Open for Innovation
KNIME

KNIME Databricks Integration User Guide

KNIME AG, Zurich, Switzerland
Version 4.2 (last updated on 2020-09-02)

Table of Contents

Overview

KNIME Analytics Platform, from version 4.1 onwards, includes a set of nodes to support Databricks™. They allow to connect to a Databricks cluster running on Microsoft Azure™ or Amazon AWS™ cluster.

The KNIME Databricks Integration is available on the KNIME Hub.

Beside the standard paid service, Databricks also offers a free community edition for testing and education purposes, with access to a very limited cluster running a manager with 6GB of RAM, but no executors.

Create a Databricks cluster

For a detailed instruction on how to create a Databricks cluster, please follow the tutorial provided by Databricks. During cluster creation, the following features might be important:

- Autoscaling: Enabling this feature allows Databricks to dynamically reallocate workers for the cluster depending on the current load demand
- Auto termination: Specify an inactivity period, after which the cluster will terminate automatically. Alternatively, you can enable the option *Terminate cluster on context destroy* in the Create Databricks Environment node configuration dialog, to terminate the cluster when the Spark Context is destroyed, e.g. when the Destroy Spark Context node is executed. For more information on the *Terminate cluster on context destroy* checkbox or the Destroy Spark Context node, please check the Advanced section.

The autoscaling and auto termination features, along with other features during cluster creation might not be available in the free Databricks community edition.

Connect to Databricks

Register the JDBC driver

Before connecting to Databricks in KNIME Analytics Platform, the Databricks JDBC driver has to be registered in KNIME. Please follow the instructions on how to register the Databricks JDBC driver to KNIME in the Database Documentation.

Connect to a Databricks cluster

This section describes how to configure the Create Databricks Environment node to connect to a Databricks cluster from within KNIME Analytics Platform.

Before connecting to a cluster, please make sure that the cluster is already created in Databricks. Check the section Create a Databricks cluster for more information on how to create a Databricks cluster.

After creating the cluster, open the configuration dialog of the Create Databricks Environment node. When configuring it you need to provide the following information:

1. The full Databricks deployment URL: The URL is assigned to each Databricks deployment. For example, if you use Databricks on AWS and log into *https://1234-5678-abcd.cloud.databricks.com/*, that is your Databricks URL.

   The URL looks different depending on whether it is deployed on AWS or Azure.

   In the free Databricks community edition, the deployment URL is *https://community.cloud.databricks.com/*.

   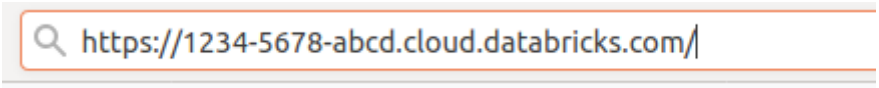

   Figure 1. Databricks deployment URL on AWS

   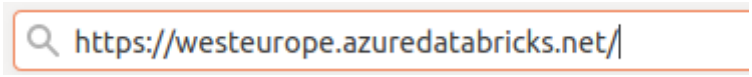

   Figure 2. Databricks deployment URL on Azure

2. The cluster ID: Cluster ID is the unique ID for a cluster in Databricks. To get the cluster ID, click the *Clusters* tab in the left pane and then select a cluster name. You can find the cluster ID in the URL of this page *<databricks-url>/#/settings/clusters/<cluster-id>/configuration*.

   The URL in the free Databricks community edition is similar to the one on Azure Databricks (see Figure 4).

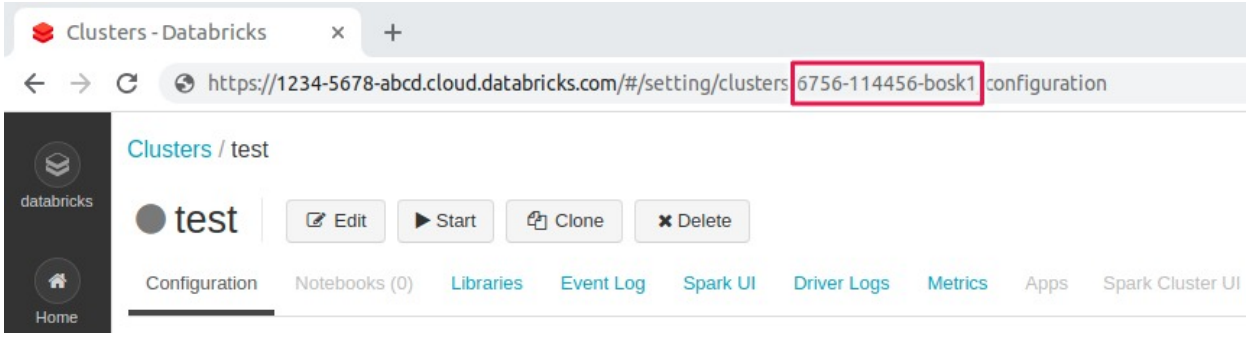

Figure 3. Cluster ID on AWS

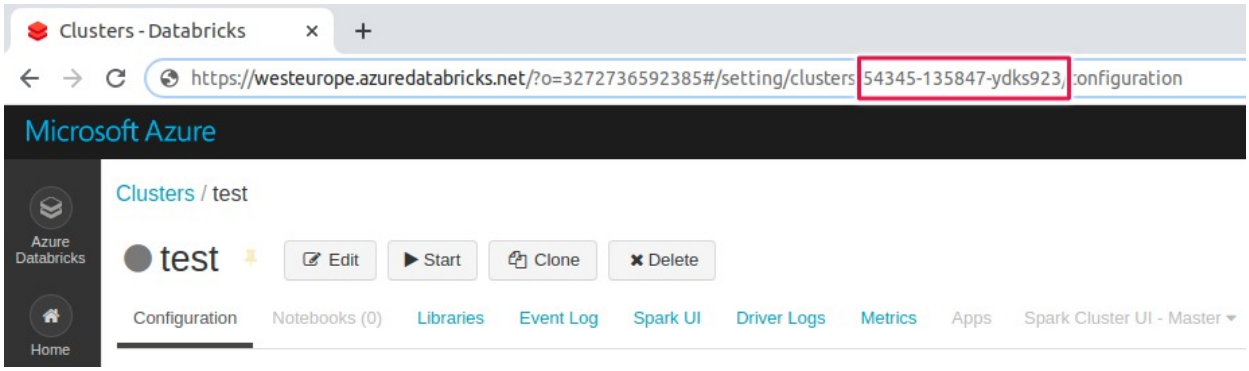

Figure 4. Cluster ID on Azure

3. Workspace ID: Workspace ID is the unique ID for a Databricks workspace. It is only available for Databricks on Azure, or if using the free Databricks community edition. For Databricks on AWS, just leave the field blank.

   The workspace ID can also be found in the deployment URL. The random number after *o=* is the workspace ID, for example, *https://<databricks-instance>/?o=327273659238_5*.

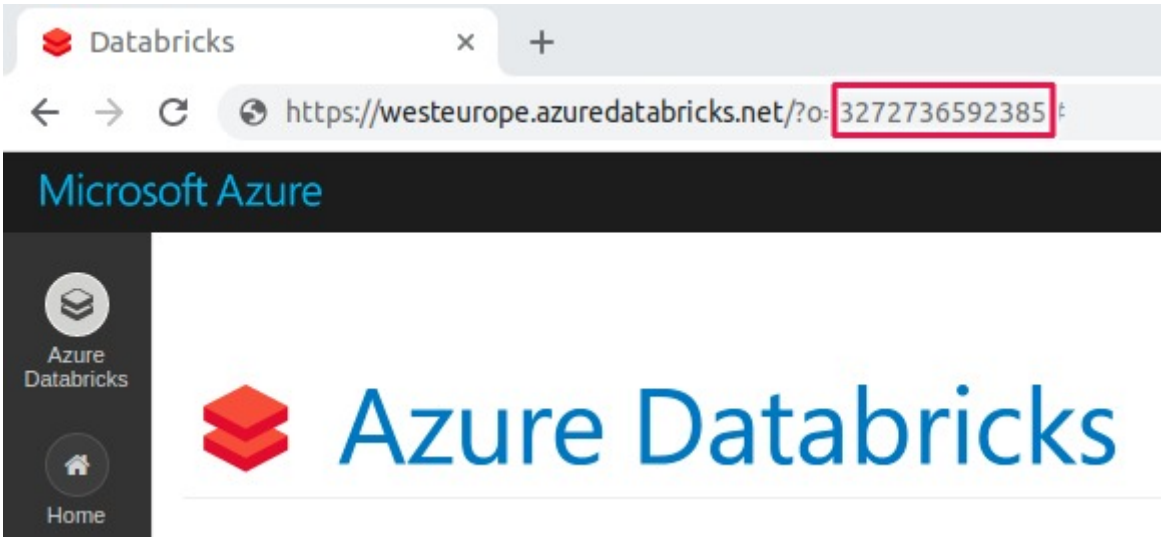

Figure 5. Workspace ID on Azure

For more information on the Databricks URLs and IDs please check the Databricks documentation.

4. Authentication: Token is strongly recommended as the authentication method in Databricks. To generate an access token:
   a. In your Databricks workspace, click on the user profile icon on the upper right corner and select *User Settings*
   b. Navigate to the *Access Tokens* tab

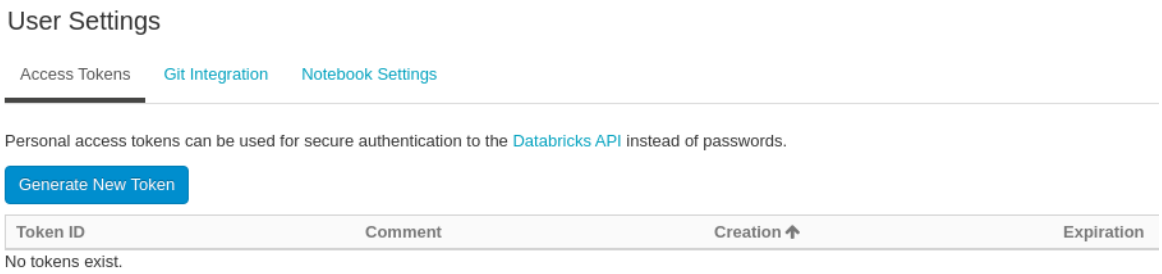

Figure 6. The Access Tokens tab

   c. Click *Generate New Token*, and optionally enter the description and the token lifetime. At the end click the *Generate* button.

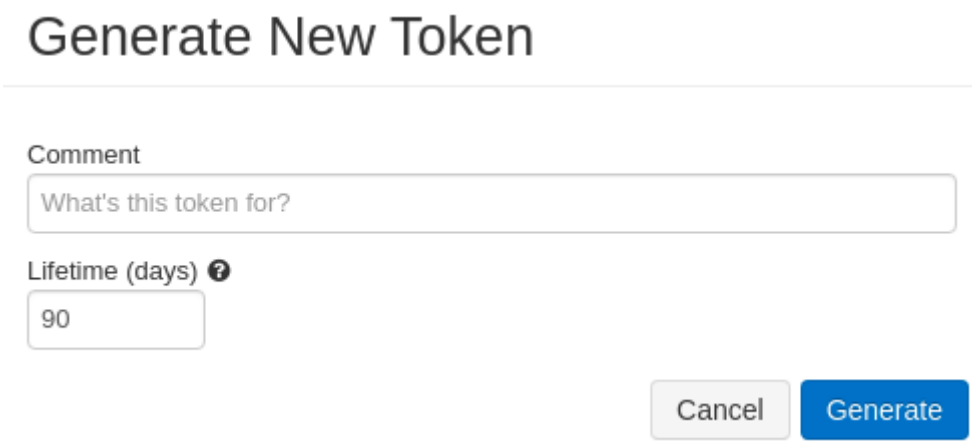

Figure 7. Generate new token

   d. Store the generated token in a safe location.

For more information on Databricks access token, please check the Databricks documentation.

Access token is unfortunately not available in the free Databricks community edition. Please use the username and password option as an alternative.

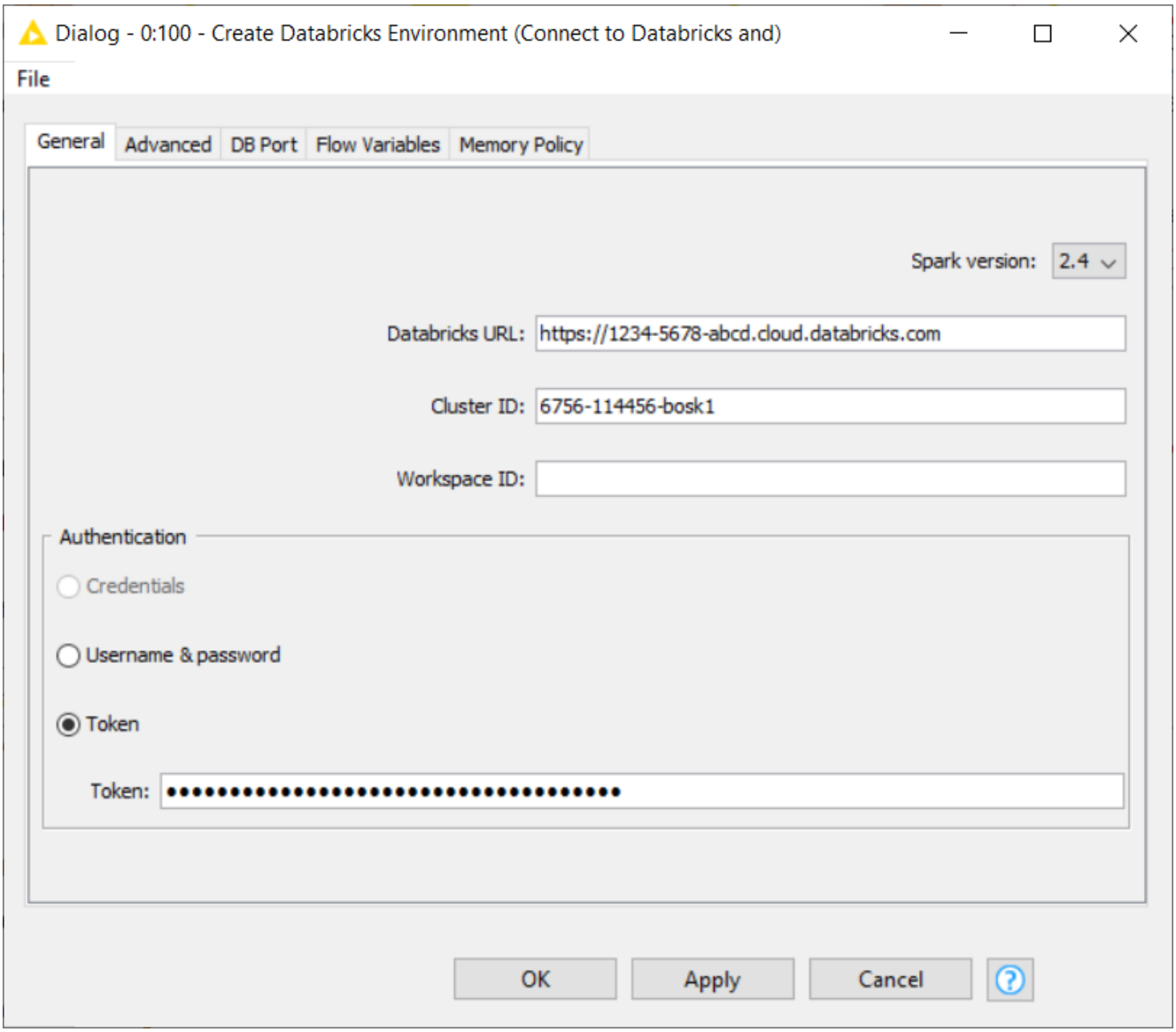

Figure 8. Create Databricks Environment node configuration dialog

After filling all the necessary information in the Create Databricks Environment node configuration dialog, execute the node. If required, the cluster is automatically started. Wait until the cluster becomes ready. This might take some minutes until the required cloud resources are allocated and all services are started.

The node has three output ports:

- Red port: JDBC connection which allows connecting to KNIME database nodes
- Blue port: DBFS connection which allows connecting to remote file handling nodes as well as Spark nodes
- Grey port: Spark context which allows connecting to all Spark nodes.

The remote file handling nodes are available under *IO > File Handling > Remote* in the node repository of KNIME Analytics Platform.

These three output ports allow you to perform a variety of tasks on Databrick clusters via KNIME Analytics Platform, such as connecting to a Databricks database and performing database manipulation via KNIME DB nodes or executing Spark jobs via KNIME Spark nodes, while pushing down all the computation process into the Databricks cluster.

Advanced

To configure more advanced options, navigate to the *Advanced* tab in the Create Databricks Environment node. For example, the following settings might be useful:

- *Create Spark context* checkbox is enabled by default to run KNIME Spark jobs on Databricks. However, if your cluster runs with Table Access Control, please make sure to disable this option because TAC does not support a Spark execution context.
- Enabling the *Terminate cluster on context destroy* checkbox will terminate the cluster when the node is reset, when the Destroy Spark Context node is executed, or when the workflow or KNIME Analytics Platform is closed. This might be important if you need to release resources immediately after being used. However, use this feature with caution! Another option is to enable the auto termination feature during cluster creation, where the cluster will auto terminate after a certain period of inactivity.
- Additionally, the DB Port tab contains all database-related configurations, which are explained in more details in the KNIME Database Extension Guide.

Working with Databricks

This section describes how to work with Databricks in KNIME Analytics Platform, such as how to access data from Databricks via KNIME and vice versa, how to use Databricks Delta features, and many others.

Databricks File System (DBFS)

Databricks File System (DBFS) is a distributed file system mounted on top of a Databricks workspace and is available on Databricks clusters. It allows you to persist files to object storage so that no data will get lost once a cluster is terminated, or to mount object storages, such as AWS S3 buckets, or Azure Blob storage.

Databricks File System Connection node

The Databricks File System Connection node allows you to connect directly to Databricks File

System (DBFS) without having to start a cluster as is the case with the Create Databricks Environment node, which is useful for simply getting data in or out of DBFS.

In the configuration dialog of this node, please provide the following information:

- The domain of the Databricks deployment URL, e.g. *1234-5678-abcd.cloud.databricks.com*
- The access token or username/password as the authentication method.

Please check the Connect to a Databricks cluster section for information on how to get the Databricks deployment URL and generate an access token.

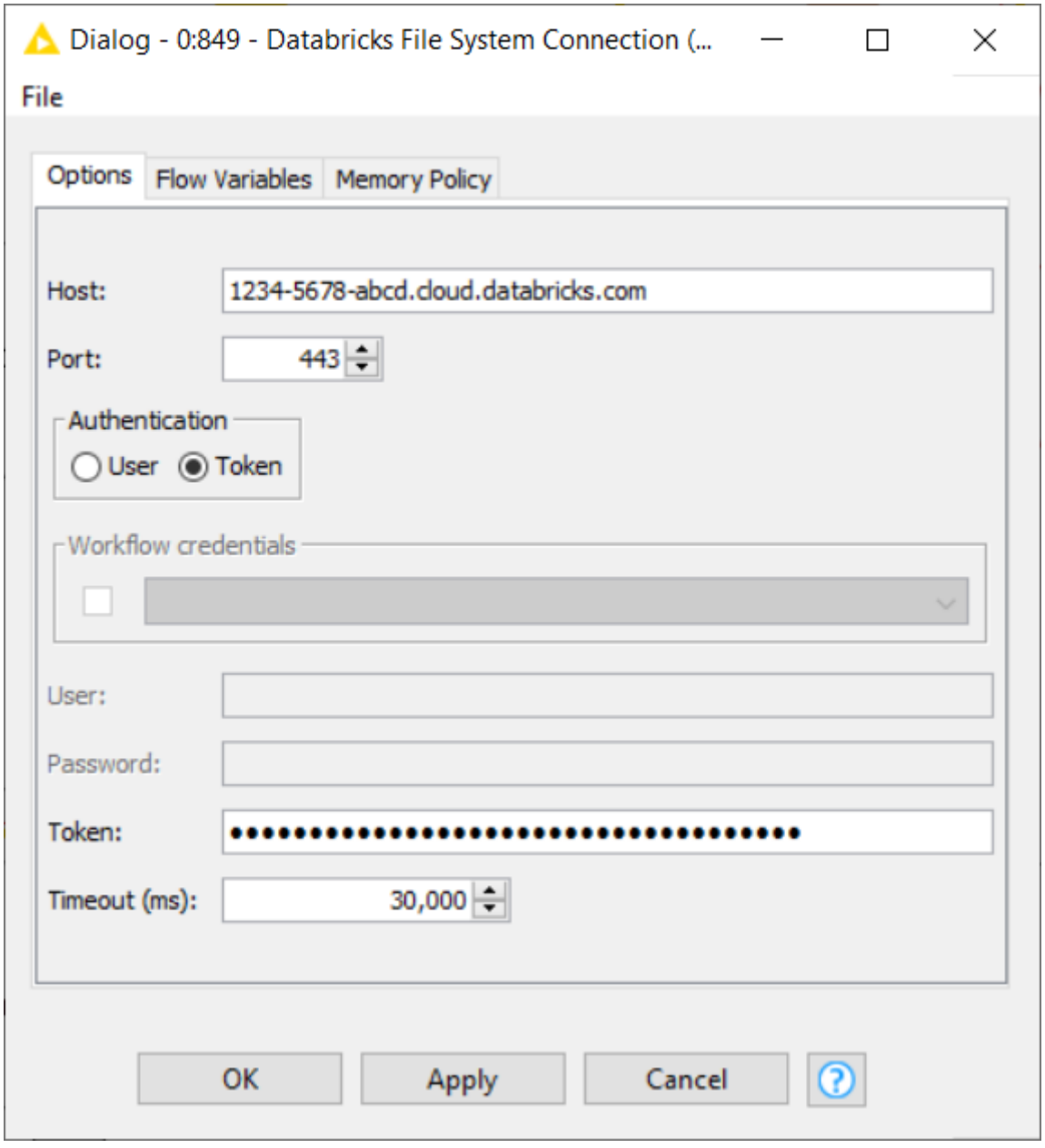

Figure 9. Configuration dialog of a Databricks File System Connection node

The output DBFS port (blue) of this node can be connected to any number of the remote file

handling nodes which are available under *IO > File Handling > Remote* in the node repository of KNIME Analytics Platform.

Mount AWS S3 buckets and Azure Blob storage to DBFS

DBFS allows mounting object storage, such as AWS S3 buckets, or Azure Blob storage. By mounting them to DBFS the objects can be accessed as if they were on a local file system. Please check the following documentation from Databricks for more information on how to:

- For AWS S3 buckets
  - Mount and unmount AWS S3 buckets with DBFS
  - Encrypt data when writing to S3 through DBFS
- For Azure Blob storage
  - Mount and unmount Azure Blob storage containers to DBFS
  - Mount Azure Data Lake Storage Gen1 resource using a service principal and OAuth 2.0
  - Mount an Azure Data Lake Storage Gen2 account using a service principal and OAuth 2.0

Spark IO nodes

KNIME Analytics Platform supports reading various file formats, such as Parquet or ORC that are located in DBFS, into a Spark DataFrame, and vice versa. It also allows reading and writing those formats directly from/in KNIME tables using the Reader and Writer nodes.

The KNIME Extension for Apache Spark is available on the KNIME Hub. These Spark IO nodes will then be accessible under *Tools & Services > Apache Spark > IO* in the node repository of KNIME Analytics Platform.

Spark to Parquet/ORC to Spark

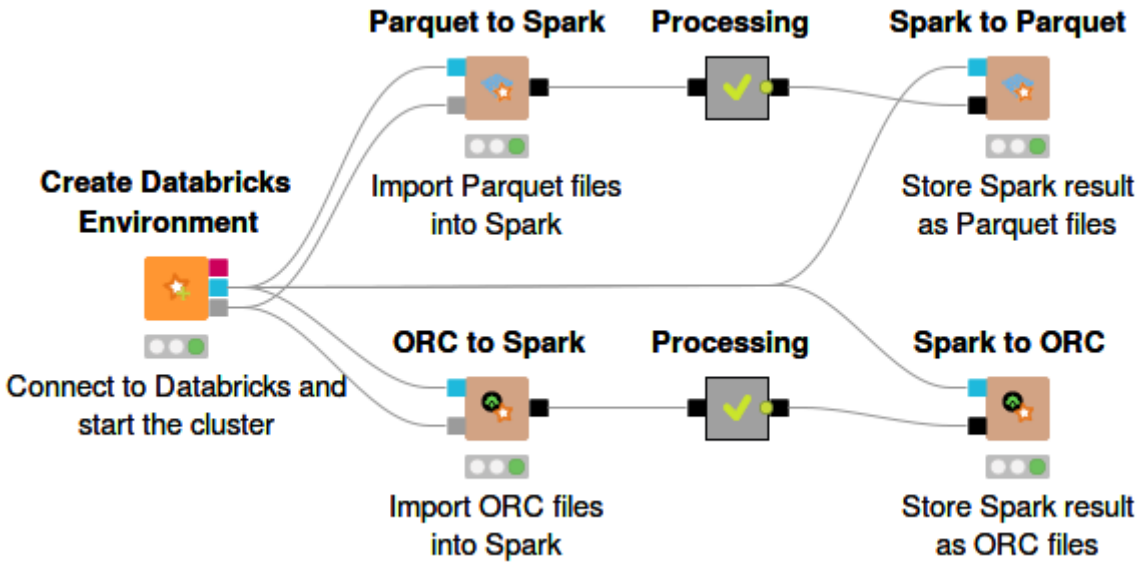

Figure 10. Parquet/ORC to Spark and Spark to Parquet/ORC node

1. To import Parquet files that are located in DBFS into a Spark DataFrame, use the Parquet to Spark node, then connect the input DBFS port (blue) and the input Spark port (grey) to the corresponding output ports of the Create Databricks Environment node (see Figure 10). In the node configuration dialog, simply enter the path to the folder where the Parquet files reside, and then execute the node.

   The Parquet data is now available in Spark and you can utilize any number of Spark nodes to perform further data processing visually.

   

   The ORC to Spark node has the same configuration dialog as the Parquet to Spark node.

2. To write a Spark DataFrame to DBFS in Parquet format, use the Spark to Parquet node. The node has two input ports. Connect the DBFS port to the DBFS port (blue) of the Create Databricks Environment node, and the second port to any node with a Spark data output port (black). To configure the Spark to Parquet node, open the node configuration dialog and:
   - Under *Target* folder, provide the path on DBFS to the destination folder where the Parquet file(s) should be created
   - *Target name* is the name of the folder that will be created and in which the Parquet file(s) will be stored
   - If the option *Overwrite result partition count* is enabled, the number of the output files can be specified. However, this option is strongly not recommended as this might lead to performance issues.
   - Under the *Partitions* tab there is an optional option whether the data should be partitioned based on specific column(s).

The Spark to ORC node has the same configuration dialog as the Spark to Parquet node.

Parquet/ORC Reader and Writer

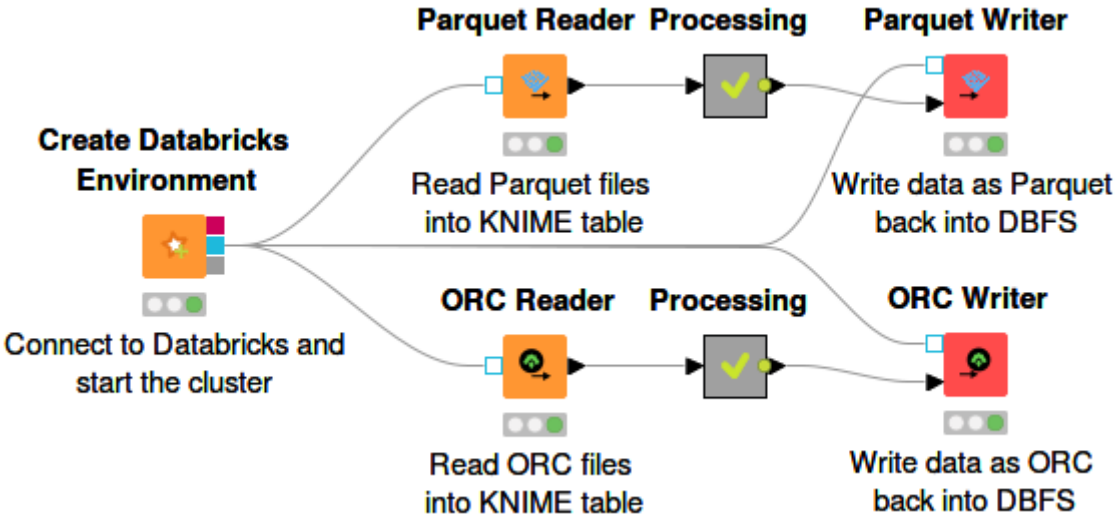

Figure 11. Parquet/ORC Reader and Writer nodes

1. To import data in Parquet format from DBFS directly into KNIME tables, use the Parquet Reader node. The node configuration dialog is simple, you just need to enter the DBFS path where the parquet file resides. Under the *Type Mapping* tab, the mapping from Parquet data types to KNIME types has to be specified.

   The Parquet data is now available locally and you can utilize any standard KNIME nodes to perform further data processing visually.

   

   The ORC Reader node has the same configuration dialog as the Parquet Reader node.

2. To write a KNIME table into a Parquet file on DBFS, use the Parquet Writer node. To connect to DBFS, please connect the DBFS (blue) port to the DBFS port of the Create Databricks Environment node. In the node configuration dialog, enter the location on DBFS where you want to write the Parquet file, and specify, under the *Type Mapping* tab, the mapping from KNIME data types to Parquet data types.

   

   The ORC Writer node has the same configuration dialog as the Parquet Writer node.

For more information on the *Type Mapping* tab, please check out the Database Documentation.

Databricks Database

Read and write from Databricks database

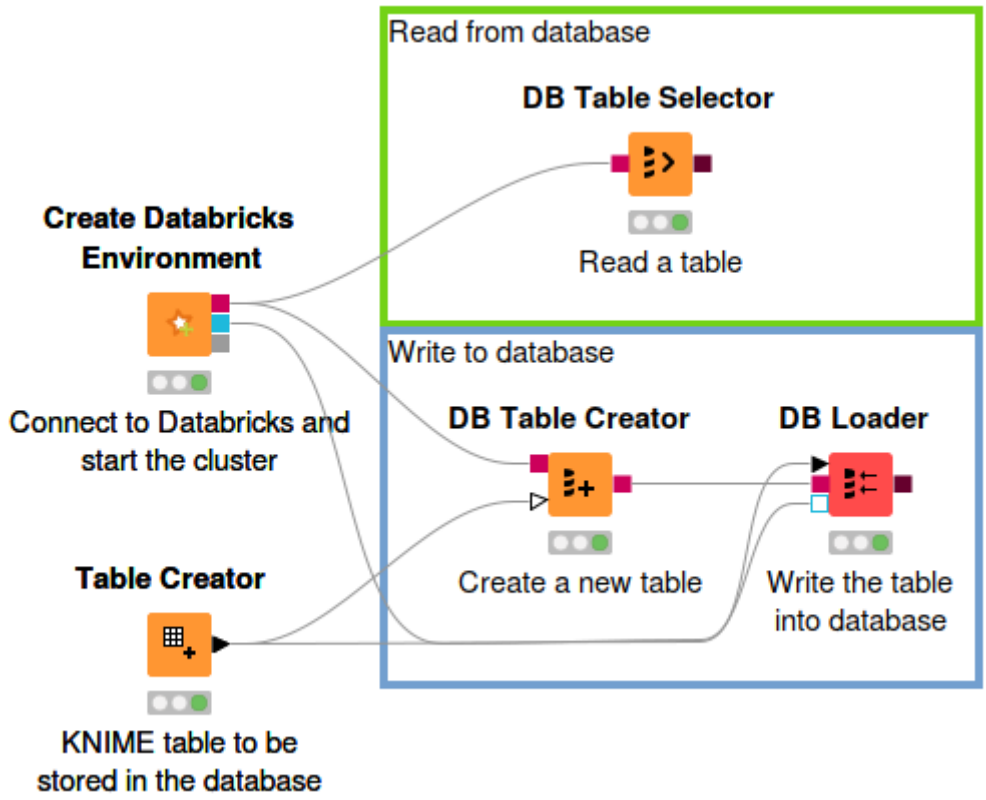

Figure 12. How to read and write from/to Databricks database

To store a KNIME table in Databricks database:

1. Use the DB Table Creator node. The node has two input ports. Connect the DB (red) port to the DB port of the Create Databricks Environment node, and the second port to the target KNIME table. In the node configuration dialog, enter the schema and the table name. Be careful when using special characters in the table name, e.g. underscore (_) is not supported. Executing this node will create an empty table in the database with the same table specification as the input KNIME table.

The DB Table Creator node offers many more funtionalities. For more information on the node, please check out the Database Documentation.

2. Append the DB Loader node to the DB Table Creator node. This node has three input ports. Connect the first port to the target KNIME table, the DB (red) port and the DBFS (blue) port to the DB port and DBFS port of the Create Databricks Environment node respectively. Executing this node loads the content of the KNIME table to the newly created table in the database.

> For more information on the DB Loader node, please check out the Database Documentation.

To read a table from a Databricks database, use the DB Table Selector node, where the input DB (red) port is connected to the DB port of the Create Databricks Environment node.

> For more information on other KNIME database nodes, please check out the Database Documentation.

Databricks Delta

Databricks Delta is a storage layer between the Databricks File System (DBFS) and Apache Spark API. It provides additional features, such as ACID transactions on Spark, schema enforcement, time travel, and many others.

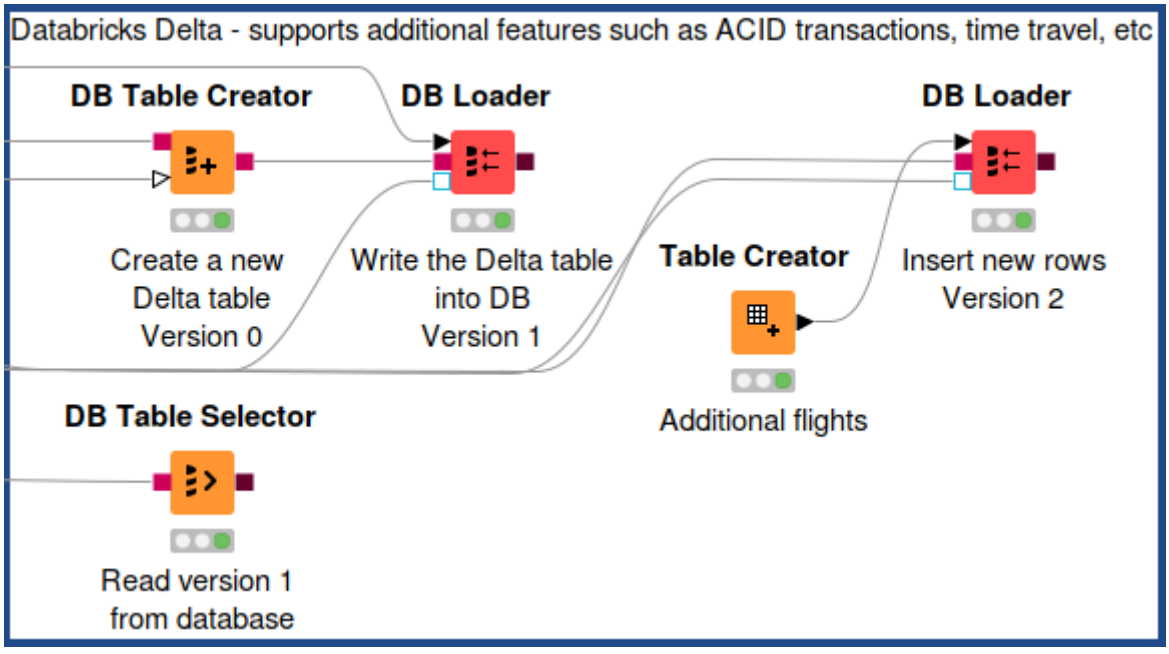

Figure 13. Databricks Delta on KNIME Analytics Platform

To create a Delta table in KNIME Analytics Platform using DB Table Creator node:

1. Connect the first port to the DB port (red) of the Create Databricks Environment node, and the second port to the target KNIME table

2. In the configuration dialog, enter the table name and schema as usual, and configure the other settings as according to your needs. To make this table become a Delta table, insert a `USING DELTA` statement under the *Additional Options* tab (see Figure 14).

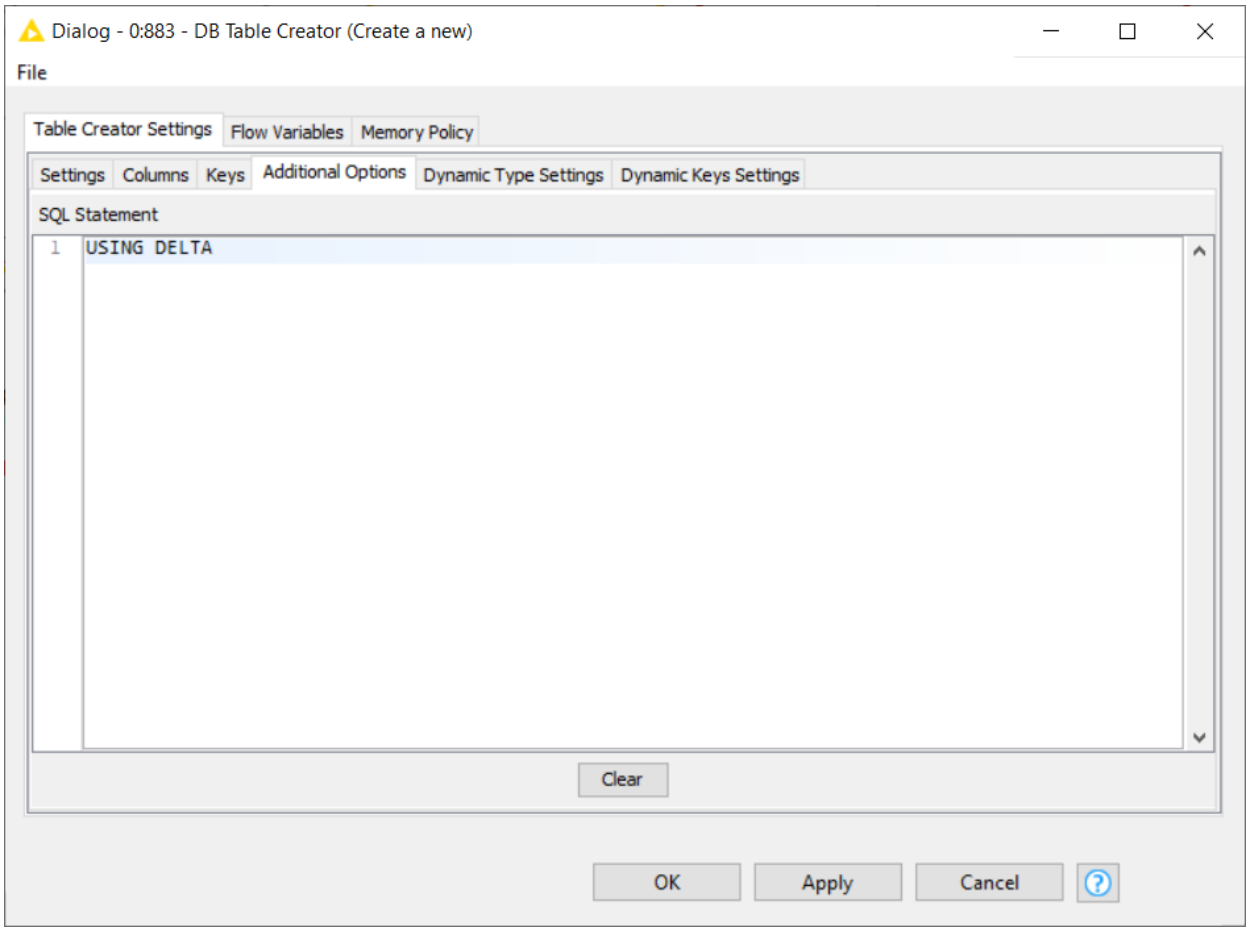

Figure 14. Additional Options tab inside the DB Table Creator node configuration dialog

3. Execute the node and an empty Delta table is created with the same table specification as the input KNIME table. Fill the table with data using e.g. the DB Loader node (see section Read and write from Databricks database).

Time Travel feature

Databricks Delta offers a lot of additional features to improve data reliability, such as time travel. Time travel is a data versioning capability allowing you to query an older snapshot of a Delta table (rollback).

To access the version history (metadata) in a Delta table on the Databricks web UI:

1. Navigate to the *Data* tab in the left pane
2. Select the database and the Delta table name
3. The metadata and a preview of the table is displayed. If the table is indeed a Delta table,

it will have an additional *History* tab beside the *Details* tab (see Figure 15).

4. Under the *History* tab, there is the versioning list of the table, along with the timestamps, operation types, and other information.

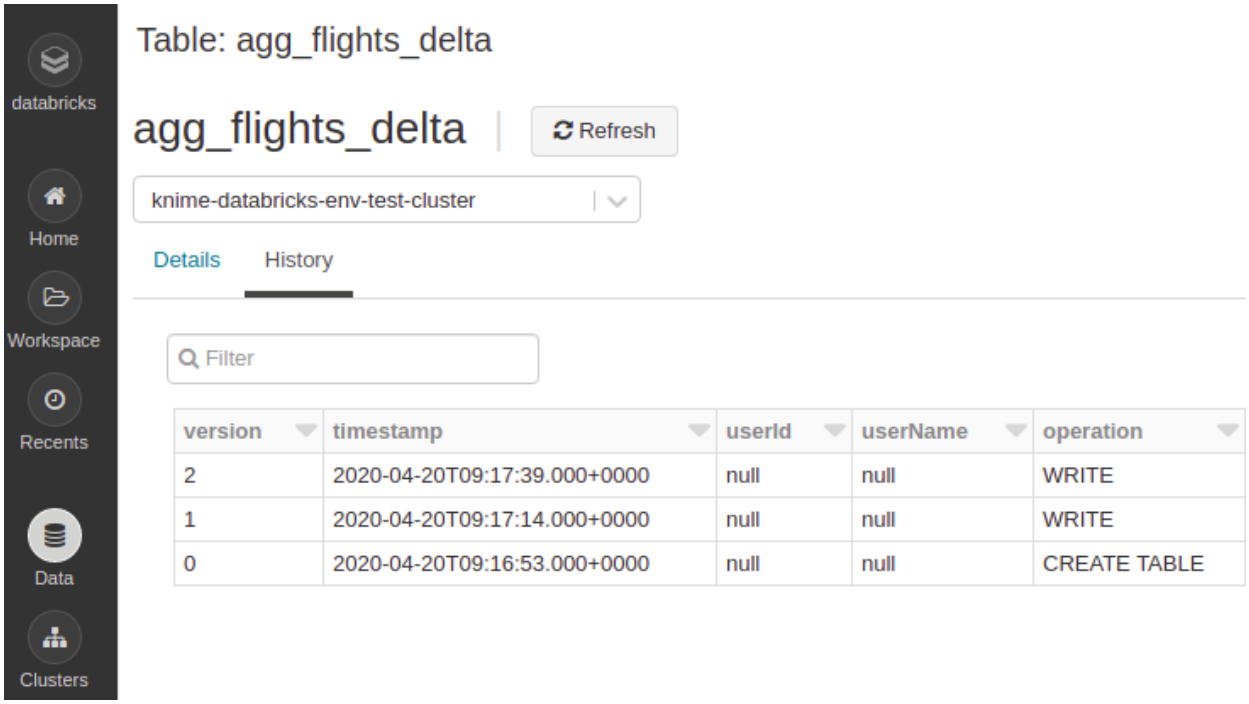

Figure 15. Delta table versioning history

Alternatively, you can also access the version history of a Delta table directly in KNIME Analytics Platform:

1. Use the DB Query Reader node. Connect the input DB port (red) of the DB Query Reader node to the DB port of the Create Databricks Environment node.
2. In the node configuration dialog, enter the following SQL statement:

```
DESCRIBE HISTORY <table_name>
```

   where `<table_name>` is the name of the table whose version history you want to access.

3. Execute the node. Then right click on the node, select *KNIME data table* to view the version history table (similar to the table in Figure 15).

For more information on Delta table metadata, please check the Databricks documentation.

Beside the version history, accessing older versions of a Delta table in KNIME Analytics Platform is also very simple:

1. Use a DB Table Selector node. Connect the input port with the DB port (red) of the Create Databricks Environment node.
2. In the configuration dialog, enter the schema and the Delta table name. Then enable the *Custom query* checkbox. A text area will appear where you can write your own SQL statement.
    a. To access older versions using version number, enter the following SQL statement:

    ```
    SELECT * FROM #table# VERSION AS OF <version_number>
    ```

    Where `<version_number>` is the version of the table you want to access. Check Figure 16 to see an example of a version number.

    b. To access older versions using timestamps, enter the following SQL statement:

    ```
    SELECT * FROM #table# TIMESTAMP AS OF <timestamp_expression>
    ```

    Where `<timestamp_expression>` is the timestamp format. To see the supported timestamp format, please check the Databricks documentation.
3. Execute the node. Then right click on the node, select *DB Data*, and click *Cache no. of rows* to view the table.

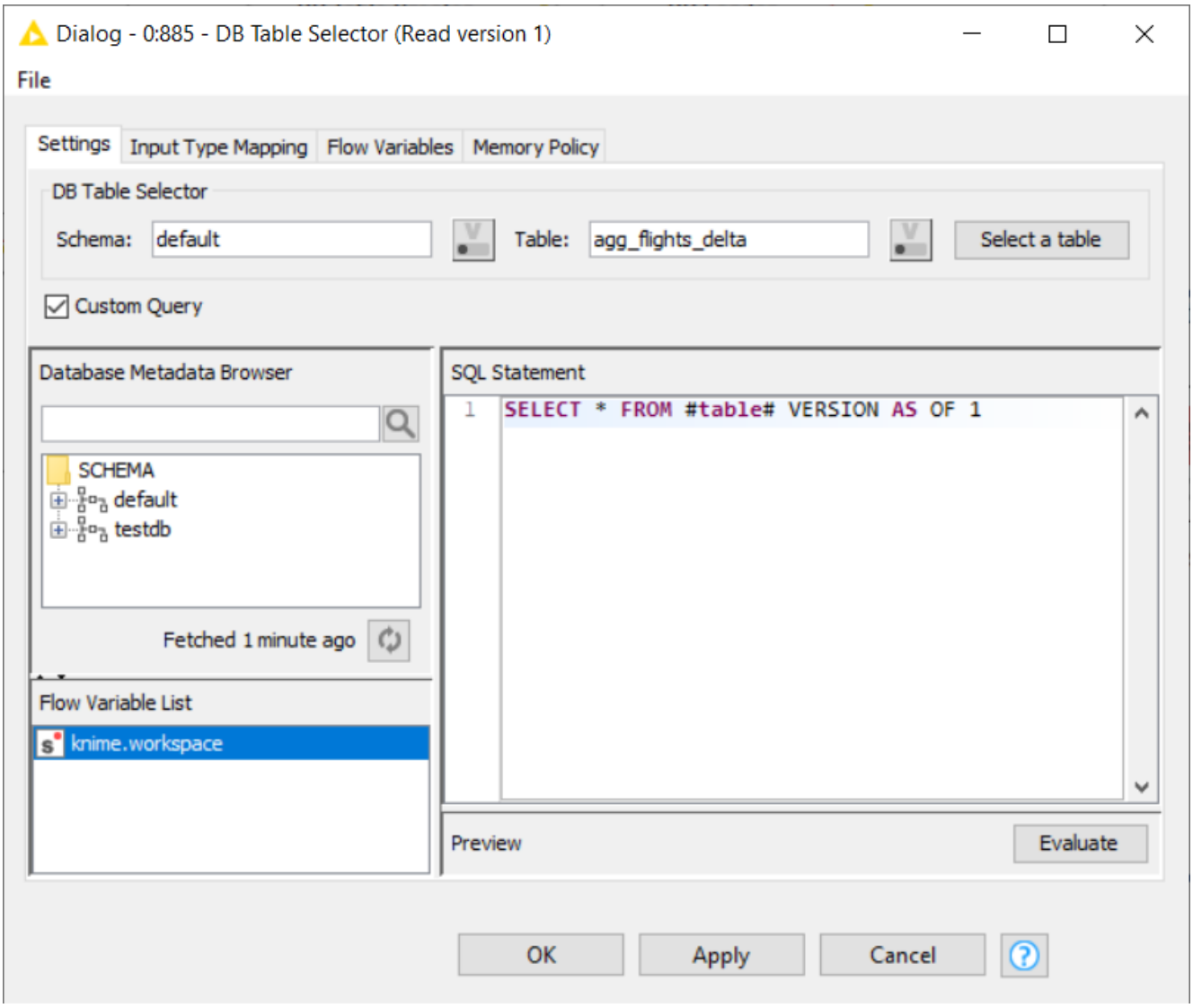

Figure 16. Configuration dialog of the DB Table Selector node

Spark to Hive / Hive to Spark

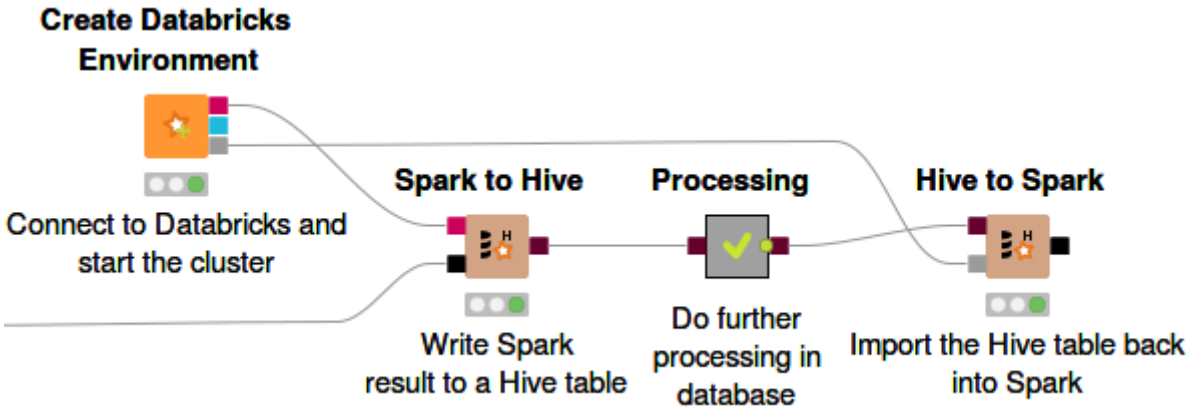

Figure 17. Example usage of Spark to Hive and Hive to Spark nodes

It is possible to store a Spark DataFrame directly in a Hive database with the Spark to Hive

node. The node has two input ports. Connect the DB port (red) to the DB port of the Create Databricks Environment node and the second Spark data port (black) to any node with a Spark data output port. This node is very useful to store Spark result permanently in a database.

On the other hand, the Hive to Spark node is used to import a Hive table back into a Spark DataFrame. The node has two input ports. Connect the Hive port (brown) to the target Hive table, and the Spark port (grey) to the Spark port of the Create Databricks Environment node.

Data preparation and analysis

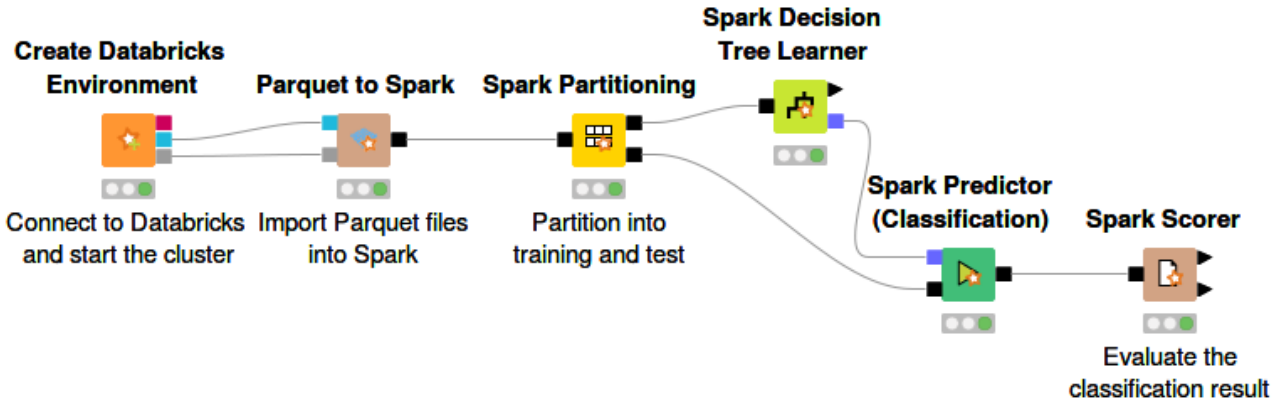

Figure 18. Example of machine learning application using Spark nodes on Databricks

The Databricks integration nodes blend seamlessly with the other KNIME nodes, which allows you to perform a variety of tasks on Databrick clusters via KNIME Analytics Platform, such as executing Spark jobs via the KNIME Spark nodes, while pushing down all the computation process into the Databricks cluster. Any data preprocessing and analysis can be done easily with the Spark nodes without the need to write a single line of code.

For advanced users, there is an option to use the scripting nodes to write custom Spark jobs, such as the PySpark Script nodes, Spark DataFrame Java Snippet nodes, or the Spark SQL Query node. These scripting nodes, in addition to the standard KNIME Spark nodes, allow for a more detailed control over the whole data science pipeline.

The scripting nodes are available under *Tools & Services > Apache Spark > Misc* in the node repository of the KNIME Analytics Platform.

For more information on the Spark nodes, please check out the KNIME Extension for Apache Spark product page.

An example workflow to demonstrate the usage of the Create Databricks Environment node to connect to a Databricks Cluster from within KNIME Analytics Platform is available on the

KNIME Hub.

KNIME AG
Talacker 50
8001 Zurich, Switzerland
www.knime.com
info@knime.com