ${...} or $[...] format.
Description	The description of the output parameter.	N/A

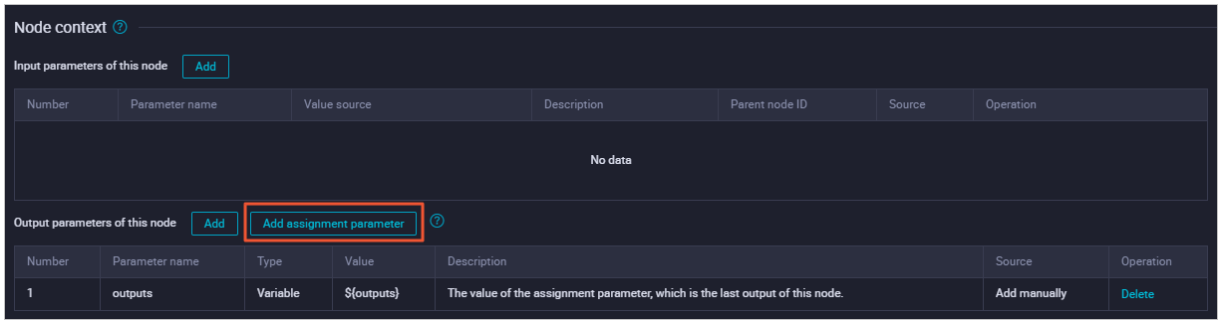
Field	Description	Remarks
Add Method	The way in which the output parameter is defined.	Valid values: **Added Automatically**, **Auto Parse**, and **Added Manually**.
Actions	You can click **Change** or **Delete** to perform the related operation.	These two operations are not supported if a node depends on the current node. Before you configure a node to reference the output parameter of the current node, make sure that the output parameter is correctly defined on the current node.

Assignment parameter

If the current node supports the assignment parameter and you want to pass the query results of the current node to its descendant node as a parameter, you can perform the following operations: Go to the configuration tab of the current node. Click the **Properties** tab in the right-side navigation pane. In the **Output Parameters** table in the **Input and Output Parameters** section, click **Add assignment parameter**. The outputs assignment parameter is added. The following types of nodes support the assignment parameter: EMR Hive, EMR Spark SQL, ODPS Script, Hologres SQL, AnalyticDB for PostgreSQL, and MySQL.

> **Note** The assignment parameter is supported only in DataWorks Standard Edition or a more advanced edition.

After you click **Add assignment parameter** to add the outputs assignment parameter, a descendant node of the current node can reference the outputs parameter to obtain the query results of the current node. If the query results of the current node are empty, the current node is not blocked. However, the descendant node that references the outputs assignment parameter of the current node may fail to run.



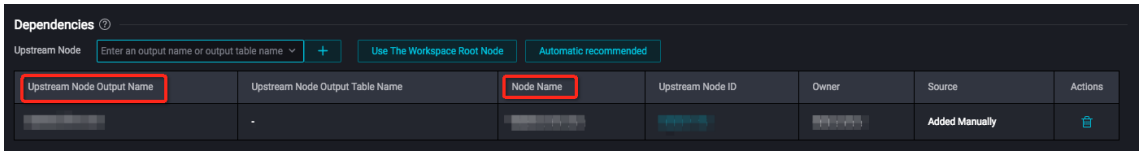
You must add the assignment parameter of the current node as an input parameter of a descendant node of the current node. The assignment parameter can be referenced by using a two-dimensional array in the code. The assignment parameter functions the same as an assignment node in the ODPS SQL language. For more information, see Configure an assignment node.

Input parameters

An input parameter of a node defines the reference to the output of an ancestor node of the node. The input parameter can be used in the same way as other parameters.

The following descriptions show how to define and use an input parameter:

- Define an input parameter
  i. In the **Dependencies** section of the Properties tab on the configuration tab of a desired node, add a node on which the current node depends.

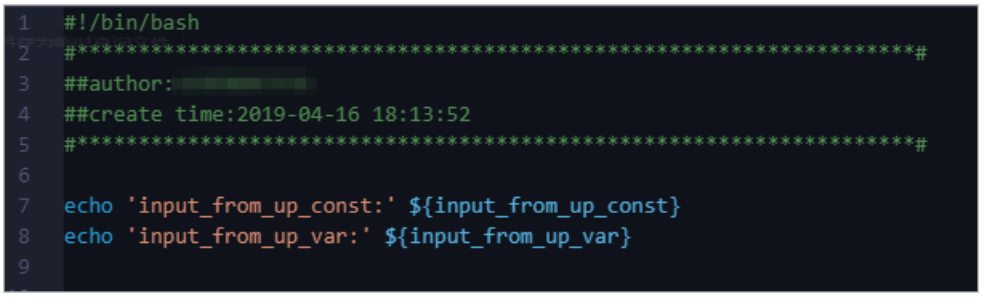


  ii. In the Input Parameters table in the **Input and Output Parameters** section, define an input parameter and configure this parameter to reference the output parameter of an ancestor node of the current node.

Field	Description	Remarks
No.	The serial number, which is automatically increased by 1 each time you define an input parameter.	N/A
Parameter Name	The name of the input parameter.	N/A
Value Source	The value source of the input parameter.	The value of the input parameter is the value of the output parameter of the ancestor node.
Description	The description of the input parameter.	N/A
Parent Node ID	The ID of the ancestor node.	The system parses the ID of the ancestor node.
Add Method	The way in which the input parameter is defined.	Valid values: **Added Automatically**, **Auto Parse**, and **Added Manually**.
Actions	You can click **Change** or **Delete** to perform the related operation.	N/A

- Use the input parameter

The defined input parameter is referenced in the code of a node in the same way as other built-in scheduling parameters. The input parameter is configured in the `${Input parameter name}` format. The following figure shows how to reference an input parameter in the code of a Shell node.



Supported built-in scheduling parameters

- Built-in scheduling parameters

Built-in scheduling parameter	Description
${projectId}	The ID of the MaxCompute project.
${projectName}	The name of the MaxCompute project.
${nodeId}	The ID of the node.
${gmtdate}	The date on which the node instance is scheduled to run, in the yyyy-MM-dd 00:00:00 format.
${taskId}	The ID of the node instance.
${seq}	The sequence number of the node instance, which indicates the ranking of this node instance among all node instances on the same day.
${cyctime}	The time at which the node instance is scheduled to run.
${status}	The status of the node instance. Valid values: SUCCESS and FAILURE.
${bizdate}	The data timestamp.
${finishTime}	The time at which the node instance finishes running.
${taskType}	The mode in which the node instance is run. Valid values: NORMAL, MANUAL, PAUSE, SKIP, UNCHOOSE, and SKIP_CYCLE.
${nodeName}	The name of the node.

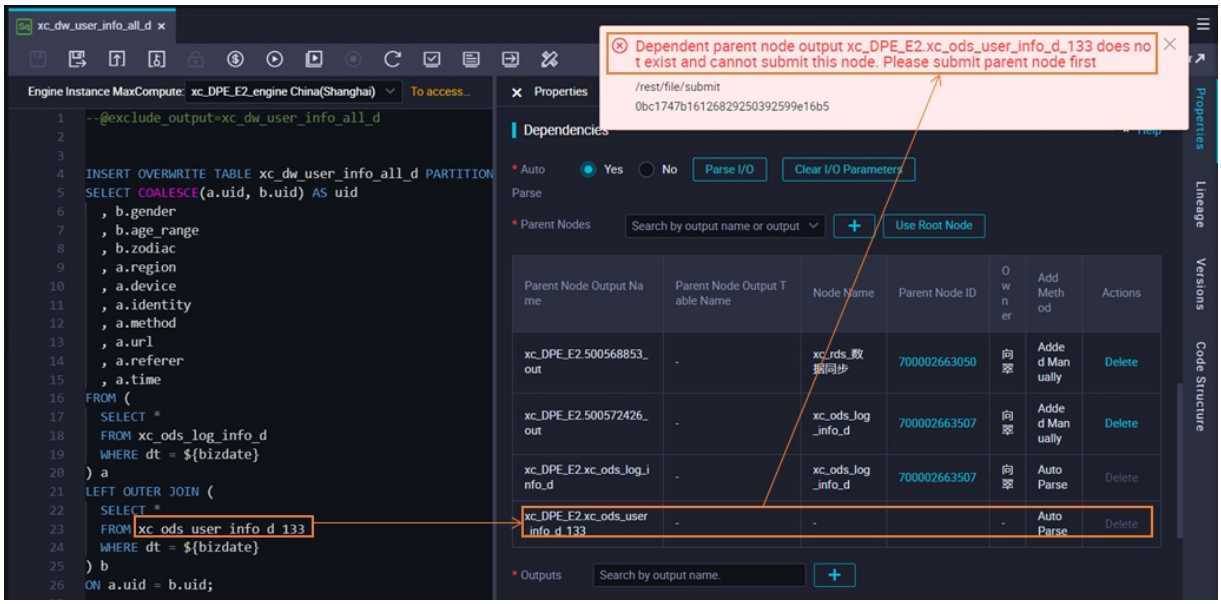
- For more information about other parameters, see Overview of scheduling parameters.

8.7. FAQ

8.7.1. When I commit Node A, the system reports an error that the output name of the dependent ancestor node of Node A does not exist. What do I do?

Problem description

When I commit Node A, the system reports an error that the output name of the dependent ancestor node of Node A does not exist.



The preceding figure is used to explain the error. The error occurs because the system cannot find the ancestor node that generates the `xc_ods_user_info_d_133` table based on the **output name of the dependent ancestor node** configured for Node A.

> **Note** The error indicates that **the output name of a node does not exist**. This does not mean that the xc_ods_user_info_d_133 table does not exist. Instead, this means that no node uses the output name of the node as an output. If the xc_ods_user_info_d_133 table exists and is generated by a node, but the system does not add the table to the output of the node, the error is also reported.

Possible cause 1: No node generates the table

- Possible cause:

  No node generates the table.

  In most cases, the system can automatically add a table generated by a node to the output of the node by using the automatic parsing feature. However, tables generated by nodes other than auto triggered nodes do not support the feature. Nodes other than auto triggered nodes generate the following tables:

- Tables uploaded from on-premises machines to DataWorks
- Dimension tables
- Tables that are not generated by nodes scheduled by DataWorks
- Tables generated by manually triggered nodes

If the code developed for a node contains the SELECT statement that specifies one of the preceding tables, the preceding error is reported.

- Solution:

  You must delete the dependency that contains tables generated by nodes other than auto triggered nodes. In this example, you must delete the **output name of the dependent ancestor node** that contains the `xc_ods_user_info_d_133` table.

  For more information about how to delete a scheduling dependency, see Manual configuration.

Possible cause 2: The table that is generated by a node is not added to the output of the node

- Possible cause:

  The table is generated by a node. However, the table is not added to the output of the node.

  In most cases, the system can automatically add a table generated by a node to the output of the node based on the automatic parsing feature. However, some nodes do not support the feature. Batch synchronization nodes, AnalyticDB for PostgreSQL nodes, AnalyticDB for MySQL nodes, and EMR nodes do not support configuration of scheduling dependencies based on the automatic parsing feature. The tables generated by these nodes must be manually added to the outputs of these nodes.

  If the SELECT statement in the code of a node specifies one of the preceding tables, and the table is not added to the output of the node that generates the table, the preceding error is reported.

- Solution:

  You must manually add the table to the output of the node that generates the table. In this example, you must manually add the `xc_ods_user_info_d_133` table to the **output** of the node that generates the table.

  For more information about how to add a scheduling dependency, see Manual configuration.

To prevent data errors caused by incorrect dependency configurations, the system compares the input and output of the table data lineage with those of the scheduling configurations when you commit a node. If they are inconsistent, the system displays an error message. For more information, see When I commit a node, the system reports an error that the input and output of the node are not consistent with the data lineage in the code developed for the node. What do I do?.

You do not need to configure node dependencies for table data generated by a node that is not scheduled to run by day. If you have configured such a dependency, it can be deleted. In this case, the system displays an error message that the lineage of the table data is inconsistent with the scheduling configurations when you commit the node. If the inconsistency is caused only by the deletion, you can forcibly commit the node.

Possible cause 3: Multiple nodes have duplicate output names

- Possible cause:

  The **output** names of multiple nodes are the same. This may be caused by the following reasons:

- Multiple nodes generate the same table.

  If multiple nodes generate the same table, and the code developed for a node contains the SELECT statement that specifies the table, the system cannot find the unique node that generates the table, and the error is reported when you commit your node.

- A workspace contains nodes that have the same name.

  When you create a node, the system automatically adds two **outputs** for the node. One is named in the `projectname.nodename` format. If a workspace contains two nodes that have the same name, the **output** names of these two nodes are the same. In this case, when you commit the created node, the system reports the error.

- Solution:

  You must change the names of the nodes based on code development specifications and UI node naming suggestions.

  - A table is generated by only one node, and the table must be configured as the output of the node.
  - Each node name must be unique in a workspace.

  After you change the names of the nodes, make sure that nodes have different **output** names.

8.7.2. When I commit a node, the system reports an error that the input and output of the node are not consistent with the data lineage in the code developed for the node. What do I do?

Problem description

When I commit a node, the system reports an error that the input and output of the node are not consistent with the data lineage in the code developed for the node.

Possible causes

The table specified in the SELECT statement is inconsistent with the table added to Parent Nodes for the node, or the table specified in the INSERT or CREATE statement is inconsistent with the table added to Outputs for the node.

For example, in the preceding figure:

- The SELECT statement in the code of the node that you committed specifies table2, but table2 is not added to Parent Nodes for the committed node.
- doc_test is added to Outputs for the committed node, but the INSERT or CREATE statement in the code of the node does not specify a table named doc_test.

Solution

- If a table that is generated not by an auto triggered node is configured as the input or output of a node, you can ignore the error and commit the node.

  Node dependencies ensure that a node can successfully obtain the table data generated by its ancestor node that is scheduled to run. However, if the ancestor node is not scheduled to run, the system cannot detect whether the ancestor node has generated the latest table data. If the SELECT statement in the code of a node specifies a table that is generated not by an auto triggered node, you can remove the table that is specified by the SELECT statement from Parent Nodes for the committed node. Tables that are generated not by auto triggered nodes include the following types:

  - Tables uploaded from on-premises machines to DataWorks
  - Dimension tables
  - Tables that are generated not by nodes scheduled by DataWorks
  - Tables generated by manually triggered nodes

- If a table is generated by an auto triggered node, you must check whether the data lineage and scheduling dependencies are correctly configured.

  If you forcibly commit a node without checking the preceding item, the following errors may occur:

  - Descendant nodes cannot obtain data from the ancestor nodes on which they depend. For example, the SELECT statement in the code of a node specifies Table A, and Table A is generated by a node scheduled to run every day. If Table A is not added to Parent Nodes for the node, and the execution of the scheduled node that generates Table A fails one day, the node may fail to obtain the data generated by the last execution of the scheduled node.
  - The output name of an ancestor node does not exist. For example, the CREATE or INSERT statement in the code of Node A specifies Table B, but Table B is not configured as the output of Node A. In this case, if the SELECT statement in the code of Node B specifies Table B, the system automatically configures Table B as the input of Node B to establish a dependency relationship between Node A and Node B. However, the system cannot find Node A based on the dependency relationship. Therefore, when you commit Node B, the system reports an error that the output name of the ancestor node of Node B does not exist. For more information, see When I commit Node A, the system reports an error that the output name of the dependent ancestor node of Node A does not exist. What do I do?.

9.Debug and submit publishing tasks

9.1. Debugging and viewing tasks

9.1.1. Debug a code snippet: Quickly run a code snippet

DataWorks provides the quick run feature to allow you to quickly run the code snippet that you select on the configuration tab of a node. You can use this feature to test whether a code snippet is correctly written. This topic describes how to quickly run a code snippet of a node.

Prerequisites

An ODPS SQL node is created, and the code of the ODPS SQL node is written. For more information about how to create an ODPS SQL node, see Create an ODPS SQL node.

Limits

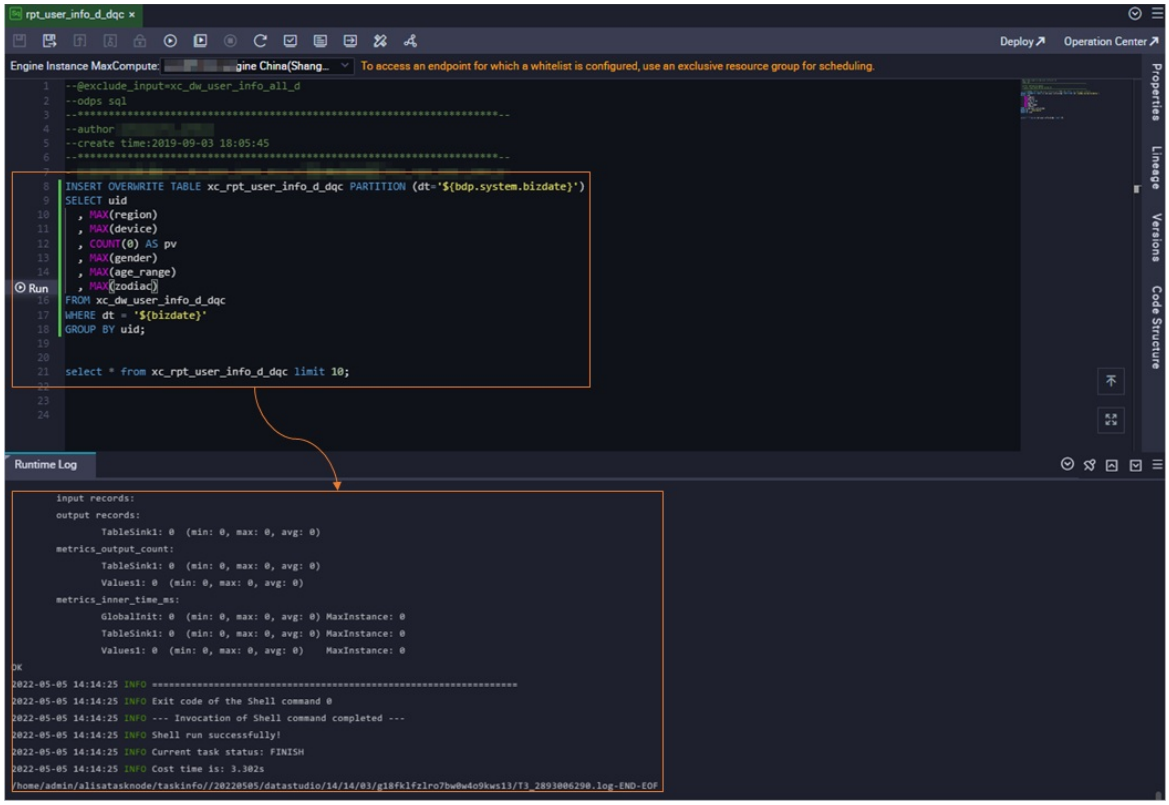
- You can use the quick run feature to quickly run the code of only an ODPS SQL node.
- You can use the quick run feature to quickly run the code only of nodes that are not running. If the code of a node is being executed, the quick run icon

is not displayed on the left side to the code line of the node.

Procedure

1. Go to the DataStudio page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region where the workspace that you want to manage resides. Find the workspace and click **Data Development** in the **Actions** column.
2. In the left-side navigation pane, click **Scheduled Workflow**, **Manually Triggered Workflows**, or **Ad Hoc Query**, find the ODPS SQL node that you create, and then double-click the ODPS SQL node to go to the configuration tab of the node.

   For more information about functionalities on the DataStudio page, see Features on the DataStudio page. For more information about ad hoc queries, see Create an ad hoc query.
3. Quickly run the code that you select and view the result.



i. Select the code that you want to run.

   In the SQL code section of the configuration tab of the ODPS SQL node, move the pointer over a line of code that you want to run. The system automatically highlights the complete code snippet to which the line of code belongs.

ii. Run the code.

> **Note**
>
> - You can use the quick run feature to quickly run the code only of nodes that are not running. If the code of a node is being executed, the quick run icon  is not displayed on the left side to the code line of the node.
> - Description of the resource group that is used to run the selected code snippet by using the quick run feature:
>   - The resource group that you use to quickly run the current code snippet is the resource group that is used for the most recent execution of the code of the ODPS SQL node. You can run the code of the ODPS SQL node by clicking the quick run icon  , run icon  , or advanced run icon  in the top toolbar of the configuration tab of the node.
>   - The first time you run the code of the ODPS SQL node, you must select a resource group for scheduling based on your business requirements. If existing resource groups do not meet your business requirements, you can create a resource group. For more information, see Create and use an exclusive resource group for scheduling.
>   - If you want to change the resource group that is used to run the code of the ODPS SQL node, click the  icon.
> - The first time you quickly run a code snippet, you must specify values for variables if the code snippet contains variables. The system saves the specified values for subsequent operations. If you want to change the specified values of the variables, click the icon. For more information about how to specify a value for a variable, see Overview of scheduling parameters.

You can use one of the following methods to run the code of the ODPS SQL node:

- Click the quick run icon  on the left side to the code line of the ODPS SQL node.
- Run the code of the ODPS SQL node by using a shortcut key.
  - Windows operating system: Press `Ctrl+Enter`.
  - macOS: Press `Cmd+Enter`.

After the execution of the code of the ODPS SQL node is complete, you can view the execution result. If an unexpected result is returned, modify the code snippet.

9.1.2. Create an ad hoc query

Ad hoc query allows you to test your code in the development environment. This way, you can perform troubleshooting or check whether your code works as expected.

Context

You do not need to commit and deploy ad hoc query nodes or configure scheduling policies for ad hoc query nodes. You can configure scheduling policies only for nodes that are created under **Business process** on the **Data Development** tab of the DataStudio page.

Create a folder

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region where the target workspace resides. Find the target workspace and click **Data Analytics** in the Actions column.
2. On the left-side navigation submenu, click the **Temporary query** icon.

   You can click the icon in the lower-left corner to show or hide the left-side navigation submenu.
3. On the Temporary query tab, move the pointer over the **+Create** icon and select **Folder**.
4. In the **New folder** dialog box, enter a name for the folder in the **Folder name** field and set the **Destination folder** parameter.

   > **Note**
   > - The folder name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
   > - DataWorks supports multi-level folders. You can save a newly created folder under another folder that already exists.
5. Click **New**.

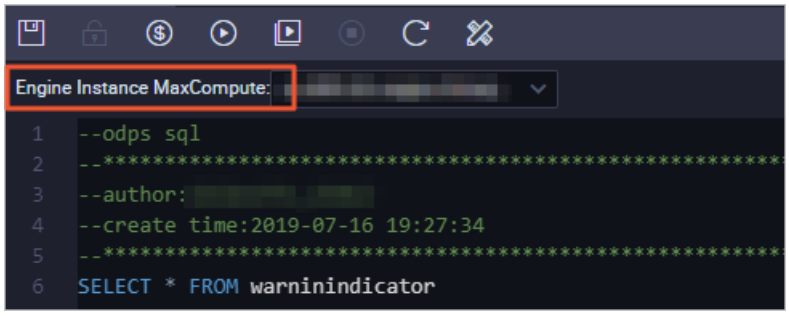
Create an ad hoc query node

On the **Temporary query** tab, you can create a variety of ad hoc query nodes, such as **EMR Hive**, **ODPS SQL**, **EMR Spark SQL**, **EMR Presto**, **EMR Impala**, **Shell**, **AnalyticDB for PostgreSQL**, **AnalyticDB for MySQL**, and **Data Lake Analytics** nodes.

In this example, create an ODPS SQL node.

1. On the **Temporary query** tab, right-click the folder you created and choose **New Node > ODPS SQL**.
2. In the **Create Node** dialog box, set the **Node Name** and **Location** parameters.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).
3. Click **Commit**.
4. On the node configuration tab that appears, enter an SQL query statement.



> Note
> - If the current workspace is bound to multiple MaxCompute compute engines, you must select a desired MaxCompute engine instance before you submit the query statement for execution.
> - The selected MaxCompute engine instance may use a default resource group that is charged in pay-as-you-go mode. In this case, you can click the icon in the toolbar to estimate the cost of this operation. The actual cost is included in your bill.

5. Click the icon in the toolbar to view the query result.

9.1.3. Runtime logs

The Runtime Logs page displays the records of all nodes that have been run in the last three days. You can click a node to view its runtime logs.

> Note The runtime logs are retained for only three days.

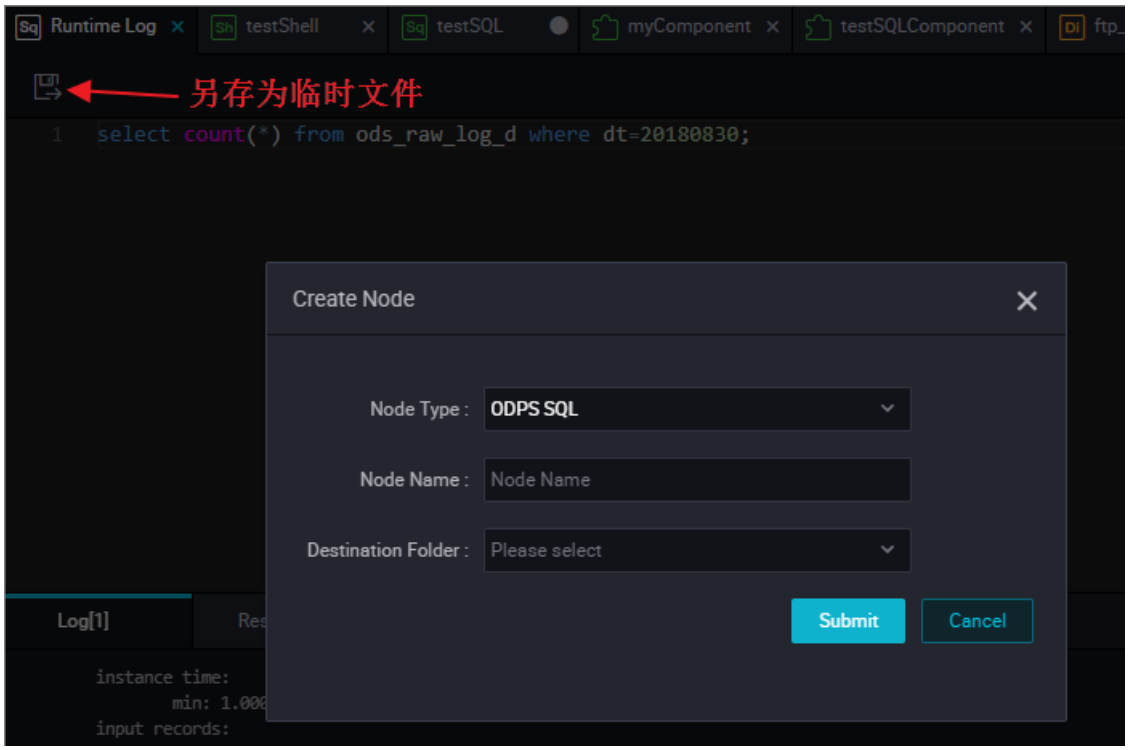
View runtime logs

1. Log on to the DataWorks console. In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
2. In the left-side navigation pane, click **Runtime Logs** to go to the Runtime Logs page. This page displays runtime logs of all node states by default.

   Select a node state from the drop-down list to view the runtime logs of nodes in the specified state.
3. Click a record to view the runtime log on the right.

Save the code as an ad hoc query node

To save the SQL statements in the runtime log, click the **Save as Ad-Hoc Query Node** icon.



In the **Create Node** dialog box, configure **Node Type**, **Location**, and **Node Name**, and click **Submit**.

9.1.4. Use workflow parameters

On the Workflow Parameters tab, you can assign a value to a variable or replace the value of a parameter for all nodes in the current workflow. This topic describes how to configure a workflow parameter. In this example, the value of the ReplaceMe parameter is replaced with ReplaceMe123 for all nodes in a manually triggered workflow.

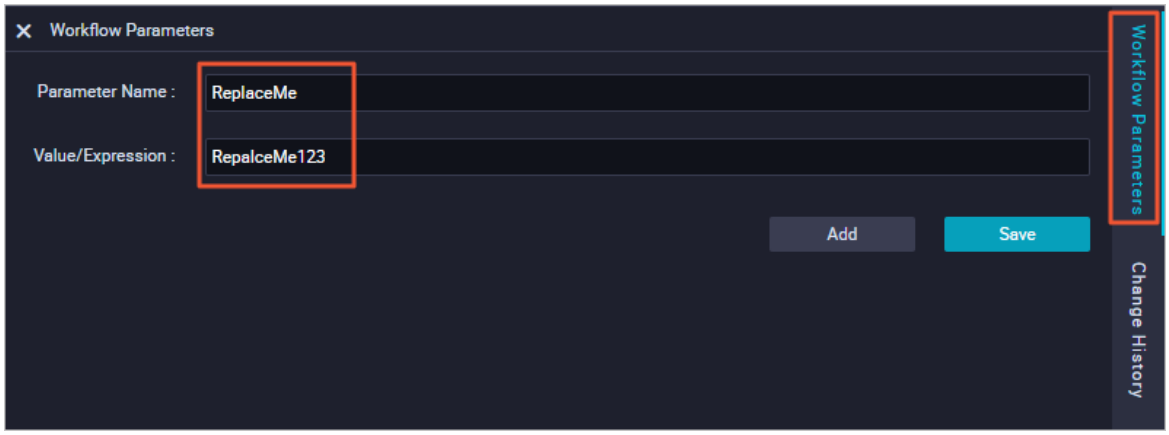
Limits

- In manually triggered workflows, ODPS SQL nodes, Shell nodes, and sync nodes support workflow parameters. The format for specifying a workflow parameter varies based on the node type. For example, a workflow parameter is specified as x=y1.
  - To configure the workflow parameter for an ODPS SQL node, double-click the node and click the **General** tab in the right-side navigation pane. On the **General** tab, enter x=aaa in the Arguments field. When the node is run, the value of the x parameter is replaced with y1. You can use ${x} to reference the workflow parameter in the code.
  - To configure the workflow parameter for a Shell node, double-click the node and click the **General** tab in the right-side navigation pane. On the **General** tab, enter $x in the Arguments field. When the node is run, the value of the x parameter is replaced with y1. You can use $1 to reference the workflow parameter in the code.
  - To configure the workflow parameter for a sync node, double-click the node and click the **General** tab in the right-side navigation pane. On the **General** tab, enter -p"-Dx=aaa" in the Arguments field. When the node is run, the value of the x parameter is replaced with y1. You can use ${x} to reference the workflow parameter in the code.
- In auto triggered workflows, only ODPS SQL nodes support workflow parameters.

- Before you use workflow parameters, you must configure the parameters of each single node in a workflow to ensure that each node is run as expected.
- If the value that is assigned to a workflow parameter is inconsistent with the value that is assigned to a node parameter, the value of the workflow parameter overwrites the value of the node parameter.
- Parameter names and values are case-sensitive.

Configure a workflow parameter

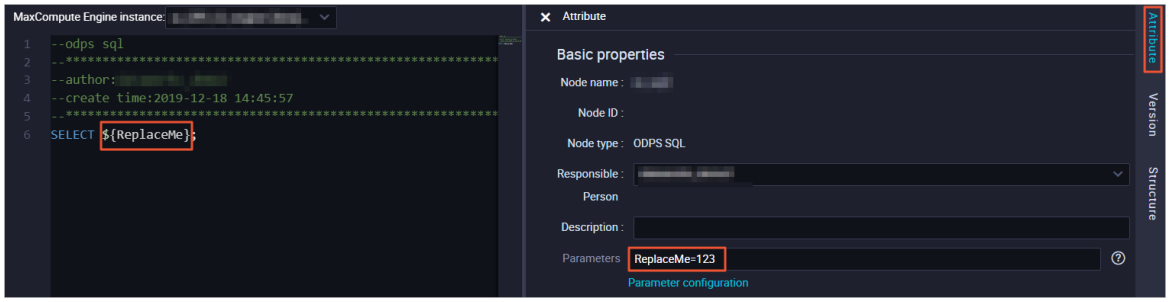
1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the left-side navigation submenu, click the **Manually Triggered Workflows** icon.
3. Double-click a manually triggered workflow. On the workflow configuration tab, click the **Workflow Parameters** tab in the right-side navigation pane.
4. On the **Workflow Parameters** tab, enter ReplaceMe in the **Parameter Name** field and ReplaceMe123 in the **Value/Expression** field.



5. On the workflow configuration tab, click the icon in the top toolbar.

Configure the workflow parameter for an ODPS SQL node

1. On the **DataStudio** page, click the **Manually Triggered Workflows** icon on the left-side navigation submenu.
2. Click the required workflow and choose MaxCompute > **Data Analytics**. Double-click the required ODPS SQL node. The node configuration tab appears.
3. Click the **General** tab in the right-side navigation pane. Enter ReplaceMe=123 in the **Arguments** field.

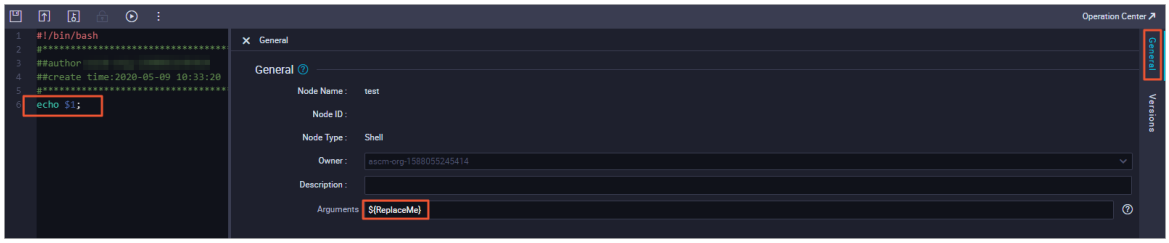


The workflow parameter is specified as ReplaceMe=ReplaceMe123. When the workflow is run, the ReplaceMe parameter is assigned the value ReplaceMe123 for the node.

4. On the workflow configuration tab, click the icon in the top toolbar.

Configure the workflow parameter for a Shell node

1. On the **DataStudio** page, click the **Manually Triggered Workflows** icon on the left-side navigation submenu.
2. Click the required workflow and click **General**. Double-click the required Shell node. The node configuration tab appears.
3. Click the **General** tab in the right-side navigation pane. Enter ${ReplaceMe} in the **Arguments** field.



> **Note** Make sure that you specify the workflow parameter in the correct format for the Shell node.

4. On the workflow configuration tab, click the icon in the top toolbar.

Configure the workflow parameter for a sync node

1. On the **DataStudio** page, click the **Manually Triggered Workflows** icon on the left-side navigation submenu.
2. Click the required workflow and click **Data Integration**. Double-click the required sync node. The node configuration tab appears.
3. Click the **General** tab in the right-side navigation pane. Enter -p"ReplaceMe=abc" in the **Arguments** field.

   The ReplaceMe parameter is assigned the value abc for the sync node. The workflow parameter is specified as ReplaceMe=ReplaceMe123. When the workflow is run, the ReplaceMe parameter is assigned the value ReplaceMe123 for the node, namely pt="ReplaceMe123".

> **Note** Make sure that you specify the workflow parameter in the -p"-DParameter name=Parameter value" format for the sync node.

4. On the workflow configuration tab, click the icon in the top toolbar.

Run the workflow and view the results

On the workflow configuration tab, click the icon in the top toolbar. When an auto triggered workflow is run, the values that are assigned to the parameters of nodes in the workflow are automatically replaced with the values of workflow parameters. When you run a manually triggered workflow on the DataStudio page, you must assign a value to each workflow parameter in the **Enter parameters** dialog box. In this example, assign a value to the ReplaceMe parameter.

- Right-click the ODPS SQL node and select **View Log**. Then, you can view the value that is assigned to the ReplaceMe parameter for the ODPS SQL node.
- Right-click the Shell node and select **View Log**. Then, you can view the value that is assigned to the ReplaceMe parameter for the Shell node.
- Right-click the sync node and select **View Log**. Then, you can view the value that is assigned to the ReplaceMe parameter for the sync node.

If you have not assigned a value to a workflow parameter on the **Workflow Parameters** tab for a manually triggered workflow, you must assign a value to the workflow parameter every time you run the workflow in the production environment.

9.2. Submit and publish tasks

9.2.1. Code review

DataWorks provides the code review feature. If the forcible code review feature is enabled, you must commit each node to a specific reviewer to review the node code. You can deploy the node only after the reviewer approves the node code.

Prerequisites

DataWorks Professional Edition or a more advanced edition is activated. If the administrator enables the forcible code review feature for a workspace in standard mode, you must commit your node for code review. If the forcible code review feature is disabled, you do not need to commit your node for code review. For a workspace in basic mode, you can choose whether to commit your node for code review.

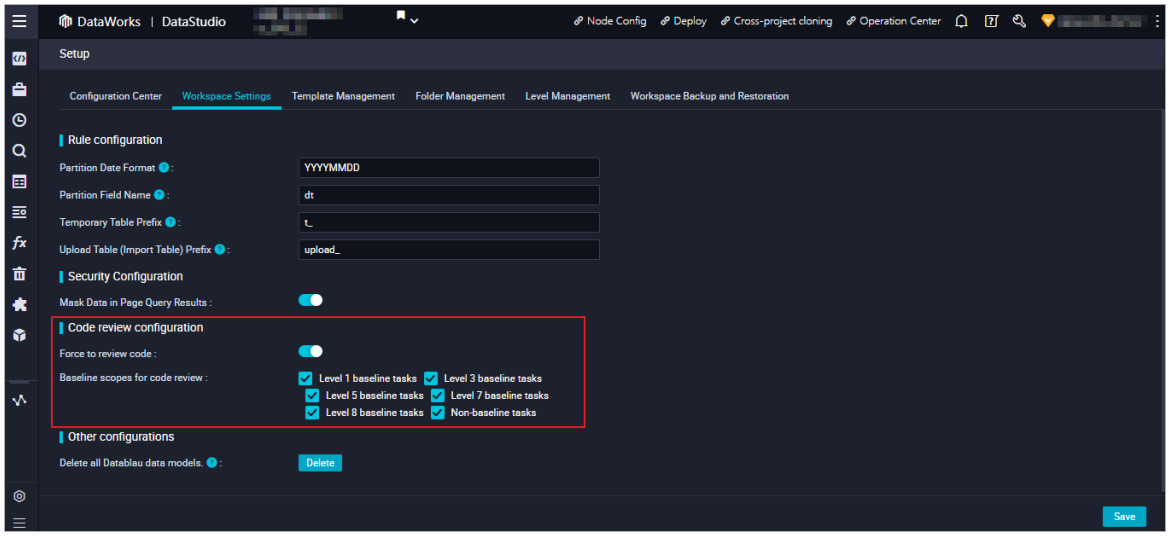
Limits

The code review feature is available only for DataWorks workspaces in standard mode.

Enable the forcible code review feature

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. On the **DataStudio** page, click the icon in the lower-left corner.
3. On the **Setup** page, click the **Workspace Settings** tab.

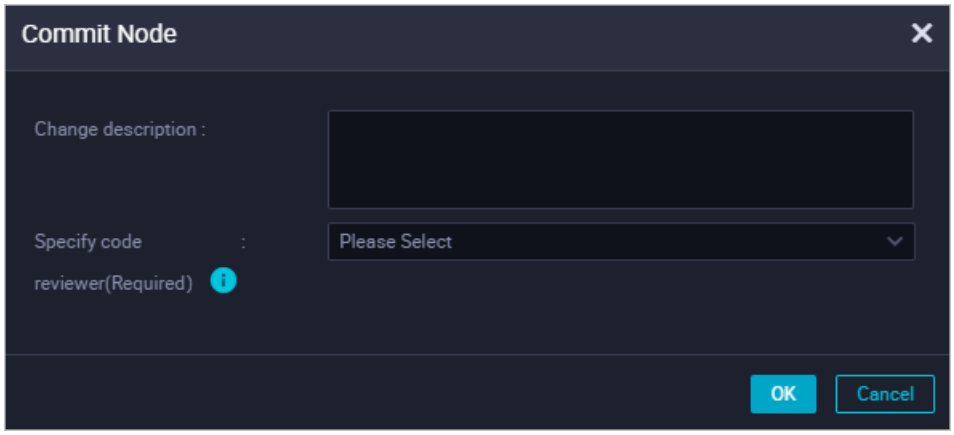
4. In the **Code review configuration** section, turn on **Force to review code** and specify **Baseline scopes for code review**.



You can set the Baseline scopes for code review to **Level 1 baseline tasks**, **Level 3 baseline tasks**, **Level 5 baseline tasks**, **Level 7 baseline tasks**, **Level 8 baseline tasks**, or **Non-baseline tasks**. A high level indicates a high priority. A baseline task has a higher priority than a non-baseline task. For more information about how to configure the priority of a baseline task, see 基线管理. After you specify Baseline scopes for code review, the code of the baseline tasks at the specified level must be reviewed before it is deployed. If multiple tasks are run in parallel, those of a higher priority preferentially preempt resources. If you specify **Baseline scopes for code review** for a task of a higher priority, a reviewer can check for code errors in the task before it is deployed to prevent tasks that contain code errors from occupying resources for a long time.

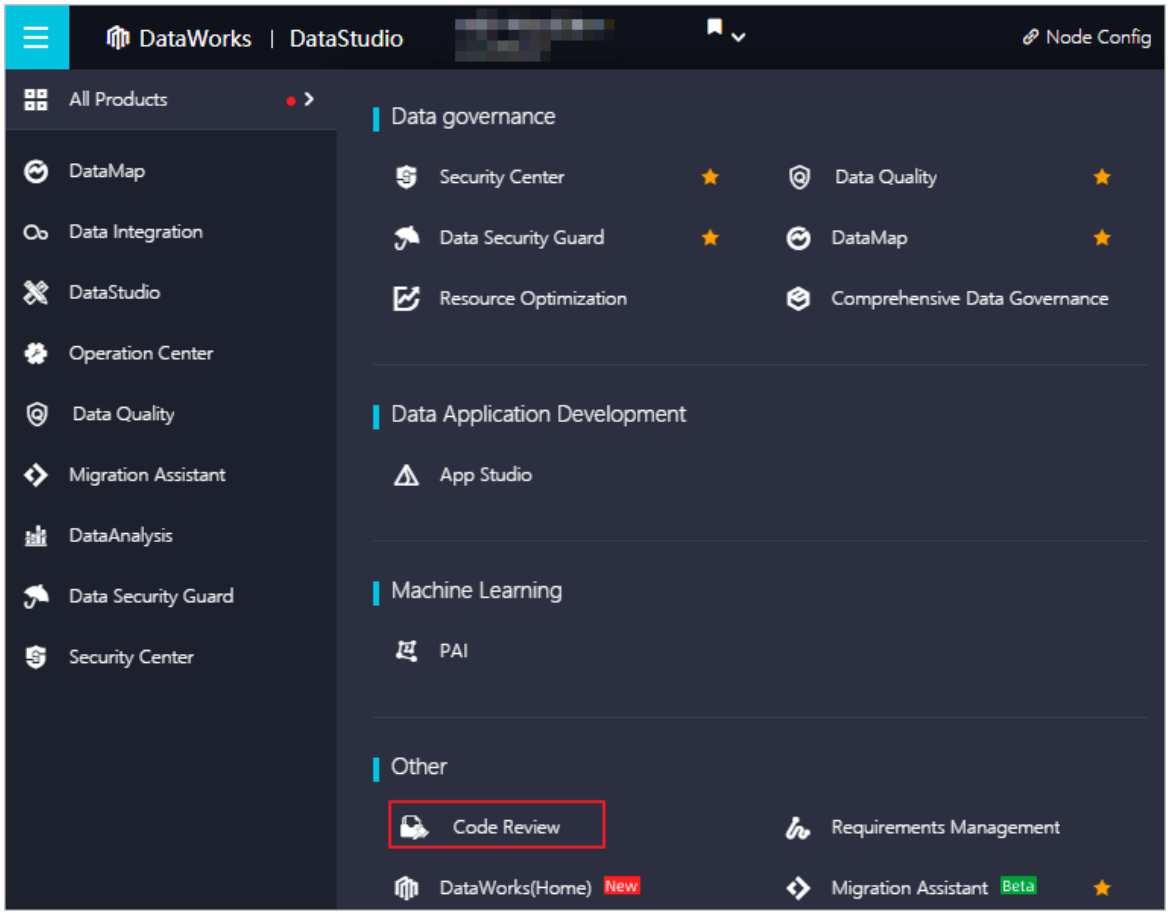
Review code

1. Go to the configuration tab of a node or workflow and click the icon in the top toolbar.
2. In the **Commit Node** dialog box, enter your comments in the **Change description** field and set the **Specify code reviewer(Required)** parameter.



> **Note**
>
> - You cannot commit a combined node such as a do-while or for-each node for code review.
> - After you specify the code reviewer, the system generates a code review ticket.
> - If the forcible code review feature is disabled for the workspace, you do not need to commit your node for code review. Otherwise, you must commit your node for code review. You can deploy the node only after the specified reviewer approves the node code.

3. Click **OK**.
4. After you commit your node for code review, you can click the icon in the upper-left corner and choose **All Products > Code Review** to go to the code review page.



5. On the code review page, click the Review by Me tab, find the node, and then review the code of the node. You can also click the Submitted by Me tab and view the records of the code review requests that you have committed.

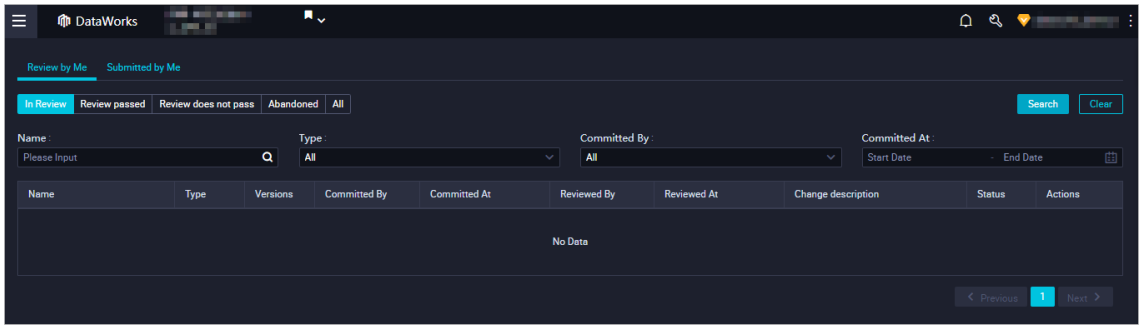
   You can perform the following operations on the code of a specific version as a code reviewer: **Write a comment**, **Pass**, **Dot not pass**, **Abandoned**, and **Reopen**. If the forcible code review feature is enabled, your review result determines whether the node can be deployed. If you perform the **Pass** operation on the code of a specific version, the node with this code version can be deployed. If you perform the **Do not pass** or **Abandoned** operation on the code of a specific version, the node with this code version cannot be deployed. You can compare the code of the

committed version and that of the production version in the Review details section.

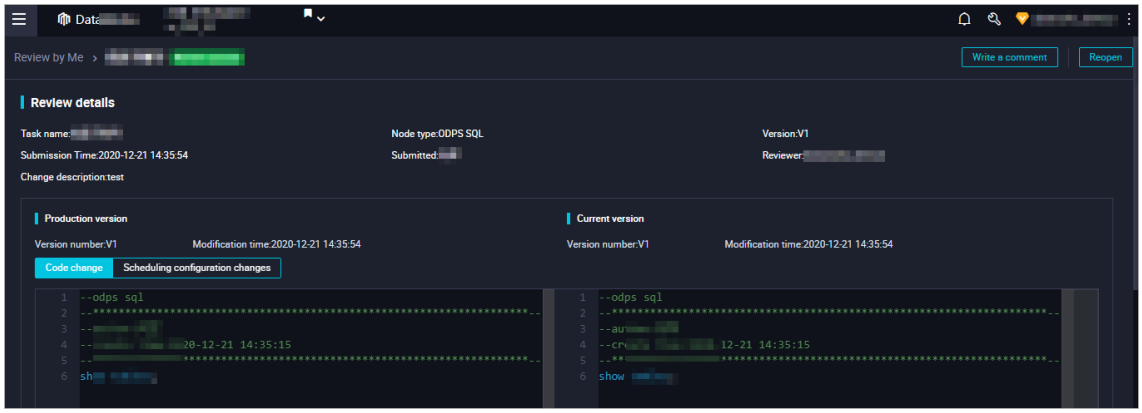
Operation	Description
Write a comment	Adds comments to the committed version.
Pass	Approves the code of the committed version.
Do not pass	Rejects the code of the committed version. **Note** If the forcible code review feature is disabled for the workspace, the node with this code version can still be deployed without code review. If the forcible code review feature is enabled for the workspace, the node with this code version cannot be deployed after you reject the code.
Abandoned	Abandons the code of the committed version.
Reopen	Reviews the code of the committed version again after you abandon it. After you abandon the code of a specific version, you can click **Abandoned** on the **Review by Me** tab, find the code of this version, and then click Reopen in the Actions column to review the code of this version again.

The code review page consists of the **Review by Me** and **Submitted by Me** tabs.

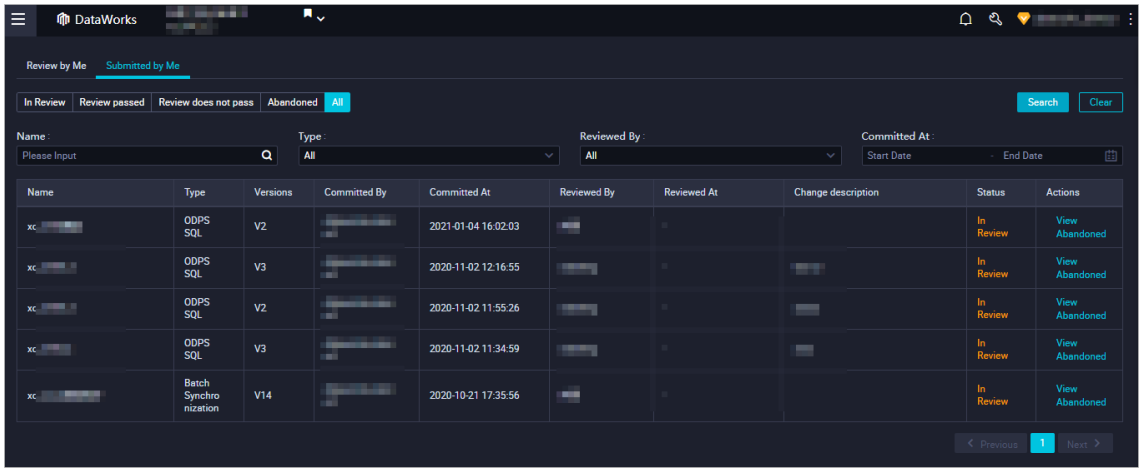
- **Review by Me**: You can view the records of all the review requests submitted to you.



Find a record and click **View** in the Actions column. On the page that appears, you can perform the following operations on the code of a specific version: **Write a comment**, **Pass**, **Do not pass**, **Abandoned**, and **Reopen**.



- **Submitted by Me**: You can view the records of all the review requests that you have submitted.



9.2.2. Publish nodes

9.2.2.1. Deploy nodes

This topic describes how to deploy nodes in a workspace in standard mode and how to use the cross-workspace cloning feature to clone and deploy nodes in a workspace in basic mode.

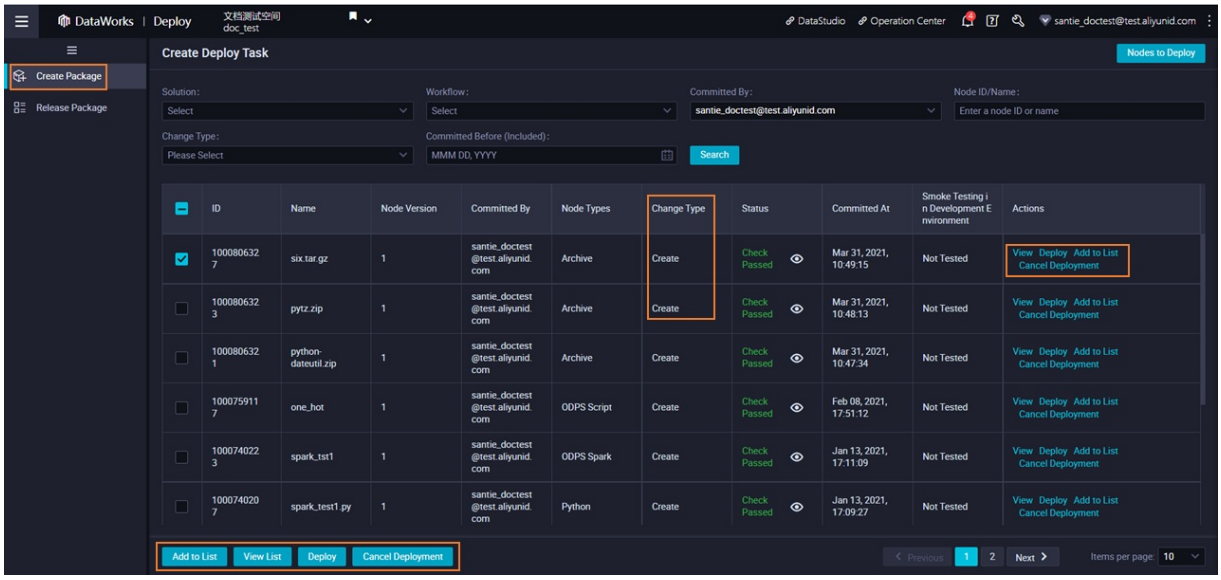
Context

In a rigorous data development process, developers develop and debug code and configure dependencies and scheduling properties for nodes in the development environment. Then, developers commit the nodes to run them in the production environment.

DataWorks workspaces in standard mode provide both the development environment and the production environment within a single workspace. We recommend that you use workspaces in standard mode to develop and produce data. For more information, see Basic mode and standard mode.

In a workspace in standard mode, committed nodes are automatically added to the **Create Deploy Task** page. This page displays created, updated, and deleted nodes, resources, and functions.

After you deploy a node on the **Create Deploy Task** page, a deployment task is generated for the node. You can view the deployment record and status of the node on the **Deploy Tasks** page.



To make the nodes, resources, and functions that you create, update, or delete on the **DataStudio** page take effect in the production environment, you must deploy them to the production environment on the **Create Deploy Task** page. On the **Create Deploy Task** page, you can add one or more nodes to the list of nodes to be deployed and deploy the nodes at a time.

On the **Create Deploy Task** page, you can modify the number of items that can be displayed on each page.

Find the node that you want to view and click **View** in the Actions column. You can view the changes that are made in the current version to the code and scheduling properties of the node. The following table describes the parameters related to the configurations of scheduling properties.

Parameter	Description
appId	The ID of the DataWorks workspace to which the node belongs. You can go to the Workspace Management page to view the ID. For more information, see Configure a workspace.
createUser	The ID of the user that created the node.
createTime	The time when the node was created.
lastModifyUser	The ID of the user that last modified the node.
lastModifyTime	The time when the node was last modified.
owner	The ID of the owner of the node. You can view the owner ID in the **General** section of the **Properties** panel. For more information, see Configure basic properties.

Parameter	Description
startRightNow	The mode in which auto triggered node instances are generated for the node. Valid values: • **0**: Auto triggered node instances are generated on the next day after the node is deployed. • **1**: Auto triggered node instances are immediately generated after the node is deployed. For more information, see Configure immediate instance generation for a node.
taskRerunTime	The number of times for which the node is rerun.
taskRerunInterval	The interval between two consecutive automatic reruns. Unit: milliseconds.
reRunAble	Indicates whether the node can be rerun. Valid values: • **0**: The node can be rerun only after it fails to run. • **1**: The node can be rerun regardless of whether it is run as expected or fails to run. • **2**: The node cannot be rerun regardless of whether it is run as expected or fails to run.
startEffectDate	The start date and time of the period during which scheduling takes effect.
endEffectDate	The end date and time of the period during which scheduling takes effect.
cycleType	The scheduling type of the node. Valid values: • **0**: The node is scheduled by day, week, month, or year. • **1**, **2**, or **3**: The node is scheduled by minute or hour.
cronExpress	The CRON expression used for configuring periodic scheduling.

extConfig	The additional configurations of the node. The value must be in the JSON format and contains the settings of the following fields:
	• **ignoreBranchConditionSkip**: specifies whether the dry-run property in the previous cycle is passed to the current cycle of the node.
	  ◦ **true**: The dry-run property in the previous cycle is passed to the current cycle of the node.
	  ◦ **false**: The dry-run property in the previous cycle is not passed to the current cycle of the node.
	  For more information, see Pass the dry-run attribute of an ancestor node.
	• **alisaTaskKillTimeout**: the timeout period. Unit: hours.

resgroupId	The ID of the resource group for scheduling used to run the node. For more information, see Configure a resource group.

Parameter	Description
isAutoParse	Specifies whether to enable the automatic parsing feature for the node. Valid values: • 1: enables the automatic parsing feature for the node. • 0: disables the automatic parsing feature for the node. For more information, see Configure same-cycle scheduling dependencies.

input
inputList
output
outputList
	The input and output configurations of the node. The parameter values contain the settings of the following fields:
	• str: the input or output value.
	• refTableName: the output table.
	• parseType: the way in which the scheduling dependencies of the node are configured. Valid values:
	  ◦ 0: The scheduling dependencies are automatically configured for the node based on the automatic parsing feature.
	  ◦ 1: The scheduling dependencies are manually configured for the node.
	  ◦ 2: The scheduling dependencies for the node are automatically configured based on the connections between nodes.
	For more information, see Logic of same-cycle scheduling dependencies.

dependentTypeList	The previous-cycle scheduling dependencies of the node. Valid values: • 0: no specified node. • 1: one or more specified nodes. • 2: child nodes. • 3: the current node. For more information, see Configure previous-cycle scheduling dependencies.
dependentDataNode	The IDs of the one or more nodes that are specified as the previous-cycle scheduling dependencies of the node. This parameter is valid only if the dependentTypeList parameter is set to 1.
inputContextList outputContextList	The context-based input and output parameters of the node. For more information, see Configure input and output parameters.
tags tagList fileId isStop dependentType	The reserved parameters.

For more information about the time properties, see Configure time properties.

The time when instances are generated varies based on the instance generation mode.

- Nodes for which the Instance Generation Mode parameter is set to Next Day: If you update and deploy an auto triggered node before 23:30, instances are generated for the updated node the next day.
- Nodes for which the Instance Generation Mode parameter is set to Immediately After Deployment: If you create and deploy a node, instances whose scheduled time is 10 minutes later than the time when the node is deployed are generated as expected. If you update and deploy a node, instances whose scheduled time is 10 minutes later than the time when the node is deployed are generated again based on the latest scheduling configuration. These instances replace those that are generated before the update. For more information, see Configure immediate instance generation for a node.
- If you create or update a node and deploy the node after 23:30, instances are generated for the new or updated node on the third day.
- If you deploy a node after 23:30 and the Instance Generation Mode parameter is set to Immediately After Deployment, instances are not immediately generated for the node.

Deploy nodes in a workspace in standard mode

Each DataWorks workspace in standard mode is associated with two MaxCompute projects, one as the development environment and the other as the production environment. You can directly commit and deploy nodes from the development environment to the production environment.

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. Find the workspace that you want to use and click **Data Development** in the Actions column.
2. Commit the node.

   In a workspace in standard mode, only members that are assigned the developer role are allowed to commit nodes.

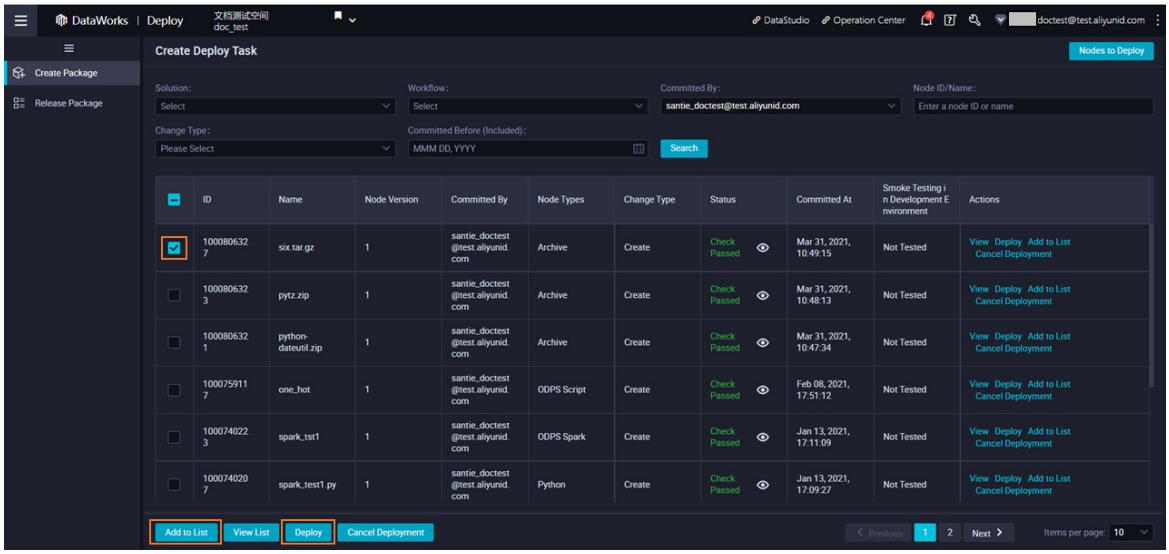
   i. Double-click a configured workflow. On the tab that appears, click the icon in the top toolbar.
   ii. In the **Commit** dialog box, select the nodes to be committed, set the **Change description** parameter, and then select **Ignore I/O Inconsistency Alerts**.

   > **Note** If all nodes of a workflow are committed and you modify only the workflow or node properties, you can enter the description and commit the workflow, without the need to select the nodes. The changes are automatically committed.
   >
   > If a node has been committed and the node code remains unchanged, you cannot select the node again.

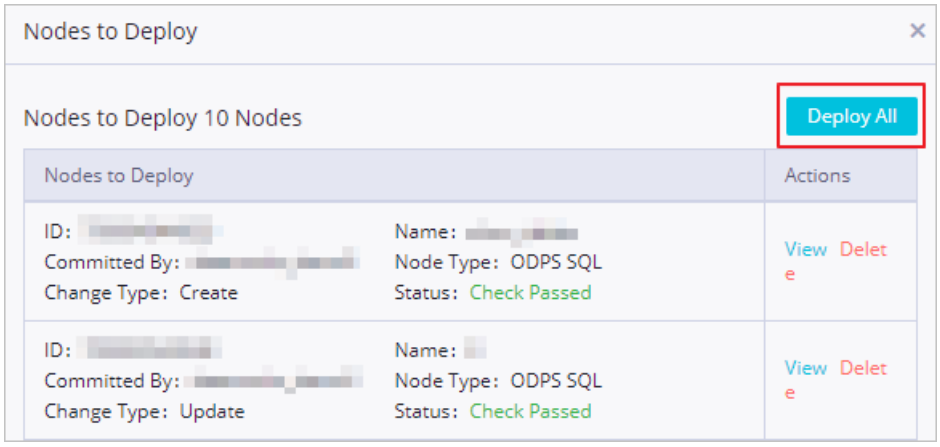
   iii. Click **Commit**.
3. After you commit the nodes, click **Deploy** in the upper-right corner.

   In a workspace in standard mode, only members that are assigned the administration expert, deployment expert, or administrator role can deploy nodes.
4. On the **Create Deploy Task** page, select the nodes to be deployed at a time and click **Add to List**.

You can filter and search for nodes by setting parameters such as **Committed By**, **Node ID/Name**, and **Change Type**. You can click **Deploy** to immediately deploy the selected nodes to the production environment.



5. Click **View List**. Check whether the node information in the list is correct and click **Create Task for All**. All nodes in the list are deployed to the production environment.



> **Note** Workspaces in basic mode do not allow you to directly perform operations on table data in the production environment. Workspaces in standard mode ensure a stable, secure, and reliable production environment. Therefore, we recommend that you deploy and run nodes in a workspace in standard mode.

Deploy nodes in a workspace in basic mode

If you want to isolate the development environment from the production environment when you use workspaces in basic mode, create two workspaces, one for development and the other for production. You can clone nodes from the development workspace to the production workspace.

For example, two workspaces in basic mode are created, one for development and the other for production. You can use the **cross-workspace cloning** feature to clone nodes from Workspace A to Workspace B, and then commit the cloned nodes to the scheduling engine in Workspace B.

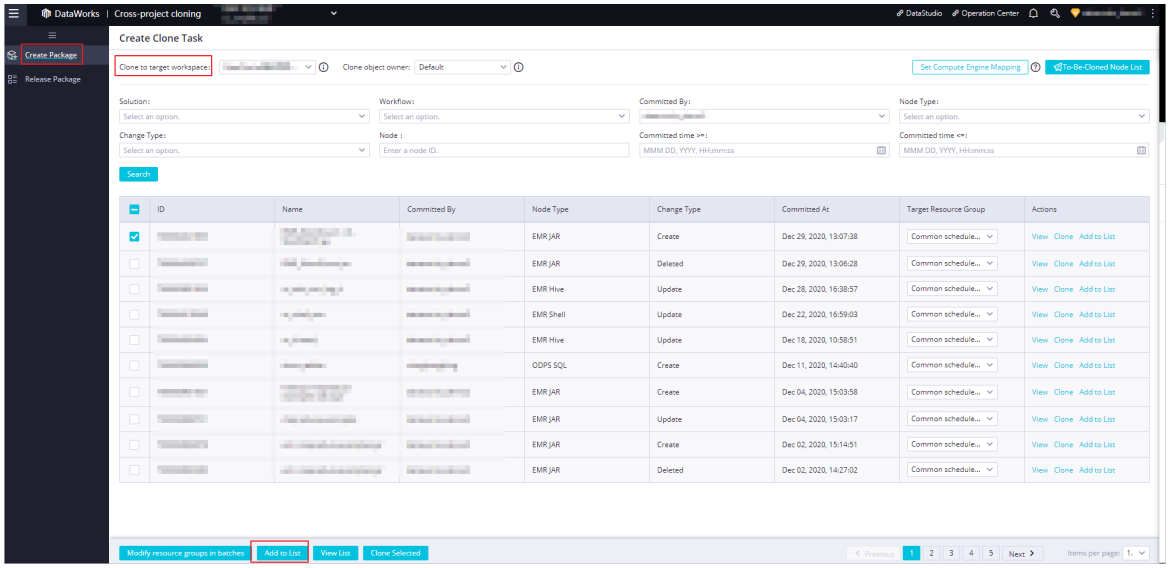
> **Note**
>
> - Permission requirement: Only workspace administrators and RAM users who are assigned the **administration expert** role can clone nodes. The administration expert role has permissions to create clone tasks and deploy cloned nodes.
> - Supported workspace type: You can clone nodes only from workspaces in basic mode to other workspaces.
> - Prerequisites: The source workspace in basic mode and the destination workspace in standard mode are created.

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. Find the workspace that you want to use and click **Data Development** in the Actions column.
2. Commit the node.
   i. Double-click a configured workflow. On the tab that appears, click the icon in the top toolbar.
   ii. In the **Commit** dialog box, select the nodes to be committed, set the **Change description** parameter, and then select **Ignore I/O Inconsistency Alerts**.
   iii. Click **Commit**.
3. Click **Cross-project cloning** in the upper-right corner.
4. On the **Create Clone Task** page, select the nodes to be cloned and set the **Target Workspace** parameter.
5. Click **Set Compute Engine Mapping**. Configure the mapping between the compute engines of the current workspace and the destination workspace.

   If the destination workspace has multiple compute engines, you must configure the mapping between the compute engines of the current workspace and the destination workspace before you clone nodes. If no mapping is configured, the nodes are cloned to the default compute engine of the destination workspace.

   > **Note**
   >
   > - If the type of the compute engine to which the nodes to be cloned does not exist in the destination workspace, a message appears in the **Set Compute Engine Mapping** dialog box. You can choose to skip these nodes that cannot be cloned. Otherwise, an error is reported during the cloning process.
   > - The **Set Compute Engine Mapping** button is displayed only if an engine type in the source or destination workspace has more than two engine instances.

6. Click **Add to List**. The selected nodes are added to the list of nodes to be cloned.



7. Click **To-Be-Cloned Node List** in the upper-right corner. Click **Clone All**.
8. After the engine mapping is prechecked, confirm the information and click **Clone**.
9. After the nodes are cloned, go to the destination workspace and view the cloned nodes. In most cases, the overall directory structure of the workflow is cloned.

9.2.2.2. Delete a node

In some scenarios, you need to delete a node from the development or production environment.

Delete a node from the development environment

1. Log on to the DataWorks console. In the left-side navigation pane, click Workspaces. On the Workspaces page, find the target workspace and click **Data Analytics** in the Actions column.
2. Search by node type and keyword for the node to be deleted.
3. Right-click the node and select **Delete**. The node is deleted from the development environment.

Delete a node from the production environment

To delete a node from the production environment, you need to delete the node from the development environment and create a deployment task to deploy the node.

> **Note** Before deleting a node from the production environment, you need to remove its dependencies with child nodes. Otherwise, a message appears, indicating that the node to be deleted has child nodes and cannot be deleted.
>
> To remove the dependencies, follow these steps:
>
> 1. Find each child node of the target node. You can view the dependencies of the target node in the workflow.
> 2. On the configuration tab of a child node, click the Properties tab in the right-side navigation pane, and change the parent node. Alternatively, delete the child node.
>
> If a message appears indicating that the child node has next-level child nodes, repeat these steps to remove the dependencies.

1. Delete the node from the development environment.

   For more information, see the preceding section.

2. Create a deployment task for the deleted node.

   > Note Only administrators and administration experts have the permission to create a deployment task. If you are not an administrator or administration expert, we recommend that you seek help from an administration expert.

   i. After deleting the node from the production environment, click **Deploy** in the upper-right corner.

   ii. On the **Create Deploy Task** page, select the node.

   iii. Click **Deploy Selected**.

   You can also click **Add to List** to add the node to **To-Be-Deployed Node List** and deploy the node in the Nodes to Deploy pane.

3. Deploy the node.

   In the **Create Deploy Task** dialog box that appears, click **Deploy** to deploy the node.

9.2.3. Cross-workspace cloning

9.2.3.1. Overview

For workspaces in basic mode under the same Alibaba Cloud account, you can use the cross-workspace cloning feature to isolate the development environment from the production environment. You can also use this feature to clone and migrate nodes, such as computing or sync nodes, across workspaces. This topic describes how to process the dependencies between nodes during cross-workspace cloning.

If you clone nodes across workspaces by using the **cross-workspace cloning** feature, DataWorks automatically modifies the output names in the destination workspace to distinguish nodes in different workspaces under the same Alibaba Cloud account. This allows you to smoothly clone or maintain node dependencies.

> Note
> - **Cross-workspace cloning** cannot be used across regions.
> - You cannot clone nodes in a workflow of earlier versions to a workspace in the latest version. You must clone the nodes to a folder under **Business Flow** and then clone the entire workflow to the destination workspace.

You can set the owner of cloned nodes to **Default** or **Clone Task Creator**.

- If the owner of the nodes to be cloned is the workspace administrator, you can set the owner of the cloned nodes to **Default** or **Clone Task Creator**.

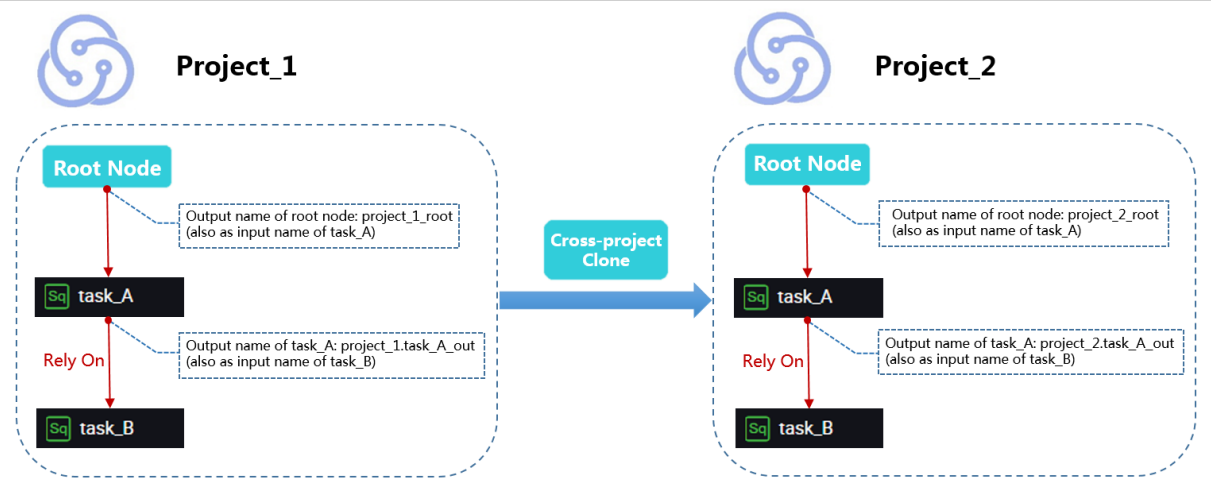
  After the nodes are cloned, their owner is preferentially set to the original owner. If the original owner is not added to the destination workspace, the clone task creator becomes the owner.

- If the owner of the nodes to be cloned is the clone task creator, you can set the owner of the cloned nodes to **Default** or **Clone Task Creator**.

After the nodes are cloned, their owner is preferentially set to the original owner. If the original owner is not added to the destination workspace, the system asks whether to change the owner. If you agree to change the owner, the clone task creator becomes the owner of the cloned nodes and the clone task succeeds. If you do not agree to change the owner, the clone task is canceled.

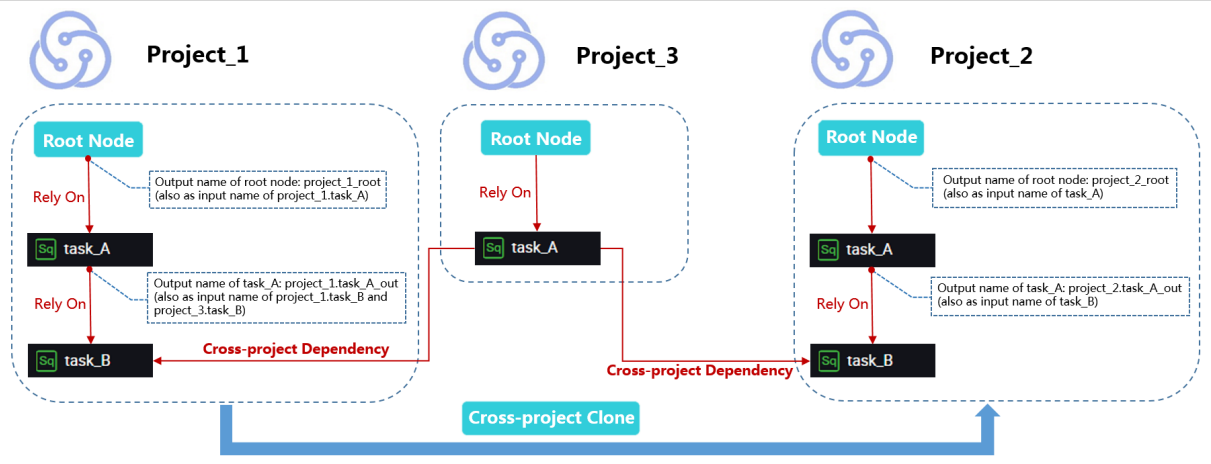
Complete workflow cloning

Assume that the output name of the task_A node in the project_1 workspace is project_1.task_A_out. If you clone the task_A node to the project_2 workspace, the node output name changes to project_2.task_A_out.



Cross-workspace dependency cloning

Assume that the task_ B node in the project_1 workspace depends on the task_A node in the project_3 workspace. If you clone the task_B node in the project_1 workspace to the project_2 workspace, the dependency between the task_A and task_B nodes is also cloned. The task_B node in the project_2 workspace also depends on the task_A node in the project_3 workspace.



9.2.3.2. Clone nodes across workspaces

This topic describes how to clone nodes across workspaces.

Scenarios

You can clone nodes across workspaces in the following scenarios:

- Clone nodes from a workspace in basic mode to another workspace in basic mode.
- Clone nodes from a workspace in basic mode to another workspace in standard mode.

> Note
> - After you clone a node, the folder and workflow to which the node belongs are also cloned to the destination workspace.
> - Before you clone a node, make sure that its ancestor nodes are cloned to the destination workspace.

Procedure

1. Go to the **DataStudio** page and double-click the target workflow on the Data Analytics tab.
2. Click **Cross-Project Cloning** in the upper-right corner. The Create Clone Task page appears.
3. Set **Target Workspace** and select the nodes to be cloned.
4. Click **Set Compute Engine Mapping**. In the dialog box that appears, set the mapping between the compute engines of the current workspace and the destination workspace.

   If the destination workspace has multiple compute engines, you must set the mapping between compute engines of the current workspace and the destination workspace before cloning nodes. If no mapping is set, the nodes are cloned to the default compute engine of the destination workspace.

   > Note
   > - If the engine type of some nodes to be cloned does not exist in the destination workspace, a message appears in the **Compute Engine Mapping** dialog box. You can select **Skip Nodes Without Target Engine Instance** to ignore these nodes. Otherwise, an error will be reported during cloning.
   > - The **Set Compute Engine Mapping** button only appears when an engine type in the source or destination workspace has more than two engine instances.

5. Click **Add to List**. The selected nodes are added to **To-Be-Cloned Node List**.
6. Click **To-Be-Cloned Node List** in the upper-right corner. On the Nodes to Deploy pane that appears, click **Clone All**.
7. In the **Compute Engine Mapping** dialog box that appears, confirm the engine mapping and click **OK**.
8. After the nodes are cloned, view the cloned nodes in the destination workspace. Generally, the overall folder structure of the workflow is cloned.

10.GUI elements

10.1. Code search

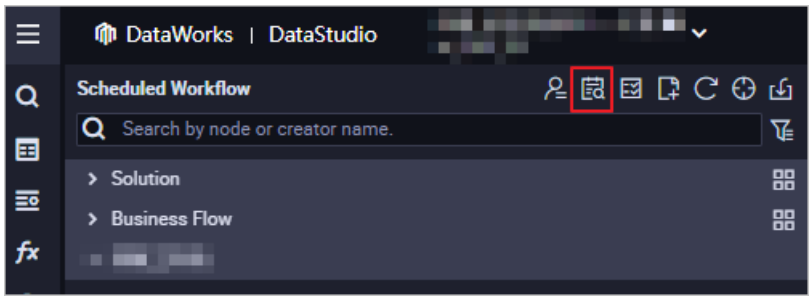
DataWorks provides the code search feature that allows you to query code segments in nodes by keyword. The search results show the details of each segment and the nodes that contain these segments. You can use this feature to trace the node that leads to changes in a destination table. This topic describes how to use the code search feature on the DataStudio page.

Limits

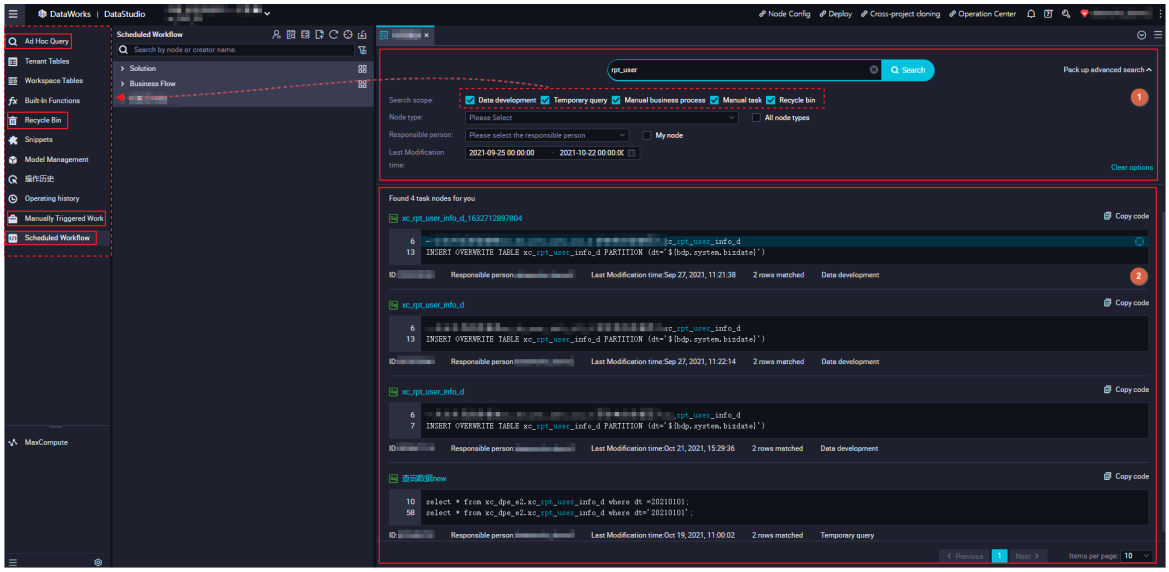
- You can use the code search feature only in DataWorks editions that are more advanced than DataWorks Basic Edition.
- The code search feature allows you to query nodes that you created in the directories of **Scheduled Workflow**, **Ad Hoc Query**, **Manually Triggered Workflows**, **Manually Triggered Nodes**, and **Recycle Bin** tabs. For more information about features of each tab, see Features on the DataStudio page.
- The **Manually Triggered Nodes** tab is only available for DataWorks V1.0.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. In the upper part of the Scheduled Workflow pane, click the icon. The Search Code tab appears on the right.



3. Configure search conditions and view search results.



i. Enter keywords and configure search conditions.

a. In Area 1, enter keywords in the search box, such as a code segment or the name of a table. The more specific the keywords, the more relevant the search results.

Example: Enter `alter table test, rpt_user`. The alter table test keyword indicates an operation and the `rpt_user` keyword indicates the name of the table.

b. Click **Expand Advanced Search** next to the search box and set the **Search scope**, **Node type**, **Responsible person**, and **Last Modification time** parameters to narrow down the query range.

> **Note**
>
> - This helps you find the required node in a more accurate and quick manner. If you do not configure the advanced search conditions, all nodes in the current workspace are searched by default.
> - The code search feature allows you to query nodes that you created in the directories of **Scheduled Workflow**, **Ad Hoc Query**, **Manually Triggered Workflows**, **Manually Triggered Nodes**, and **Recycle Bin** tabs. For more information about features of each tab, see Features on the DataStudio page.
> - The **Manually Triggered Nodes** tab is only available for DataWorks V1.0.

ii. Click **Search**.

iii. View the search results.

In Area 2, you can view all the code segments that match the keywords and the nodes that contain the segments and match the search conditions. This helps you find the required node that causes the changes in the destination table. You can also perform the following operations:

- Click the name of the node to view the details of the node.
- Click **Copy code** to copy the segment for further use.

10.2. Lineage

The Lineage tab displays the relationships between a node and other nodes. You can view the node dependencies and the lineage parsed from the code of the node.

Dependencies

You can check the node dependencies presented based on the current configuration. If the node dependencies fail to meet your expectations, you can reconfigure the node dependencies on the Properties tab.

Lineage

The lineage is parsed based on the code of the current node. For example, an ODPS SQL node contains the following SQL statements:

```
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;
```

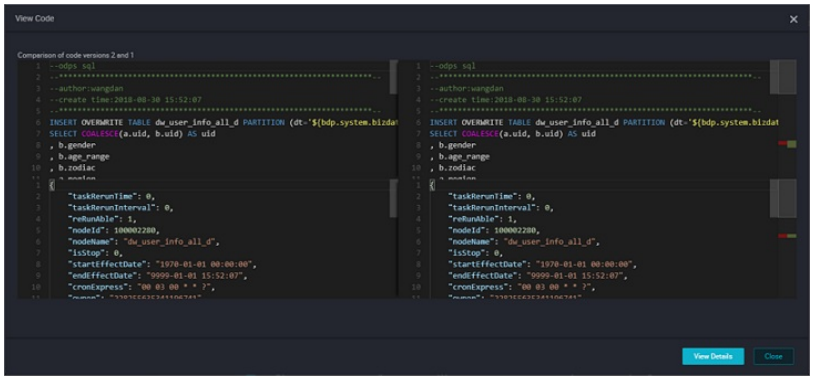
The following figure shows the lineage parsed from the preceding SQL statements. The results queried from the ods_log_info_d and ods_user_info_d tables are joined and then inserted to the dw_user_info_all_d table.

10.3. Versions

The Versions tab displays the version information that is generated each time a node is committed and deployed. You can view the historical versions and information about each version. This information includes the user that committed the node, time when the node was committed, change type, status, and description.

> **Note** Only a committed node has version information. Each time a node is committed, a version is generated and added to the **Versions** tab.

Parameter or button	Description
File ID	The ID of the node.
Version	The version of the node. A version is generated each time the node is committed and deployed. V1 indicates version 1 and V2 indicates version 2. The version number increases by 1 each time an additional version is generated.
Committed By	The user that committed the node.
Committed At	The time when the node was committed. By default, the time when the node was last committed is displayed.
Change Type	The operation that is performed on the node. The first time that the node is committed and deployed, the value of this parameter is Create. However, if the node is modified, committed, and then deployed, the value of this parameter is Change.

Parameter or button	Description
Status	The status of the node. Valid values: • **Yes**: The node is committed to the development environment but the related deployment task has not been created. The node has not been deployed in the production environment. • **Not Deployed**: The node is committed to the development environment and the deployment task is created. The node is pending for deployment. • **Deployed**: The node is committed to the development environment and deployed in the production environment.
Description	The change description of the node when it is committed. This description helps other users find the relevant version when they manage the node.
Actions	The actions that you can perform on the version. Two actions are available: **View Code** and **Roll Back**. • **View Code**: Click the button to view the code of the current version. • **Roll Back**: Click the button to roll back the node from the current version to the required version. After you roll back a node, you must commit and deploy it again.
Compare	Click the button to compare the code and properties between two selected versions.  **Note** You can compare only two versions. If you select only one version or more than two versions, the comparison is not supported.

10.4. View the code structure

DataWorks allows you to view the SQL code structure parsed from the code of a node. The code structure helps you troubleshoot issues and view and modify the code.

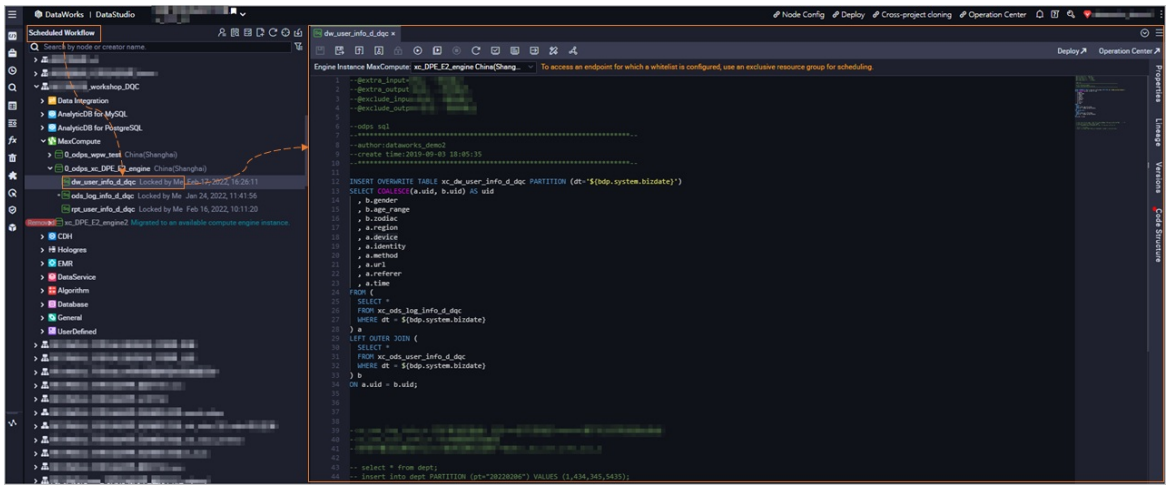
View the code structure of a node

1. Go to the DataStudio page.
   i. Log on to the DataWorks console. In the left-side navigation pane, click **Workspaces**.

ii. Find the workspace that you want to manage and click **Data Development** in the Actions column to go to DataStudio.

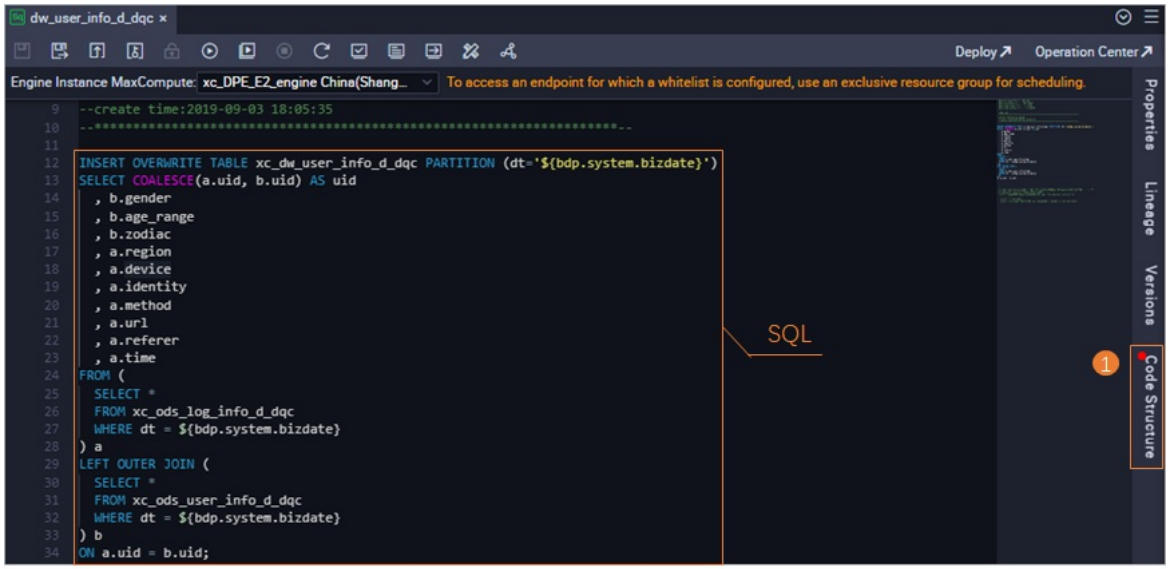
2. Go to the configuration tab of a node.

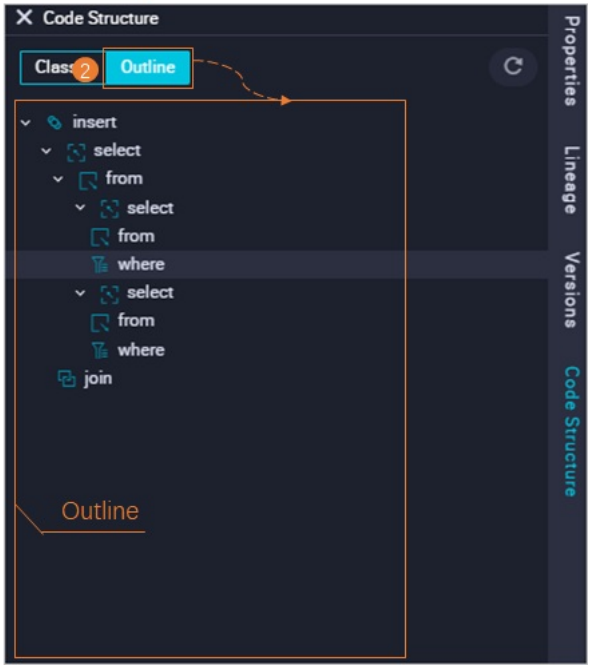
   You can search for a node in the **Scheduled Workflow** or **Manually Triggered Workflows** pane. After that, double-click the node to go to the configuration tab of the node.



3. View the code structure of the node.

   In the right-side navigation pane, click **Code Structure** to view the code structure of the node in classic mode or outline mode.

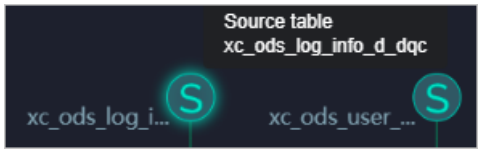




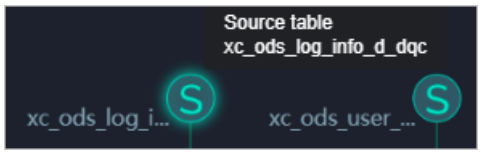
- Classic mode: The SQL code is run based on the organized SQL operators to obtain the final result. The classic mode displays the SQL operators involved in the structure and the associations between the operators.

  You can view the type of an SQL operator by moving the pointer over this operator. The following list describes the common SQL operators used in the code structure.

  - **Source table**: the table to be queried.

- **Filter**: the condition for filtering the partitions in the table to be queried.



- **Intermediate table (query view)**: the temporary table that stores the query results.

  In this example, three intermediate tables are involved. Two intermediate tables displayed in the first section are used to store the query results from the corresponding source tables. The intermediate table displayed in the second section is used to store the joined query results. This intermediate table can be stored for three days and is automatically deleted after three days.

- **Join**: the operation for joining the query results.

  - Outline mode: The outline mode displays the directory of the main SQL statements of the code.

  You can view a specified code block by clicking the corresponding SQL operator in classic mode or the statement in outline mode. After that, you can modify the code as needed.

10.5. Change the resource groups for scheduling for one or more nodes

The resource group orchestration feature of DataWorks allows you to change the resource groups for scheduling for nodes in the specified workflow. If you have multiple resource groups for scheduling in your workspace, you can change the resource groups for scheduling for nodes based on your business requirements. This helps you promote resource efficiency. This topic describes how to use the resource group orchestration feature of DataWorks.

Prerequisites

- A workflow is created. The resource group orchestration feature allows you to change the resource groups for scheduling for nodes in a specified workflow. For more information about how to create a workflow, see Create a workflow.
- One or more resource groups for scheduling are created for your workspace. This way, you can use the resource group orchestration feature to allocate resource groups for scheduling in the current workspace.

  If you activate DataWorks, the system automatically creates the shared resource group for scheduling for you to run nodes. If the shared resource group for scheduling cannot meet your requirements, you can purchase exclusive resource groups for scheduling as needed. For more information about exclusive resource groups for scheduling, see Overview.

Context

The resource group orchestration feature is used to change the resource groups for scheduling used by multiple nodes in a specified workflow during data development. The maximum number of concurrent nodes that can be run by a resource group for scheduling varies based on the specifications of the resource group. You can specify the nodes to run on the resource group for scheduling whose specifications meet the needs of the nodes. For more information about the specifications of exclusive resource groups for scheduling and the maximum number of concurrent nodes supported by each specification, see Billing of exclusive resource groups for scheduling (subscription).

> **Note** For information about how to change the resource groups for scheduling for multiple nodes in the production environment, see View auto triggered nodes.

Limits

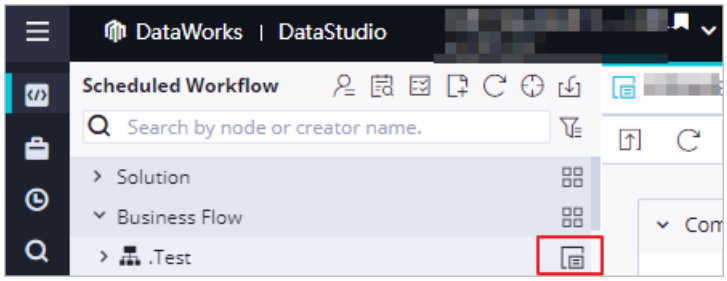
- DataWorks allows you to change the resource groups for scheduling only for auto triggered nodes.
- Zero load nodes do not occupy scheduling resources. Therefore, you cannot change the resource groups for scheduling for zero load nodes.

Procedure

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. In the top navigation bar, select the region in which the workspace that you want to manage resides. Find the workspace and click **DataStudio** in the Actions column.
2. Go to the Resource Group Orchestration tab.

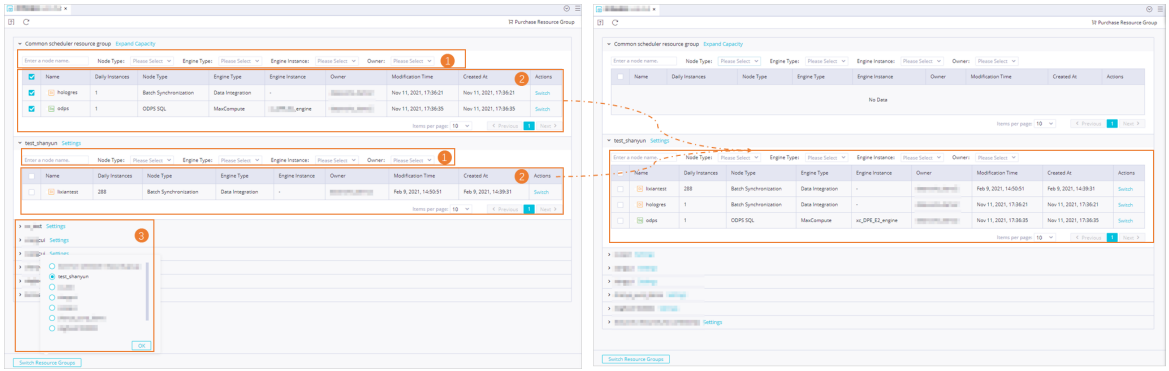
   On the DataStudio page, move the pointer over the required workflow and click the icon next to the workflow name to go to the Resource Group Orchestration tab.

   > **Note** Before you go to the Resource Group Orchestration tab, you must find the workflow in which you want to change the resource groups for scheduling for the nodes.

3. Change the resource groups for scheduling for multiple nodes.

> **Note**
>
> - You can specify the same resource group for scheduling for the nodes that run on different resource groups for scheduling at a time.
> - If you want to specify different resource groups for scheduling for the nodes that run on multiple resource groups, you must specify a resource group for scheduling for part of the nodes at each time.



i. (Optional)Find the required nodes by configuring the filter conditions, such as the **Name**, **Node Type**, and **Engine Type** parameters.

ii. Check the nodes in the search results and select the nodes for which you want to change the resource groups for scheduling.

iii. Click **Switch Resource Groups** and select a resource group for scheduling from the list.

iv. Click **OK**.

You can perform the following operations for the resource groups for scheduling:

- **Scale out the resource groups for scheduling**: You can scale out only the shared resource groups for scheduling. Click **Expand Capacity** next to the **Common scheduler resource group**. Then, the Public Resource Groups tab appears. You can click Buy Now in the Public Resource Group Usage section to purchase resource plans of shared resource groups for scheduling.
- **Modify specifications**: You can modify specifications for only exclusive resource groups for scheduling. Click **Settings** next to the name of an **exclusive resource group for scheduling**. Then, the Exclusive Resource Groups tab appears. You can view the details of the exclusive resource group for scheduling and modify specifications as needed.
- **Purchase resource groups for scheduling**: If no resource groups are available in the current workspace, you can click **Purchase Resource Group** in the upper-right corner to purchase resource groups for scheduling.
- **Change the resource group for scheduling configured for a single node**: Click **Switch** in the **Actions** column that corresponds to the node to change the resource group for scheduling for the node.

4. Click the icon to commit the resource group changes.

5. Deploy the nodes whose resource groups for scheduling are changed.

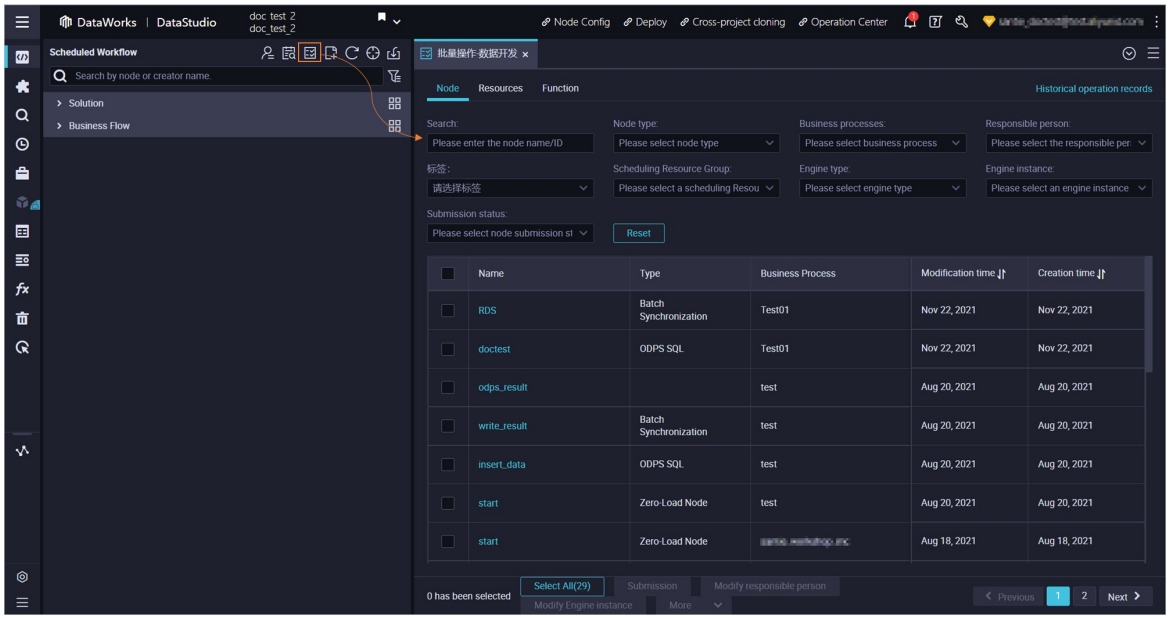
   After the resource groups for scheduling of nodes are modified, you must deploy the nodes to the production environment on the Deploy page. After that, the nodes are run on the newly allocated resource group for scheduling in the production environment.

10.6. Perform operations on multiple DataWorks objects at a time

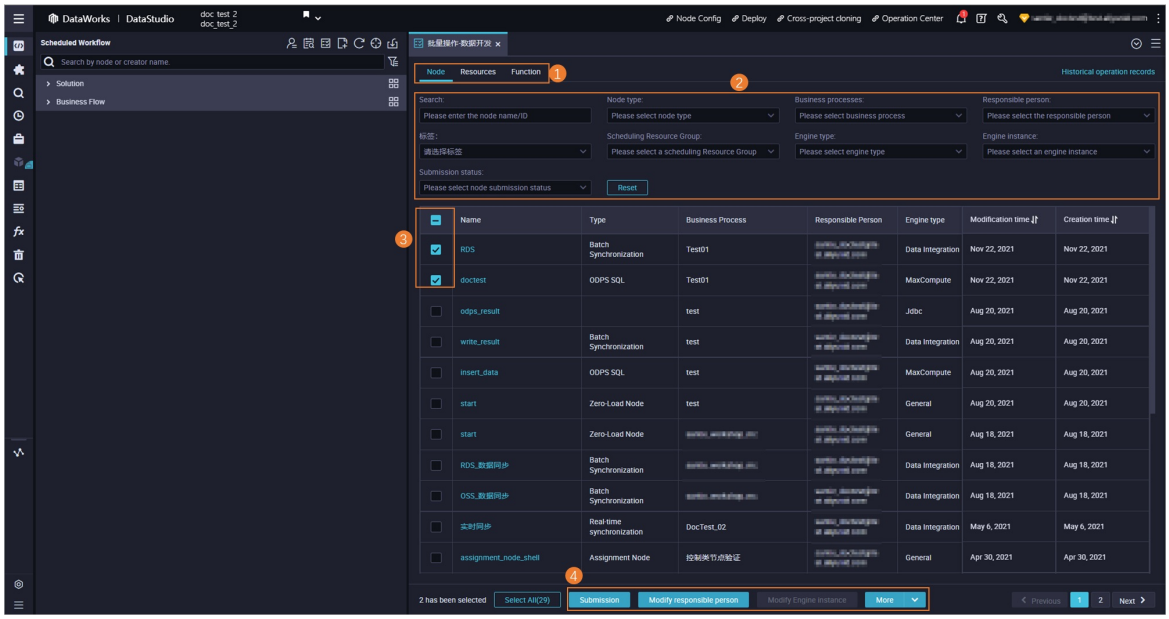
DataWorks allows you to modify the configurations, such as the owner, of multiple nodes, resources, and functions at a time. After that, you can commit and deploy the modified objects to the production environment at a time. This topic describes how to use the batch operations feature in DataWorks.

Procedure

1. Log on to the DataWorks console, go to the **DataStudio** page, and then click the **Batch Operation** icon in the top toolbar in the left-side navigation pane.



2. Modify multiple DataWorks objects at a time.



i. On the **Batch Operation-Data Development** tab, you can perform operations on multiple nodes, resources, or functions at a time on the **Node**, **Resource**, or **Function** tab.

ii. On each tab, you can set the filter conditions, such as the **Business processes** and **Responsible person** parameters, to search for required objects.

> Note
>
> - Batch sync nodes can be filtered by resource group for Data Integration, types of the source and the destination, and names of the source and the destination.
> - The filter conditions on the **Node** tab are different from those on the **Resource**, and the **Function** tabs.

iii. If the search results are displayed on the tab, you can select multiple nodes, resources, or functions that you want to modify.

iv. After that, click a button at the bottom to perform the operation on the selected objects as needed.

> Note
>
> - In the dialog box that appears, if you set the Mandatory modification parameter to **Yes**, nodes that are locked by other owners will be modified. If you set this parameter to **No**, only the nodes that are locked by you will be modified.
> - If you modify and commit the objects on the Batch Operation-Data Development tab, the operations take effect only in the development environment. To update the configurations of nodes, resources, or functions in the production environment, you must deploy the objects to the production environment on the Deploy page.

3. You can click **Historical operation records** in the upper-right corner to view the details of historical operations.

10.7. Import data to a MaxCompute table

DataWorks allows you to upload local files in the specified formats, such as CSV files, to DataWorks and import the data to MaxCompute tables. This topic describes how to import data to a MaxCompute table.

Prerequisites

A MaxCompute table is created.

You can specify an existing MaxCompute table or create a MaxCompute table. For more information about how to create a MaxCompute table, see Create a MaxCompute table.

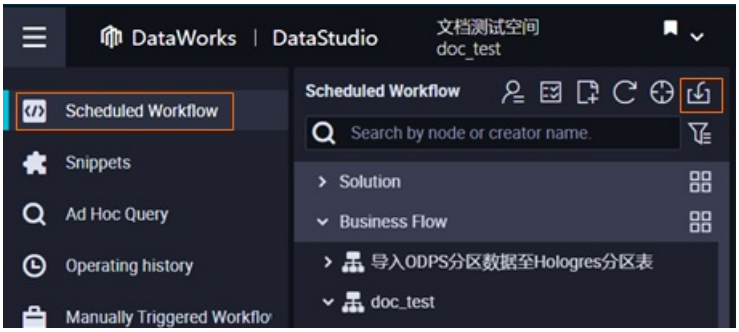
Limits

You can import only the data of your local files to a MaxCompute table.

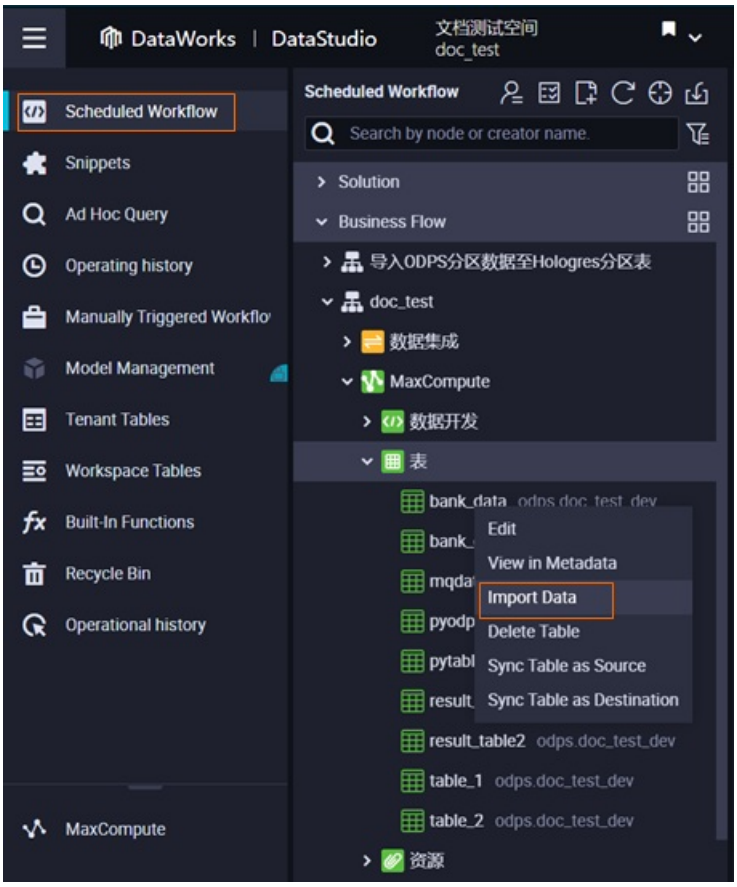
Entry points for data import

You can import data to a MaxCompute table in the top toolbar of the Scheduled Workflow pane, in the MaxCompute folder of the Business Flow section, or in the Workspace Tables pane. The following figures show the entry points for data import.

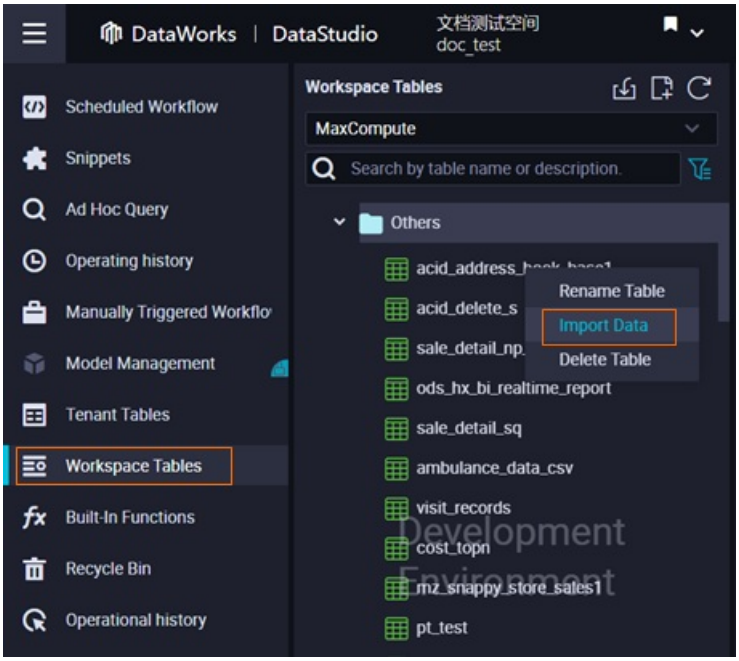
- Click the Import icon in the top toolbar of the **Scheduled Workflow** pane.



- In the **Scheduled Workflow** pane, find the required workflow, right-click the table to which you want to import data in the MaxCompute folder, and then select Import Data.



- In a workspace in standard mode, click **Workspace Tables** in the left-side navigation pane, right-click the table to which you want to import data, and then select Import Data.



Import data

You can import only the data of your local files to a MaxCompute table.

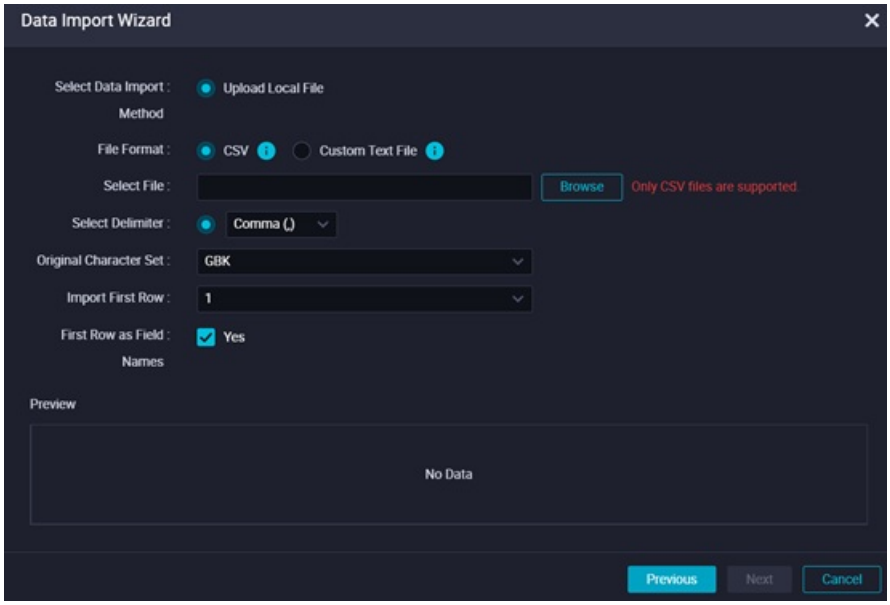
1. Perform one of the preceding operations to open the **Data Import Wizard** dialog box.
2. In the **Data Import Wizard** dialog box, check the name of the table to which you want to import data. Then, specify the partition to which the data is imported and check whether the partitions exist. After that, click **Next**.

   > **Note** If the table that you select is a non-partitioned table, click **Next** to skip this step.

   You can specify the partition to which the data is imported. After that, click **Check** to check whether the partition exists.

   - If the specified partition does not exist, the system creates a partition for subsequent operations and the data will be imported to the created partition.
   - If the specified partition exists, the data will be imported to this partition.

3. Set the parameters, such as the File Format, Select File, and Select Delimiter parameters, based on your business requirements. Then, click **Next**.

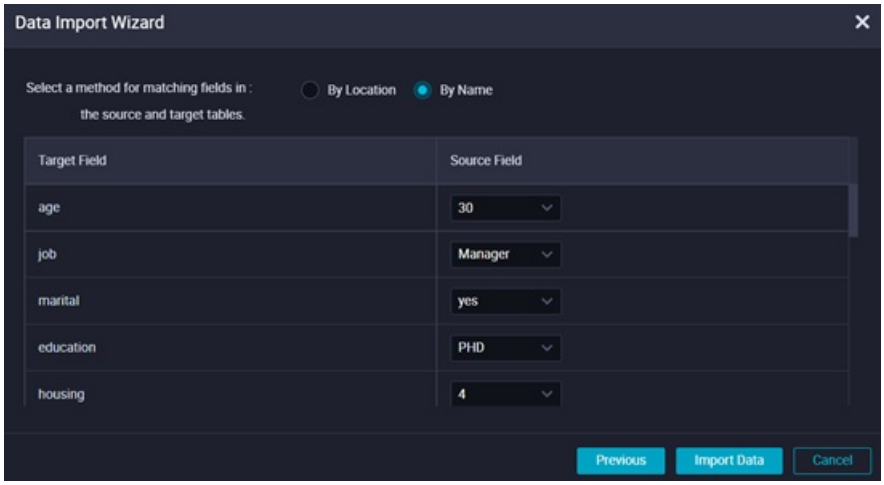


The following table describes the parameters.

Parameter	Description
File Format	The format of the file to be uploaded. Valid values: **CSV** and **Custom Text File**. If you set the parameter to Custom Text File, you can upload `.txt`, `.csv`, and `.log` files.
Select File	The file to be uploaded. You can click **Browse** and select the file to be uploaded as prompted.
Select Delimiter	The delimiter that is used to separate fields. Valid values: Comma (,), Tab, Semicolon (;), Space, |, #, and &. **Note** If you set the File Format parameter to Custom Text File, you can enter a character or string in the Select Delimiter field as the delimiter to separate fields.
Original Character Set	The encoding format of the file to be uploaded. Valid values: **GBK**, **UTF-8**, **CP936**, and **ISO-8859**.
Import First Row	The row from which the data is imported.
First Row as Field Names	Specifies whether the first row of the file to be imported is set as the header row. ○ If you select **Yes**, the first row of the file is not imported. ○ Otherwise, the first row of the file is imported.

4. Specify the method used to match the fields between the uploaded file and the MaxCompute table. Then, click **Import Data**.

You can set the Select a method for matching fields in the source and target tables parameter to By Location or By Name.



After the operation is complete, a message appears to inform you that the data of the local file is imported to the MaxCompute table. You can view the imported data by running an ad hoc query. For more information, see Create an ad hoc query.

10.8. Editor shortcuts

This topic describes the general shortcuts of the code editor.

Shortcuts for Google Chrome in Windows

`Ctrl+S` : Save the configuration.

`Ctrl+Z` : Cancel the action.

`Ctrl+Y` : Redo the action.

`Ctrl+D` : Select identical strings.

`Ctrl+X` : Cut a line.

`Ctrl+Shift+K` : Delete a line.

`Ctrl+C` : Copy the current line.

`Ctrl+I` : Select a line.

`Shift+Alt+Drag` : Multi-select consecutive lines to modify the content.

`Alt+Click` : Multi-select specific lines and indent them by pressing Tab.

`Ctrl+Shift+L` : Place the cursor after identical strings for batch edit.

`Ctrl+F` : Search for specific content.

`Ctrl+H` : Replace the searched data with required content.

`Ctrl+G` : Locate the specific line.

`Alt+Enter` : Select all matching strings.

`Alt+↓` or `Alt+↑` : Move the current line down or up.

`Shift+Alt+↓` or `Shift+Alt+↑`: Copy the current line down or up.

`Shift+Ctrl+K`: Delete the current line.

`Shift+Ctrl+\`: Move the cursor to matching parentheses.

`Ctrl+]` or `Ctrl+[`: Increase or decrease the units of indentation.

`Home` or `End`: Move the cursor to the beginning or end of the current line.

`Ctrl+Home` or `Ctrl+End`: Move the cursor to the beginning or end of the current file.

`Ctrl+→` or `Ctrl+←`: Move the cursor one word to the right or left.

`Shift+Ctrl+[` or `Shift+Ctrl+]`: Hide or show the area that is pointed by the cursor.

`Ctrl+K+Ctrl+[` or `Ctrl+K+Ctrl+]`: Hide or show the sub-area that is pointed by the cursor.

`Ctrl+K+Ctrl+0` or `Ctrl+K+Ctrl+j`: Hide or show all areas.

`Ctrl+/`: Comment or uncomment the line or code block that is pointed by the cursor.

Shortcuts for Google Chrome in Mac

`Cmd+S`: Save the configuration.

`Cmd+Z`: Cancel the action.

`Cmd+Y`: Redo the action.

`Cmd+D`: Select identical strings.

`Cmd+X`: Cut a line.

`Cmd+Shift+K`: Delete a line.

`Cmd+C`: Copy the current line.

`Cmd+I`: Select the current line.

`Cmd+F`: Search for specific content.

`Cmd+Alt+F`: Replace the searched data with required content.

`Alt+↓` or `Alt+↑`: Move the current line down or up.

`Shift+Alt+↓` or `Shift+Alt+↑`: Copy the current line down or up.

`Shift+Cmd+K`: Delete the current line.

`Shift+Cmd+\`: Move the cursor to matching parentheses.

`Cmd+]` or `Cmd+[`: Increase or decrease the units of indentation.

`Cmd+←` or `Cmd+→`: Move the cursor to the beginning or end of the current line.

`Cmd+↑` or `Cmd+↓`: Move the cursor to the beginning or end of the current file.

`Alt+→` or `Alt+←`: Move the cursor one word to the right or left.

`Alt+Cmd+[` or `Alt+Cmd+]`: Hide or show the area that is pointed by the cursor.

`Cmd+K+Cmd+[` or `Cmd+K+Cmd+]`: Hide or show the sub-area that is pointed by the cursor.

`Cmd+K+Cmd+0` or `Cmd+K+Cmd+j`: Hide or show all areas.

`Cmd+/` : Comment or uncomment the line or code block that is pointed by the cursor.

Multiple selections

`Alt+Click` : Insert cursors.

`Alt+Cmd+↑` or `Alt+Cmd+↓` : Move cursors up or down.

`Cmd+U` : Undo last cursor actions.

`Shift+Alt+I` : Insert cursors at the end of each line in the selected code block.

`Cmd+G` or `Shift+Cmd+G` : Find the next or previous string.

`Cmd+F2` : Select all the pointed strings.

`Shift+Cmd+L` : Select all the pointed lines.

`Alt+Enter` : Select all matching strings.

`Shift+Alt+Drag` : Multi-select lines.

`Shift+Alt+Cmd+↑` or `Shift+Alt+Cmd+↓` : Multi-select lines vertically.

`Shift+Alt+Cmd+←` or `Shift+Alt+Cmd+→` : Multi-select lines horizontally.

11.Setup

11.1. Personal settings

On the Personal Settings tab, you can customize the modules to be displayed in the left-side navigation pane of DataStudio, the settings of the code editor and the directed acyclic graph (DAG), and the theme of DataStudio. This topic describes the settings you can configure on this tab.

Go to the Personal Settings tab

1. Log on to the DataWorks console. In the left-side navigation pane, click **Workspaces**.
2. On the Workspaces page, find the required workspace and click **Data Development** in the Actions column to go to DataStudio.
3. In the lower-part of the left-side navigation pane, click the icon to go to the **Personal Settings** tab of the **Settings** page.

   On the Personal Settings tab, you can configure the following settings:

   - Customize the modules to be displayed in the left-side navigation pane of DataStudio. For more information, see Customize the modules to be displayed.
   - Configure the default settings of code editor, such as the minimap feature, the error check feature, and the front size. For more information, see Configure the settings of the code editor.
   - Specify the theme of DataStudio. For more information, see Configure the display settings of the DAG and DataStudio.

   After you configure these settings, click **Apply to All Workspaces** in the lower-right corner of the **Personal Settings** tab to apply these settings to all workspaces.

Customize the modules to be displayed

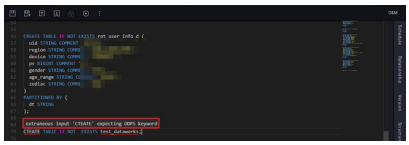
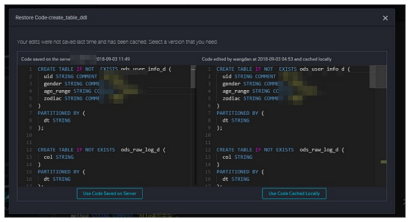
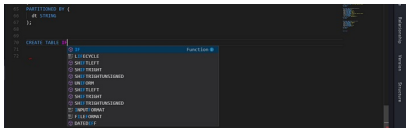
In the **DataStudio Modules** section of the Personal Settings tab, you can customize the modules to be displayed in the left-side navigation pane of DataStudio.

You can click **Public** or **MaxCompute** to go to the corresponding tab. Then, you can click modules that have a plus (+) icon in the **All Modules** section to add them to the **Added Modules** section. This way, the added modules are displayed in the left-side navigation pane of DataStudio. You can click the icons that represent these modules to perform corresponding data development operations.

> **Note**
>
> - If you cannot find a required module in the left-side navigation pane, go to the **DataStudio Modules** section of the Personal Settings tab and add the module to the Added Modules section.
> - If you want to remove a module from the left-side navigation pane, find the module in the **Added Modules** section and click the icon next to it. After that, the module is removed only from the left-side navigation pane of DataStudio, but not from DataWorks.

Configure the settings of the code editor

In the **Editor Settings** section of the Personal Settings tab, you can configure the settings of the code editor. The changes to the settings immediately take effect. The following table describes the parameters in the Editor Settings section.

Parameter	Description	Example
Minimap	Specifies whether to display a minimap that provides an overview of the code in the code editor. When the code is long, you can move the pointer in the minimap to specify the code block to be displayed.	
Error Annotation	Specifies whether to check the code syntax for errors. If you select this check box, the errors are marked in the red shading. If you move the pointer over the red shading, you can see the error details.	
Auto Save	If you select this check box, DataWorks automatically saves the code that is being edited. This way, if the code editor of a node is unexpectedly closed, you can click **Use Version Saved on Server** or **Use Version Saved in Local Cache** when you re-open the node.	
Auto-Completed Code Style	Specifies whether the keywords in the code are uppercase or lowercase.	
Auto Line Wrap	Specifies whether to automatically wrap lines for long clauses. Valid values: • **Disable** • **Enable**	None

Parameter	Description	Example
Wrap at Column	• If you set the Auto Line Wrap parameter to **Disable**, the value of Wrap at Column is *80* by default, which cannot be modified. • If you set the Auto Line Wrap parameter to **Enable**, you can set the maximum number of columns that is allowed in a line.	None
Code Font Size	Valid values: an integer from *12* to *18*. You can set this parameter based on your habits and code size.	None
Press Enter for Suggestions	If you select this check box, DataWorks automatically displays suggestions on code after you press the Enter key. **Note** • If you clear this check box, a new line is started after you press the Enter key. • In addition to the Enter key, you can also use the Tab key to trigger the display of suggestions on code.	
Auto Completion	You can enable the following types of hints for code completion: • **Continuous Smart Tips**: specifies whether to analyze the intentions of code writing and give hints on the code to write. • **Keyword**: specifies whether to enable keyword hints. • **Syntax Template**: specifies whether to enable syntax template hints. • **Project**: specifies whether to enable project name hints. • **Table Name**: specifies whether to enable table name hints. Higher priority is given to tables that are recently used. • **Field**: specifies whether to enable field name hints.	None

Configure the display settings of the DAG and DataStudio

In the **General Settings** section of the Personal Settings tab, you can configure the display settings of the DAG and DataStudio. The following table describes the parameters in the General Settings section.

Parameter	Description	Example
Display Node Engine Information on DAG	Specifies whether to display the information about compute engines of nodes in the DAG.	
Theme	Specifies the theme of DataStudio. You can set the theme to black or white.	None

11.2. Configure a code template

A code template provides the content that is displayed at the beginning of the code for a node. You can configure a code template for the following types of nodes based on your business requirements: ODPS SQL, ODPS MR, and Shell.

Limits

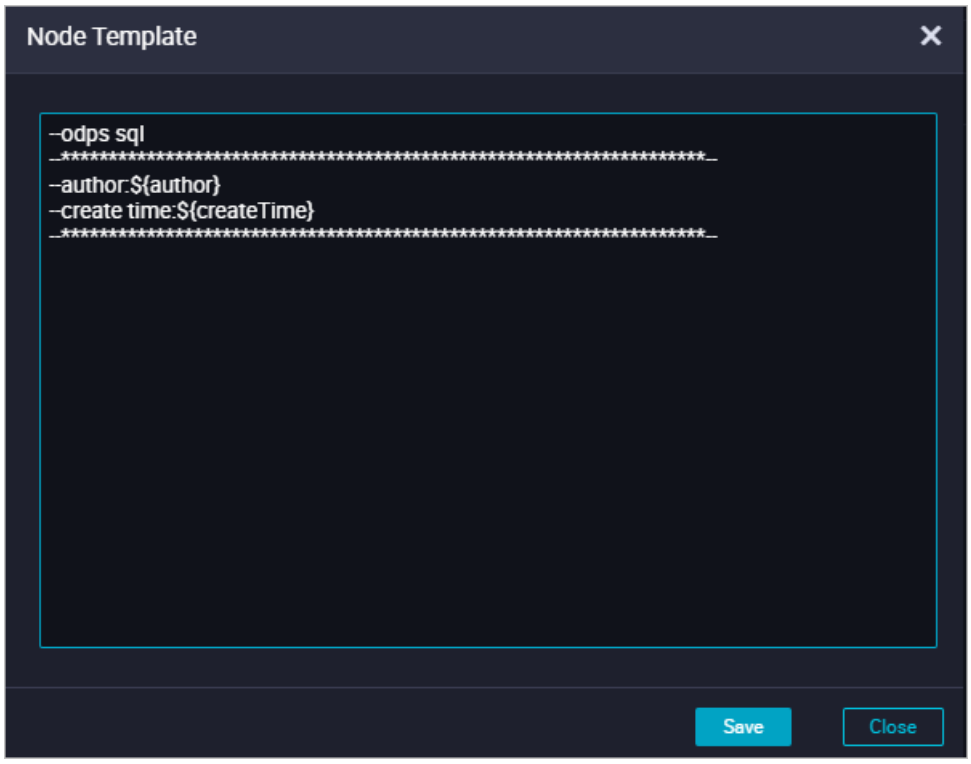
- Only a workspace administrator can modify code templates. If you want to modify a code template, you must obtain the permissions of a workspace administrator. For more information, see Manage roles and permissions.
- You can configure code templates only for the following types of nodes: ODPS SQL, ODPS MR, and Shell.

Configure a code template

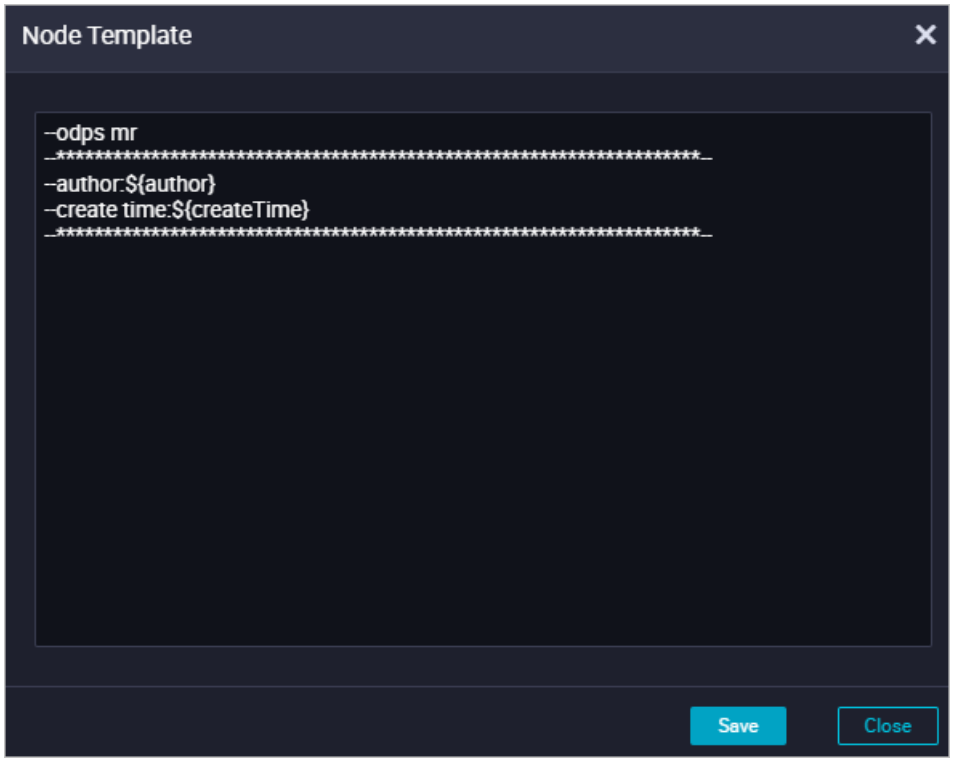
1. Go to the Code Templates tab.
   i. Log on to the DataWorks console. In the top navigation bar, select the region where your workspace resides. In the left-side navigation pane, click **Workspaces**.
   ii. On the Workspaces page, find your workspace and click **Data Development** in the Actions column to go to the DataStudio page.
   iii. In the left-side navigation pane of the DataStudio page, click the icon to go to the **Settings** page.
   iv. On the **Settings** page, click **Code Templates** to go to the **Code Templates** tab.
2. Configure a code template.
   i. Modify a code template.

   On the **Code Template** tab, find the code template that you want to modify and click **Change** in the Actions column. In the Node Template dialog box, you can modify the code template based on your business requirements. DataWorks provides the following types of default code templates:

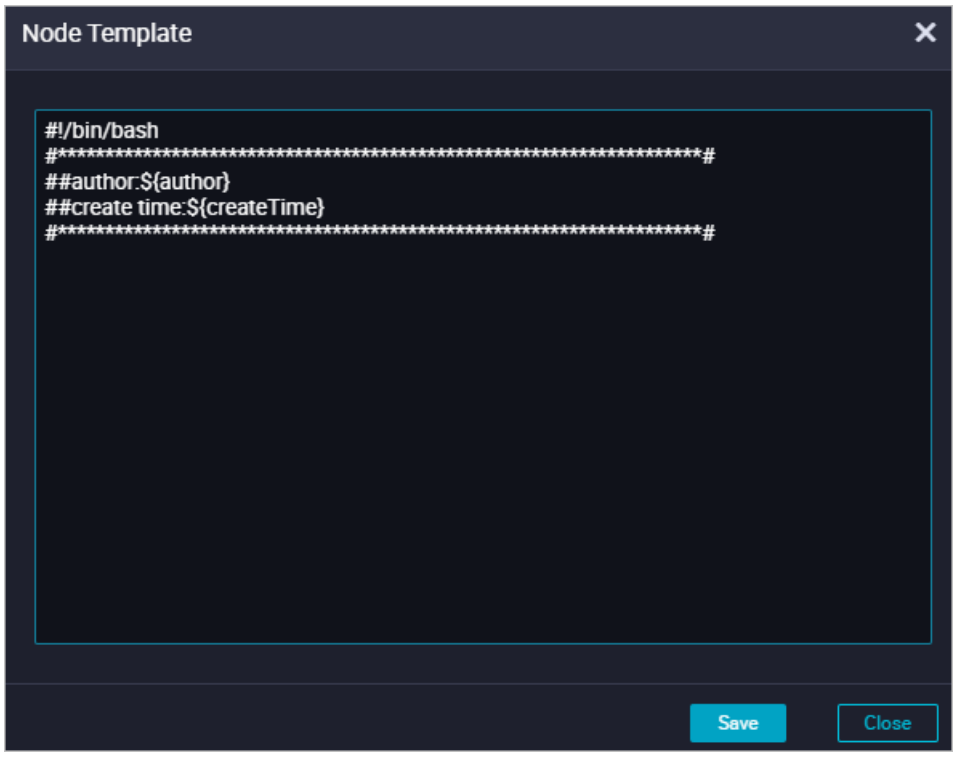
- Code template for ODPS SQL nodes



- Code template for ODPS MR nodes



- Code template for Shell nodes



ii. After you modify the code template, click **Save**.

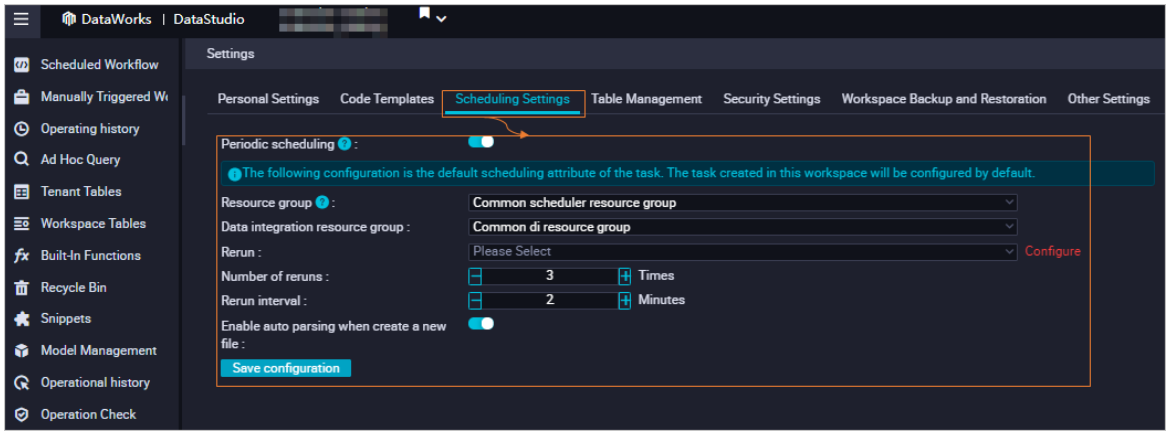
After you create a node of one of the preceding types, the related code template is used for the node.

11.3. Configure scheduling settings

To run auto triggered nodes as scheduled, you must go to the Scheduling Settings tab in DataStudio to enable periodic scheduling. This topic describes how to enable periodic scheduling and configure the default scheduling settings for auto triggered nodes.

Configure the default scheduling settings for auto triggered nodes

1. Log on to the DataWorks console. In the left-side navigation pane, click **Workspaces**.
2. Find the workspace that you want to manage and click **Data Development** in the Actions column to go to DataStudio.
3. In the lower part of the left-side navigation pane, click the icon to go to the **Settings** page.
4. Click the **Scheduling Settings** tab. On the tab that appears, set the following parameters.



Parameter	Description
Periodic scheduling	Specifies whether to enable periodic scheduling. Auto triggered nodes in the workspace can be run as scheduled only if you turn on this switch.
Resource group	The default resource group used to schedule nodes.
Data integration resource group	The default resource group for Data Integration used to schedule Data Integration nodes.
Rerun	The default rerun policy for auto triggered nodes. Valid values: ○ **Allow Regardless of Running Status** ○ **Allow upon Failure Only** ○ **Disallow Regardless of Running Status** **Note** If you set the **Rerun** parameter to Allow Regardless of Running Status or Allow upon Failure Only, make sure that the data idempotence of auto triggered nodes is not affected. Otherwise, data quality issues may arise after multiple reruns.
Number of reruns	The default number of times that an auto triggered node is rerun after it fails to be run as scheduled. Valid values: 1 to 10. A value of 1 indicates that the node is rerun once after it fails to be run as scheduled. A value of 10 indicates that the node is rerun ten times after it fails to be run as scheduled. You can change the value of this parameter based on your business requirements.
Rerun interval	The default interval between two consecutive reruns. Valid values: 1 to 30. Unit: minutes.

Parameter	Description
Enable auto parsing when create a new file	Specifies whether to enable automatic parsing for auto triggered nodes. If you enable this feature, after a node is committed, DataWorks automatically parses the output names of the node and its ancestor nodes based on the code.

5. Click **Save** to save the preceding settings that you configure.

   Then, DataWorks uses these settings as the default settings to schedule new auto triggered nodes.

11.4. Manage settings for tables

You can manage settings such as partition formats, identifiers of partition fields, table name prefixes, and folders, levels, and categories for tables on the Table Management tab of the DataStudio page.

Limits

Only a workspace administrator can create folders and classify tables into different categories by attribute such as purpose or name. Before you can create a folder, you must obtain the permissions of a workspace administrator. For more information, see Manage roles and permissions.

Go to the Table Management tab

1. Log on to the DataWorks console. In the left-side navigation pane, click **Workspaces**. Select the region where your workspace resides.
2. Find your workspace on the Workspaces page and click **Data Development** in the Actions column to go to the DataStudio page.
3. In the left-side navigation pane of the DataStudio page, click the  icon to go to the **Settings** page.
4. On the **Settings** page, click the **Table Management** tab.

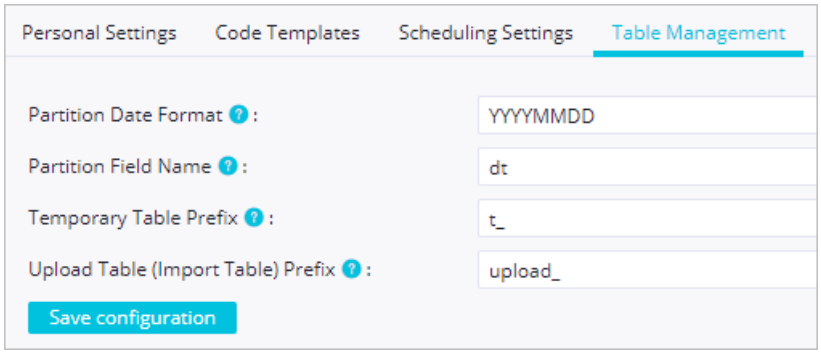
   On the Table Management tab, you can perform the following operations:

   - Configure basic attributes for tables. For more information, see Configure basic attributes for tables.
   - Create or manage a folder for tables. For more information, see Create or manage a folder for tables.
   - Create or manage a level or category for tables. For more information, see Create or manage a level or category for tables.

   After the desired operations are complete, click **Save configuration**.

Configure basic attributes for tables

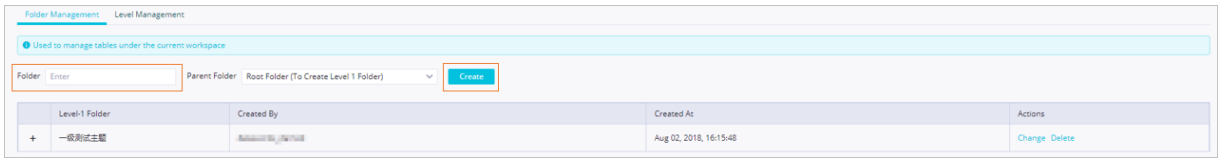
You can configure partition formats, identifiers of partition fields, and table name prefixes on the **Table Management** tab.



Parameter	Description
Partition Date Format	The date format of a partitioned table. Default value: **YYYYMMDD**.
Partition Field Name	The identifier of a partition field. We recommend that you set this parameter to **dt**.
Temporary Table Prefix	The prefix for the name of a temporary table. Default value: **t_**.
Upload Table (Import Table) Prefix	The prefix for the name of a table that is uploaded to DataStudio. Example: **upload_**.

Create or manage a folder for tables

Tables that you create in a workspace are displayed in the Workspace Tables pane of the DataStudio page. You can select a folder level when you create a table. Then, the table is stored in the folder at the specified level after the table is committed and deployed. Folders are used to store tables.

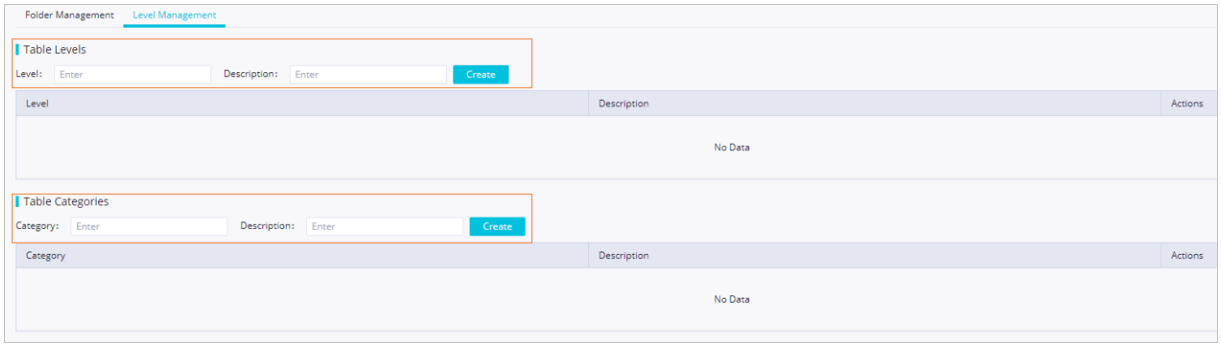


You can enter a folder name in the **Folder** field on the Table Management tab and click **Create** to create a folder. You can also click **Change** or **Delete** in the **Actions** column that corresponds to a folder to modify or delete the folder.

Create or manage a level or category for tables

The level management feature allows you to design physical levels for tables. Table levels are used to define and manage the layers of data warehouses. In most cases, we recommend that you divide a data warehouse into various layers, such as the operational data store (ODS) layer, data warehouse detail (DWD) layer, data warehouse summary (DWS) layer, and application data service (ADS) layer. For more information, see Create a data layer. Table categories allow you to categorize tables in a finer-grained manner from the business dimension.

> **Note** After a workspace is created, the system does not provide default levels for tables. You must create levels for tables as a project owner or workspace administrator based on your business requirements.



- **Table Levels**: used to define the levels to which tables belong. You can create, modify, or delete a level based on your business requirements.
  - Create a level: Enter a level name in the **Level** field, enter a description for the level in the **Description** field, and then click **Create**.
  - Modify or delete a level: Find the desired level and click **Change** or **Delete** in the **Actions** column.
- **Table Categories**: used to define the categories to which tables belong. You can create, modify, or delete a category based on your business requirements.
  - Create a category: Enter a category name in the **Category** field, enter a description for the category in the **Description** field, and then click **Create**.
  - Modify or delete a category: Find the desired category and click **Change** or **Delete** in the **Actions** column.

11.5. Configure security settings

DataWorks allows you to configure security settings based on your business requirements. You can determine whether to mask sensitive information in the returned results of queries that you perform in DataStudio in the current workspace.

Context

DataStudio provides built-in rules for data masking. If you turn on **Mask Data in Page Query Results** on the Security Settings tab, the system masks sensitive information displayed in the data that is returned after you run the code of a node in DataStudio based on the built-in rules for data masking. If the built-in rules for data masking do not meet your business requirements, you can create a custom rule for data masking in Data Security Guard. For more information, see Data Security Guard.

> **Note**
> - The configuration of **Mask Data in Page Query Results** takes effect only in the current workspace.
> - Data masking in DataStudio is dynamic data masking, which does not affect underlying data.

Limits

The configuration of Mask Data in Page Query Results takes effect only in the current workspace. If you want to mask sensitive information in the returned results of queries in all workspaces, you must turn on Mask Data in Page Query Results for all workspaces.

> **Note** For example, you turn on Mask Data in Page Query Results for Workspace A and do not turn on the switch for Workspace B. If a member in Workspace B is granted the permissions to query tables in Workspace A, data in the tables is displayed in plaintext when the member queries the tables from Workspace B.

Turn on Mask Data in Page Query Results

1. Go to the Security Settings tab.
   i. Log on to the DataWorks console, select a region, and then click **Workspaces** in the left-side navigation pane.
   i. On the Workspaces page, find the workspace that you want to manage and click **Data Development** in the Actions column. The DataStudio page appears.
   ii. In the lower part of the left-side navigation pane of the DataStudio page, click the icon to go to the **Settings** page.
   iii. On the **Settings** page, click **Security Settings**. The **Security Settings** tab appears.
2. On the **Security Settings** tab, turn on **Mask Data in Page Query Results**.

   The configuration takes effect immediately after you turn on Mask Data in Page Query Results. The system masks sensitive information in the returned results of queries that you perform in DataStudio based on the built-in rules for data masking.

   > **Note**
   > - You can use this feature together with Data Security Guard. If the built-in rules for data masking do not meet your business requirements, you can create a custom rule for data masking in Data Security Guard. For more information, see Data Security Guard.
   > - If you do not turn on Mask Data in Page Query Results, sensitive information may be leaked.

   The following table describes the built-in rules for data masking provided by DataWorks.

Item	Data masking rule	Data before masking	Data after masking
ID card number	Only the first and last digits in a 15-digit or 18-digit ID card number are displayed in plaintext. All the other digits are displayed as asterisks (*).	111222190002309999	1**********9
Mobile phone number	Only the first seven digits in a mobile phone number in the Chinese mainland are displayed in plaintext. The last four digits are displayed as asterisks (`*`).	13900001234	1390000****

Item	Data masking rule	Data before masking	Data after masking
Email address	◦ If the number of characters that precede the at sign (`@`) is greater than or equal to three, the first three characters are displayed in plaintext, and all the other characters that precede the at sign (@) are displayed as asterisks (*). ◦ If the number of characters that precede the at sign (`@`) is less than three, all the characters are displayed in plaintext and three asterisks (`***`) are added before the at sign (@).	◦ username@example.com ◦ a@example.net	◦ use***@example.com ◦ a***@example.net
Bank card number	Only the last four digits in a credit card number or deposit card number are displayed in plaintext. All the other digits are displayed as asterisks (*).	◦ 6888 8888 8888 8888 ◦ 4666 6666 6666 6666	◦ **** **** **** 8888 ◦ **** **** **** 6666
IP or MAC address	Only the first section of an IP address or a media access control (MAC) address is displayed in plaintext. All the other sections are displayed as asterisks (`*`).	◦ 192.168.0.1 ◦ 01-80-C2-00-00-00	◦ 192.***.*.* ◦ 01-**-**-**-**-**
License plate number	`Only the region information and the last three characters of the license plate number are displayed in plaintext` . All the other characters are displayed as asterisks (`*`).	◦ (One-character provincial abbreviation) A666666 ◦ (One-character provincial abbreviation) A888888	◦ (One-character provincial abbreviation) A***666 ◦ (One-character provincial abbreviation) A***888

11.6. Workspace backup and restoration

The workspace backup and restoration feature is used for code migration between workspaces. This topic describes how to back up and restore a workspace.

On the **DataStudio** page, click the icon in the lower-left corner. The **Setup** page appears on the right.

In the top navigation bar, click **Workspace Backup and Restoration**. The **Workspace Backup and Restoration** tab appears.

- On the **Backup** tab, you can compress the node code, node dependencies, resources, and functions in a workspace into one package.
- On the **Restore** tab, you can restore a workspace to its original scheduling configurations. After the workspace is restored, all nodes in the workspace are saved but not committed.

Go to the Workspace Backup and Restoration tab

1. Log on to the DataWorks console. Find the required workspace and click **Data Analytics**.
2. Click the icon in the lower-left corner. The **Setup** page appears on the right.
3. In the top navigation bar, click **Workspace Backup and Restoration**.

Back up the workspace

A workspace backup is a compressed package that contains the node code, node dependencies, resources, and functions in a workspace.

- Only workspace administrators can export backups and restore data from backups on the Setup page.
- Workflows and node groups of earlier versions cannot be backed up. We recommend that you use workflows of the latest version for data analytics.
- A node that is backed up to a path in a workspace will override the original node with the same name in the path. We recommend that you create another workspace to restore data.
- Data in tables is not backed up when you back up a workspace. You can synchronize the table data in the following ways:
  - Click the **Workspace Manage** icon in the upper-right corner. Go to the **Data Source** page and configure a MaxCompute connection. Then, create a sync node to back up the data.
  - In Workspace A, execute the DDL statement `create table select * from Workspace B.Table name` to migrate the data.

1. Go to the **Workspace Backup and Restoration** tab. Click the **Backup** tab. Then, click **Create Backup** in the upper-right corner.
2. In the **Create Backup** dialog box, set the **Method** and **Version** parameters.
   - **Method**

     You can set the backup method to **Full Backup** or **Incremental Backup**.

     - **Full Backup**: backs up all the node code, node dependencies, resources, and functions in the workspace.
     - **Incremental Backup**: backs up all the new or modified nodes from the timestamp that is specified by the **Start Timestamp** parameter to the current time.

     > **Note** If you use the incremental backup method, make sure that the dependencies between incremental sync nodes are correct. Otherwise, the workspace may fail to be restored. We recommend that you set this parameter to **Full Backup**.

   - **Version**

     Valid values are **Public Cloud**, **Apsara Stack V3.6.1 - V3.8.1**, and **Apsara Stack < V3.6.1**.

3. After the configuration is completed, click **Create**.

Restore the workspace

1. Go to the **Workspace Backup and Restoration** tab. Click the **Restore** tab. Then, click **Restore** in the upper-right corner.
2. In the **Restore** dialog box, click **Select File**.

   > **Note** You can upload the compressed package that you previously backed up to the workspace.

3. After the configuration is completed, click **Restore**.
4. In the **Set Compute Engine Mapping** dialog box, set the **Compute Engine Instances of Target Workspace** parameter.

   If the workspace that you backed up contains multiple compute engines, the system scans all compute engine instances during restoration. The system restores only nodes of the existing compute engines in the workspace to be restored. In this case, you must configure the mappings between the compute engines in the source and destination before you restore the destination workspace.

   > **Note**
   > - If the workspace to be restored does not contain a compute engine type such as E-MapReduce, or no instance is available for the compute engine type, nodes of this engine type are not restored.
   > - The custom node types vary based on the region. Therefore, you must also configure the mappings between the custom node types in the source and destination. For example, you must configure the mappings for the existing custom node types such as Hologres development node, Data Lake Analytics node, AnalyticDB for MySQL node, and AnalyticDB for PostgreSQL node.

11.7. Other settings

DataWorks supports various settings for data development. On the **Other Settings** tab in DataStudio, you can enable forcible code review and specify one or more code reviewers to control the code quality of your nodes. You can also delete all invalid DATABLAU data models on this tab. This topic describes the settings that you can configure on the **Other Settings** tab in DataStudio.

Limits

The following limits apply to the deletion of DATABLAU data models:

- Only after the DATABLAU service expires and before the service is renewed can you delete DATABLAU data models.
- Only **workspace administrators** can delete DATABLAU data models.

Go to the Other Settings tab

1. Log on to the DataWorks console. In the left-side navigation pane, click **Workspaces**.
2. In the top navigation bar, select the region in which the workspace that you want to manage

resides. Find the workspace and click **DataStudio** in the Actions column to go to DataStudio.

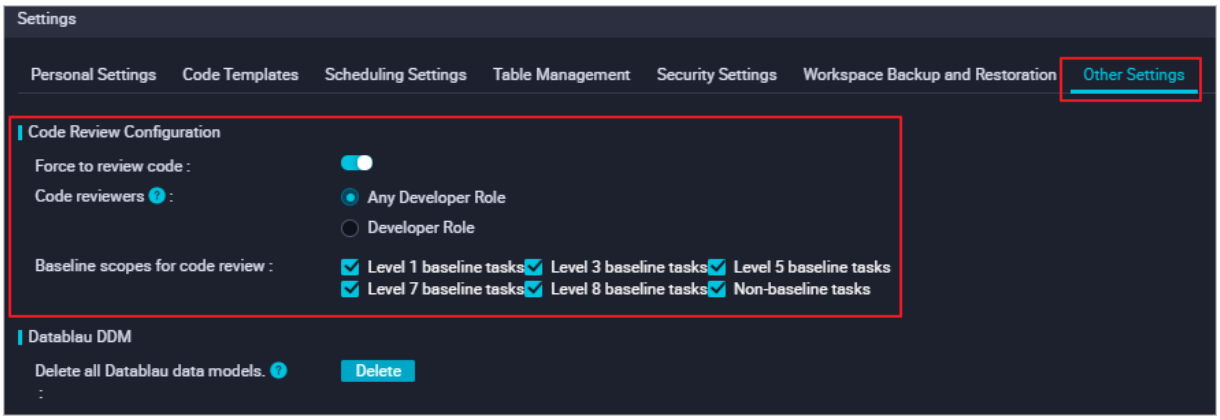
3. In the lower part of the left-side navigation pane, click the icon to go to the **Settings** page.
4. On the **Settings** page, click **Other Settings** to go to the **Other Settings** tab.

   On this tab, you can perform the following operations:

   - Enable the forcible code review feature. For more information, see Enable the forcible code review feature.
   - Delete all DATABLAU data models. For more information, see Delete all DATABLAU data models.

Enable the forcible code review feature

The forcible code review feature helps you control the code quality of your nodes. To enable and configure forcible code review, turn on the Force to review code switch and set the Code reviewers parameter to **Any Developer Role** or **Developer Role**. If you select Developer Role, you must manually specify one or more reviewers. After forcible code review is enabled, a node can be deployed only after its code is approved by a reviewer.

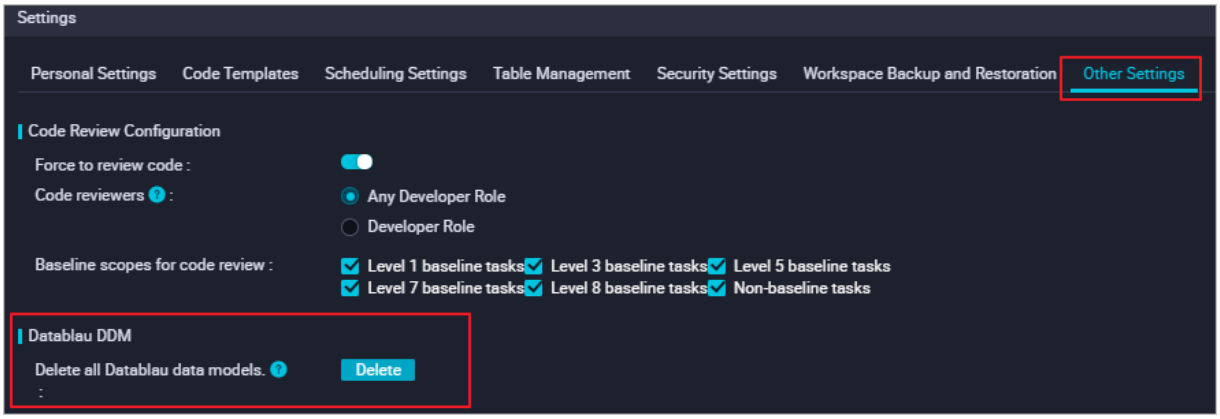


You can also set the Baseline scopes for code review parameter. Valid values of the parameter are **Level 1 baseline tasks**, **Level 3 baseline tasks**, **Level 5 baseline tasks**, **Level 7 baseline tasks**, **Level 8 baseline tasks**, and **Non-baseline tasks**.

- You can specify one or more baseline scopes. A baseline scope contains a group of baselines at the same priority level. The priority level is identified by a fixed value. The higher the value, the higher the priority level. For more information, see 基线管理.
- After a baseline scope is specified, the code of the nodes within the scope is reviewed by a reviewer after the nodes are committed. You can deploy a node only after the reviewer approves the node code.
- If multiple nodes are run at the same time, those at a higher priority level preempt resources. Therefore, we recommend you specify an appropriate baseline scope for nodes at a higher priority level so that reviewers can detect and stop the nodes that preempt resources and contain code errors from being committed to and run in the development environment. This way, you can ensure that resources are allocated in a proper and efficient manner. For more information about code review, see Code review.

Delete all DATABLAU data models

After the DATABLAU service expires, existing DATABLAU data models cannot be used if you do not renew the service. To delete all existing DATABLAU data models, click **Delete** in the **Datablau DDM** section on the Other Settings tab.



> **Note**
>
> - Only after the DATABLAU service expires and before the service is renewed can you delete DATABLAU data models.
> - Only **workspace administrators** can delete DATABLAU data models.
> - The deletion of DATABLAU data models does not affect the table schemas and data on compute engines, but DATABLAU data models cannot be recovered once deleted. We recommend you perform this operation with caution.

11.8. Workspace settings

The Workspace Settings tab displays five parameters: Partition Column Date Format, Partition Field Name, Temporary Table Prefix, Upload Table (Import Table) Prefix, and Mask Data in Page Query Results.

Click the icon in the lower-left corner of the **DataStudio** page to display the **Settings** page on the right side.

In the top navigation bar, click the **Workspace Settings** tab to configure a workspace.

Parameter	Description
Partition Column Date Format	The default date format of partition field values. You can modify the format based on your business requirements.
Partition Field Name	The default name of the partition field.
Temporary Table Prefix	The prefix of temporary table names. By default, tables with the prefix t_ in their names are identified as temporary tables.
Upload Table (Import Table) Prefix	The prefix of the names of tables uploaded on the **DataStudio** page.

Parameter	Description
Mask Data in Page Query Results	Specifies whether to de-identify the data in query results. If the switch is turned on, the result returned for an ad hoc query node in the current workspace is de-identified.

Enable de-identification for DataWorks workspaces

De-identification needs to be enabled for DataWorks workspaces one by one. After de-identification is enabled, the results returned for ad hoc query nodes in the current workspace are de-identified. The data stored at underlying layers is not affected because only dynamic de-identification is performed.

> **Note** For example, de-identification is enabled for Workspace A but is disabled for Workspace B. If you initiate a request in Workspace B to query tables in Workspace A, the query result is displayed in plaintext.

Click the **Workspace Settings** tab and turn on **Mask Data in Page Query Results**. Then, click **Save** in the lower-right corner of the tab. The results returned for ad hoc query nodes in the current workspace will be de-identified.

> **Note** By default, the Mask Data in Page Query Results switch is turned off, and you are not allowed to download data.

After de-identification is enabled for a DataWorks workspace, data of the types listed in the following table is de-identified by default.

Type	De-identification rule	Raw data	De-identified data
ID card information	Only the first and last digits in a 15- or 18-digit ID card number are displayed in plaintext. All the other digits are displayed as asterisks (*).	111222190002309999	1**********9
Mobile phone number	Only the first seven digits in a mobile number in the Chinese mainland are displayed in plaintext. The last four digits are displayed as asterisks (*).	13900001234	1390000****
Email address	If the string before the at sign (@) in an email address contains three or more characters, only the leftmost three characters are displayed in plaintext, followed by three asterisks (*). If the string before the at sign (@) contains only one or two characters, the entire string is displayed in plaintext, followed by three asterisks (*).	• username@example.com • a@example.net	• use***@example.com • a***@example.net

Type	De-identification rule	Raw data	De-identified data
Bank card number	Only the last four digits in a credit or deposit card number are displayed in plaintext. All the other digits are displayed as asterisks (*).	• 6888 8888 8888 8888 • 4666 6666 6666 6666	• **** **** **** 8888 • **** **** **** 6666
IP address or MAC address	Only the first segment in an IP address or a MAC address is displayed in plaintext. All the other characters are displayed as asterisks (*).	• 192.168.0.1 • 01-80-C2-00-00-00	• 192.***.*.* • 01-**-**-**-**-**
License plate number	Only the one-character provincial abbreviation and the last three characters in a license plate number in the Chinese mainland are displayed in plaintext. All the other characters are displayed as asterisks (*).	• (One-character provincial abbreviation) A 666666 • (One-character provincial abbreviation) A 888888	• (One-character provincial abbreviation) A***666 • (One-character provincial abbreviation) A***888

Note If you want to de-identify more types of data or have special requirements on the formats in which data is de-identified, complete your de-identification settings in Data Protection. The feature of de-identifying data for DataWorks workspaces must work with Data Security Guard. For more information, see Overview.