Open for Innovation
KNIME®

KNIME Extension for Apache Spark User Guide

KNIME AG, Zurich, Switzerland
Version 4.0 (last updated on 2019-06-29)

Table of Contents

Introduction 1
Installation 2
Overview 3
Managing Spark contexts 4
    Create Spark Context (Livy) 4
    Create Spark Context node 6
    Proxy settings 7

Introduction

The KNIME Extensions for Apache Spark™ provides KNIME nodes to create workflows that execute on Apache Spark in a cluster environment that provides Spark itself. This document describes the usage of *KNIME Extension for Apache Spark* for users of KNIME Analytics Platform.

Before using *KNIME Extension for Apache Spark* with an existing cluster environment, additional installation steps are required on the **cluster-side**. If you are looking for information on this topic, please consult [TODO].

Installation

KNIME Extension for Apache Spark can be installed via the KNIME Update Manager:

1. Go to *File* > *Install KNIME Extensions*.
2. Open the category *KNIME Big Data Extensions*.
3. Select *KNIME Extension for Apache Spark*.
4. Click on *Next* and follow the subsequent dialog steps to install the extension.

Note that this will also implicitly install the *KNIME Big Data Connectors* extension, as some of the Spark nodes require access remote file systems, such as Apache Hadoop® File System (HDFS).

If you do not have direct internet access you can also install the extension from a local update site.

Overview

KNIME Extension for Apache Spark provides a set of nodes to offload a variety of operations to a Spark cluster, including

- managing and configuring Spark contexts,
- reading/writing data from/to HDFS, Hive, Impala, Amazon S3 or Azure Blob Storage in file formats such as Parquet or ORC,
- transforming and preparing of data, such as sampling, partitioning, joining, grouping or pivoting,
- conducting predictive analytics and scoring with Apache Spark MLlib,
- running Java or Python code snippets that use the Spark/PySpark APIs.

All nodes are interactive and provide data preview, and it is possible to mix-and-match them with nodes from other KNIME extensions.

Example: Figure [kmeansexample] shows an example of how to do a k-Means clustering in Apache Spark with data from Apache Hive.

[030 trainkmeans] | *030_trainkmeans.png*

The first step in any Spark workflow is to create a Spark context. The *Create Spark Context (Livy)* node connects to an Apache Livy server — running on the cluster — and creates a Spark context there. See also Managing Spark contexts for more information on how to work with Spark contexts.

Then the *Hive to Spark* node imports the results of a Hive SQL query into the Spark context, keeping the column schema information. Data in Spark is stored in a distributed fashion in your Spark cluster and is internally represented as DataFrames (Spark 2) or RDDs (Spark 1). You can interactively view such data in the output port, as in other KNIME nodes.

Afterwards, the *Spark Partitioning* node splits the DataFrame into training and test data. The training set flows into the *Spark k-Means* node that trains a clustering model (using Apache Spark's MLlib) on the data and hands it to the *Spark Cluster Assigner* node. This node uses the model to label the previously unseen test data.

Finally, the *Spark to Hive* node stores the labeled data back into a Hive table. It is also possible to import the labeled test data into your local KNIME Analytics Platform using the *Spark to Table* node.

Managing Spark contexts

The first step in any Spark workflow is to create a Spark context, which represents represents the connection to a Spark cluster. Creating a Spark context reserves resources (CPU cores and memory) in your cluster environment to be exlusively used by your KNIME workflow. Hence, a Spark context should be created at the beginning of a KNIME workflow and destroyed at the end, in order to release the resources.

```
There are several ways to create a Spark context with KNIME nodes:
```

- Create Spark Context (Livy) (recommended)
- Create Spark Context node (legacy)
- *Create Local Big Data Environment* (requires installation of the KNIME Extension for Local Big Data Environments)

INFO: The *Create Local Big Data Environment* node allows you to create a fully functional local big data environment including Spark, Hive and HDFS. It allows you to try out the nodes of the KNIME Big Data Extensions without a Hadoop cluster.

Create Spark Context (Livy)

The *Create Spark Context (Livy)* node connects to an Apache Livy server to create a new Spark context.

[create spark context livy] | *create_spark_context_livy.png*

Requirements

- **Apache Livy service** Livy needs to be installed as a service in your cluster. Please consult the [TODO] for more details.
- **Network Connectivity:** The node needs to be able to make a network connection to the Livy service in your cluster (default port TCP/8998). Currently, only proxies that do not require any authentication are supported. Note that KNIME does not provide the proxy software itself.
- **Authentication**: If the Livy server you are connecting to requires Kerberos authentication, then KNIME Analytics Platform needs to be set up accordingly. Please follow the steps outlined in the Speaking Kerberos with KNIME Big Data Extensions blog post.

- **Remote file system**: The node requires access to a remote file system to exchange temporary files between KNIME Analytics Platform and the Spark context (running on the cluster). Supported file systems are:
  - HDFS, webHDFS and httpFS. Note that the node must access the remote file system with the same user as the Spark context, otherwise Spark context creation fails. When authenticating with Kerberos against both HDFS/webHDFs/httpFS and Livy, then the same user will be used. Otherwise, this must be ensured manually.
  - Amazon S3 and Azure Blob Store, which is recommended when using Spark on Amazon EMR/Azure HDInsight. Note that for these file systems a staging area must be specified in the **Advanced** tab of the *Create Spark Context (Livy)* node.

Node dialog

[create spark context livy tab1] | *create_spark_context_livy_tab1.png*

The node dialog has two tabs. The first tab provides the most commonly used settings when working with Spark:

1. **Spark version:** Please choose the Spark version of the Hadoop cluster you are connecting to.
2. **Livy URL:** The URL of Livy including protocol and port e.g. `http://localhost:8998/`.
3. **Authentication:** How to authenticate against Livy. Supported mechanism are Kerberos and None.
4. **Spark Executor resources:** Sets the resources to be request for the Spark executors. If enabled, you can specify the amount of memory and the number of cores for each executor. In addition you can specify the Spark executor allocation strategy.
5. **Estimated resources:** An estimation of the resources that are allocated in your cluster by the Spark context. The calculation uses default settings for memory overheads etc and is thus only an estimate. The exact resources might be different depending on your specific cluster settings.

[create spark context livy tab2] | *create_spark_context_livy_tab2.png*

The second tab provides the advanced settings that are sometimes useful when working with Spark:

1. **Override default Spark driver resources:** If enabled, you can specify the amount of memory and number of cores to be allocated for the Spark driver process.

2. **Set staging area for Spark jobs:** If enabled, you can specify a directory in the connected remote file system, that will be used to transfer temporary files between KNIME and the Spark context. If no directory is set, then a default directory will be chosen, e.g. the HDFS user home directory. However, if the remote file system is Amazon S3 or Azure Blob Store, then a staging directory must be provided.
3. **Set custom Spark settings:** If enabled, you can specify additional Spark settings. A tooltip is provided for the keys if available. For further information about the Spark settings refer to the Spark documentation.

Create Spark Context node

This node connects to Spark Job Server to create a new Spark context.

[create spark context] | *create_spark_context.png*

Its node dialog has two main tabs. The first tab is the Context Settings tab which allows you to specify the following Spark Context settings:

1. **Spark version:** Please choose the Spark version of the Hadoop cluster you are connecting to.
2. **Context name:** A unique name for the Spark context.
3. **Delete objects on dispose:** KNIME workflows with Spark nodes create objects such as DataFrames/RDDs during execution. This setting specifies whether those objects shall be deleted when closing a workflow.
4. **Override Spark settings:** Custom settings for the Spark context, e.g. the amount of memory to allocate per Spark executor. These settings override those from Job Server's `environment.conf`.
5. **Hide context exists warning:** If not enabled the node will show a warning if a Spark Context with the defined name already exists.

[create spark context tab1] | *create_spark_context_tab1.png*

Figure 1. Create Spark Context: Context Settings tab

The second tab is the **Connection Settings** tab which allows you to specify the following connection settings:

1. **Job server URL**: This is the HTTP/HTTPS URL under which the Spark Job Server WebUI can be reached. The default URL is `http://localhost:8090/`.
2. **Credentials:** If you have activated user authentication, you need to enter a username and password here.

3. **Job timeout in seconds/Job check frequency:** These settings specify how long a single Spark job can run before being considered failed, and, respectively, in which intervals the status of a job shall be polled.

[create spark context tab2] | *create_spark_context_tab2.png*

Figure 2. Create Spark Context: Connection Settings tab

Destroy Spark Context node

Once you have finished your Spark job, you should destroy the created context to free up the resources your Spark Context has allocated on the cluster. To do so you can use the **Destroy Spark Context** node.

[destroy spark context] | *destroy_spark_context.png*

Figure 3. The Destroy Spark Context node

[simple spark workflow example] | *simple_spark_workflow_example.png*

Figure 4. Simple example of a Spark workflow

Adapting default settings for the Create Spark Context node

The default settings of the Create Spark Context node can be specified via a preference page. The default settings are applied whenever the node is added to a KNIME workflow. To change the default settings, open **File** > **Preferences** > **KNIME** > **Big Data** > **Spark** and adapt them to your environment (see [knime_ext_create_spark_context]).

Proxy settings

If your network requires you to connect to Spark Job Server via a proxy, please open **File** > **Preferences** > **Network Connections**. Here you can configure the details of your HTTP/HTTPS/SOCKS proxies. Please consult the official Eclipse documentation on how to configure proxies.

KNIME AG
Technoparkstrasse 1
8005 Zurich, Switzerland
www.knime.com
info@knime.com