Open for Innovation
KNIME

KNIME Big Data Extensions User Guide

KNIME AG, Zurich, Switzerland
Version 4.0 (last updated on 2020-09-02)

Table of Contents

Overview

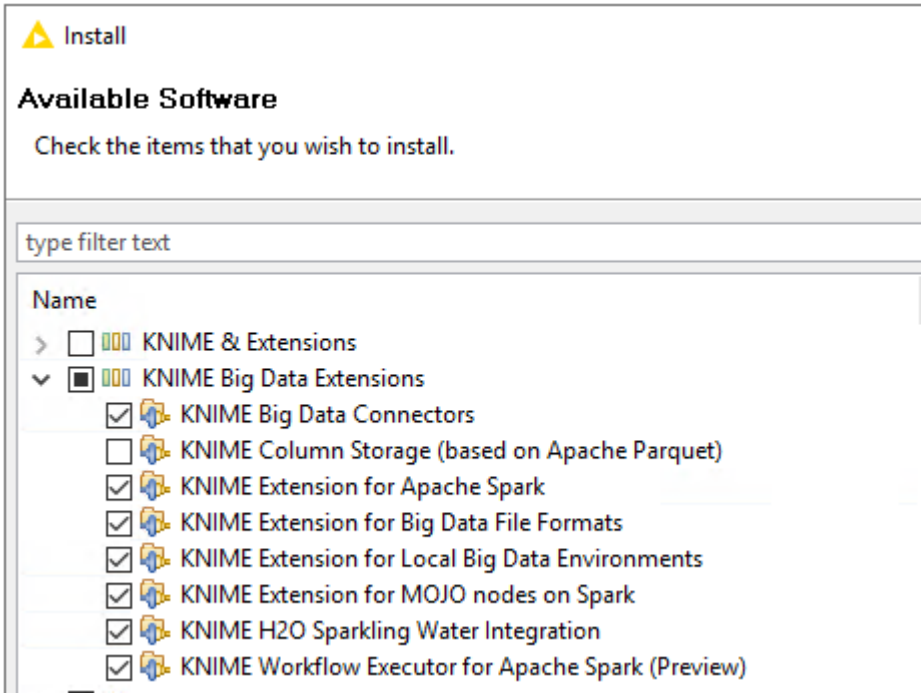
KNIME Big Data Extensions integrate Apache Spark and the Apache Hadoop ecosystem with KNIME Analytics Platform.

This guide is aimed at users of KNIME Analytics Platform who want to build workflows that need to access, process and analyze large amounts of data in a big data environment.

Note that additional installation and configuration steps may be necessary in your big data environment. Please consult the Big Data Extensions Admin Guide for details.

Installation

Navigate to *File → Install KNIME Extensions* and open the *KNIME Big Data Extensions* category. Check the boxes of those extensions that you wish to install.



KNIME Big Data Extensions are a set of several extensions that build on each other:

- *KNIME Big Data Connectors* provide connector nodes to read/write files in HDFS and query Hive and Impala with SQL.
- *KNIME Extension for Big Data File Formats* allows to read/write popular file formats such as Parquet in HDFS, Amazon S3 and Azure Blob Store.
- *KNIME Extension for Apache Spark* provides over 60 nodes for data access and wrangling, as well as predictive analytics in Spark. The following extensions add even more functionality around Spark:

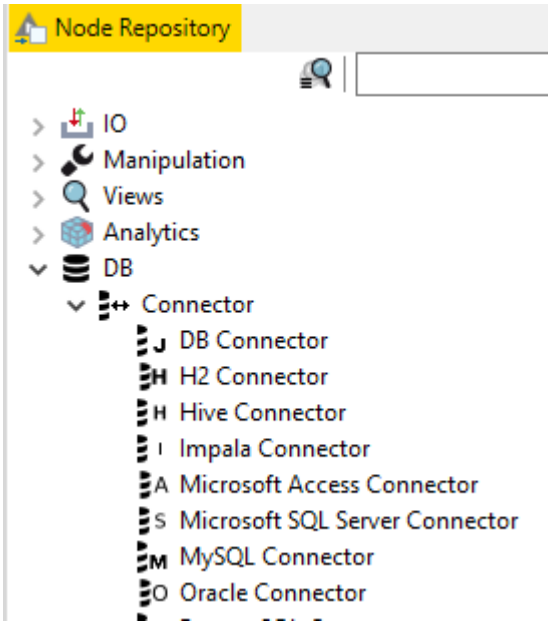
- *KNIME Extension for Local Big Data Environments* provides a node to create a completely local big data environment with Spark and Hive, without any additional configuration or software installation.
- *KNIME H2O Sparkling Water Integration* integrates the the KNIME H2O nodes with Spark to learn H2O models on data in Spark.
- *KNIME Extension for MOJO nodes on Spark* provides to nodes to do prediction with H2O MOJOs in Spark.
- *KNIME Workflow Executor for Apache Spark* allows to execute non-Spark KNIME nodes on Apache.

Once you have installed the extension(s), restart KNIME Analytics Platform.

If you don't have direct internet access you can also install the extensions from a zipped update site. Follow the steps outlined in Adding Local Update Sites.

Hive and Impala

The *KNIME Big Data Connectors* extension provides nodes to connect to Hive and Impala.



Hive Connector

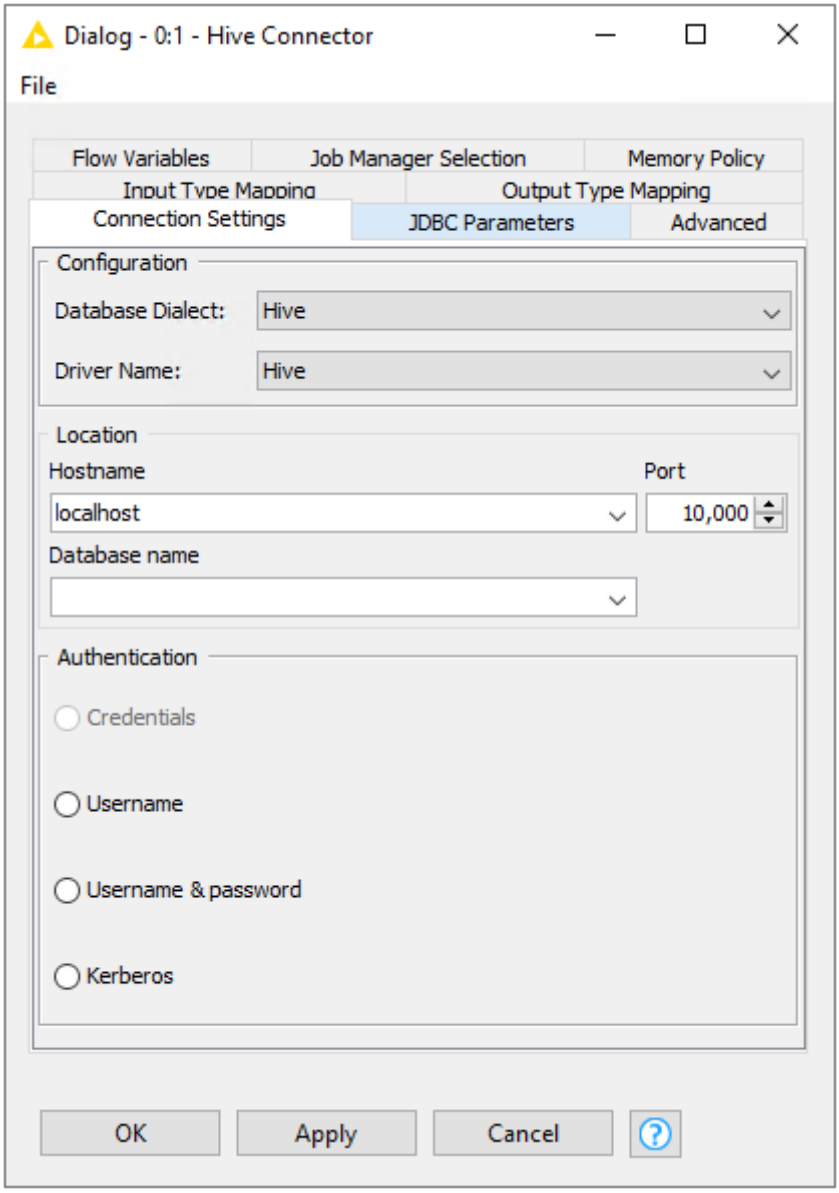


Figure 1. Hive Connector configuration dialog

The *Hive Connector* node creates a connection to Hive via JDBC. You need to provide the following information:

- the hostname (or IP address) of the server
- the port
- a database name.
- An authentication method (as required by Hive):
  - *Credentials* where username and password are be supplied via credentials flow variable (see *Credentials Input* node).
  - *Username* where the username is supplied in the dialog.
  - *Username & Password* where username and password are supplied in the dialog.

- Kerberos where authentication is performed via Kerberos.

When using Kerberos authentication: Additional parameters need to be provided in the JDBC Parameters tab. The exact parameters depend on the JDBC driver selected in the *Driver Name* setting.

The built-in driver with name "Hive" requires the following parameters for Kerberos:

- `kerberosAuthType=fromSubject`
- `principal=hive/<hostqdn>@<REALM>`, where
  - `<hostqdn>` is the fully qualified hostname of the Hive service
  - `<REALM>` is the Kerberos realm of the Hive service

Proprietary Hive drivers (e.g. provided by Cloudera) require the following parameters:

- `AuthMech=1`
- `KrbServiceName=hive`
- `KrbHostFQDN=<hostqdn>`, where `<hostqdn>` is the fully qualified hostname of the Hive service
- `KrbRealm=<REALM>`, where `<REALM>` is the Kerberos realm of the Hive service

Please note the proprietary drivers need to registered first as described in Register your own JDBC drivers (KNIME Database Extension Guide).

Impala Connector

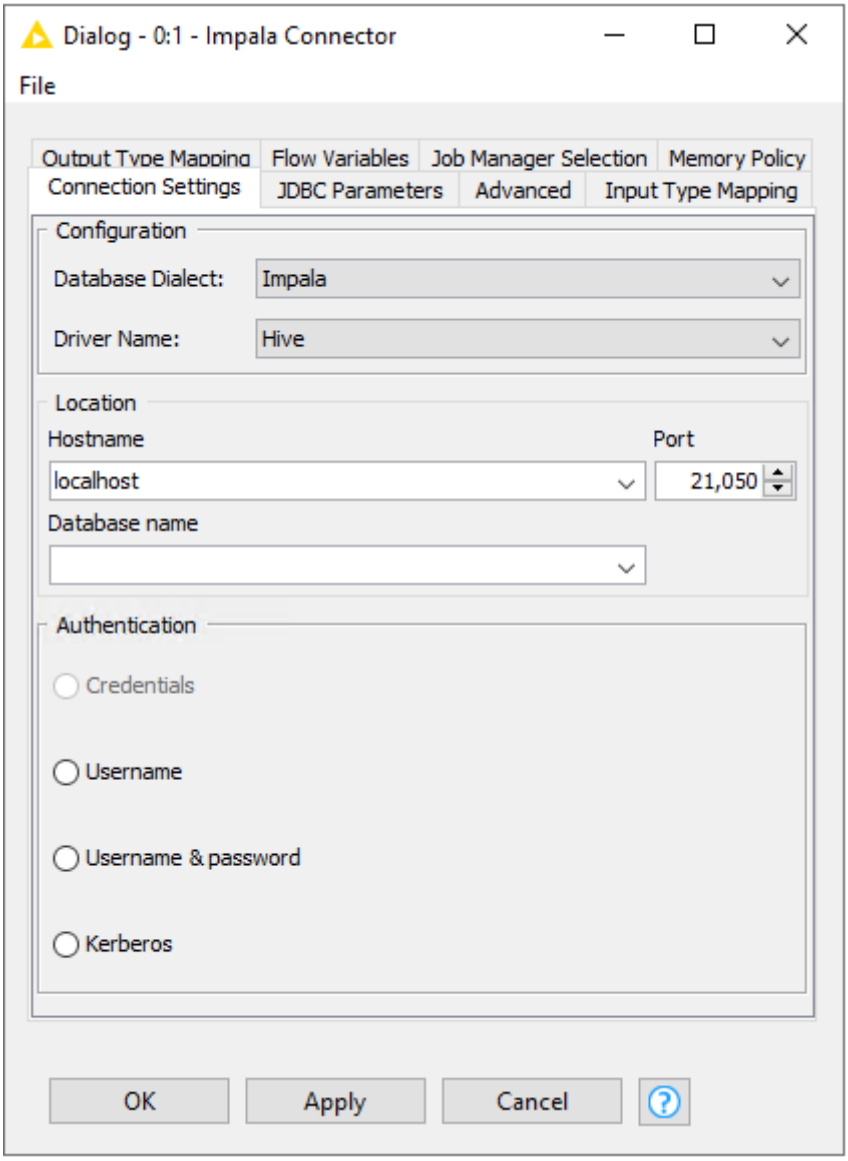


Figure 2. Impala Connector configuration dialog

The *Impala Connector* node creates a connection to Impala via JDBC. You need to provide the following information:

- the hostname (or IP address) of the Impala service
- the port
- a database name.
- An authentication method (as required by Impala):
  - *Credentials* where username and password are be supplied via credentials flow variable (see *Credentials Input* node).
  - *Username* where the username is supplied in the dialog.
  - *Username & Password* where username and password are supplied in the dialog.
  - Kerberos where authentication is performed via Kerberos.

When using Kerberos authentication: Additional parameters need to be provided in the JDBC Parameters tab. The exact parameters depend on the JDBC driver selected in the *Driver Name* setting.

The built-in driver with name "Hive" requires the following parameters for Kerberos:

- `kerberosAuthType=fromSubject`
- `principal=impala/<hostqdn>@<REALM>`, where
  - `<hostqdn>` is the fully qualified hostname of the Hive service
  - `<REALM>` is the Kerberos realm of the Hive service

Proprietary Hive drivers (e.g. provided by Cloudera) require the following parameters:

- `AuthMech=1`
- `KrbServiceName=impala`
- `KrbHostFQDN=<hostqdn>`, where `<hostqdn>` is the fully qualified hostname of the Hive service
- `KrbRealm=<REALM>`, where `<REALM>` is the Kerberos realm of the Hive service

Bulk data loading

The *DB Loader* node supports bulk loading of data from KNIME Analytics Platform into a Hive or Impala table. Note that the database table needs to exist prior to executing the *DB Loader* node. The example below uses the *DB Table Creator* node to create the table prior to loading the data into the table, but this is not necessary if the table does already exist.

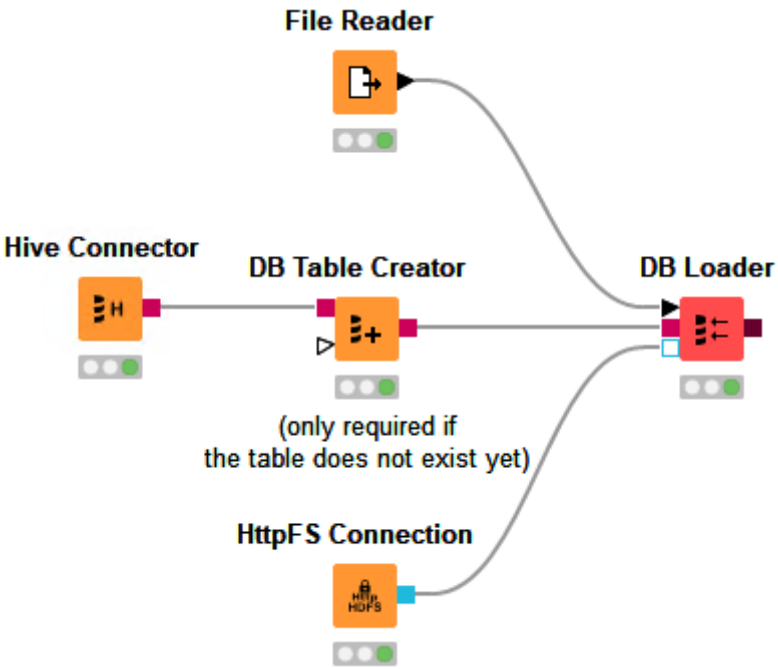
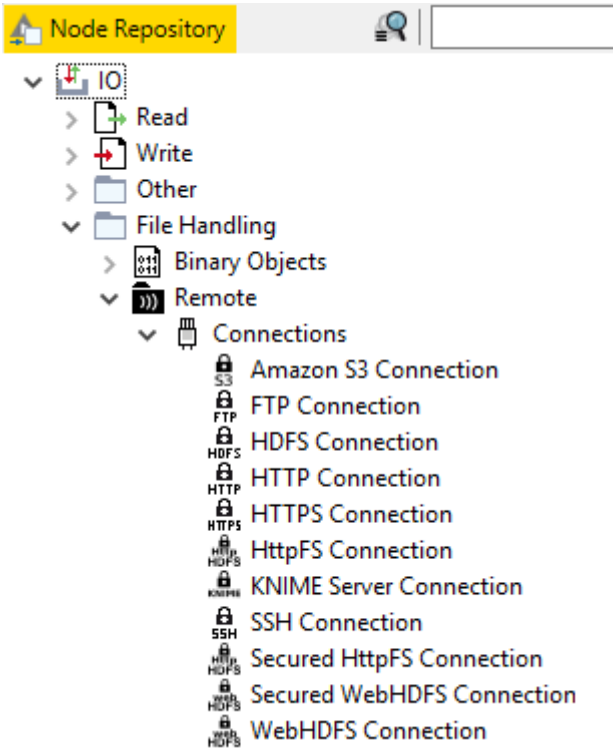


Figure 3. Workflow that creates a Hive table and then loads data into it.

HDFS

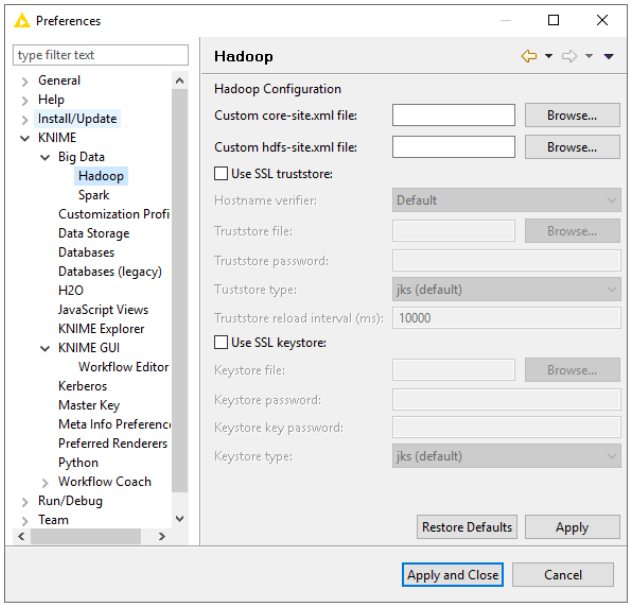


The *KNIME Big Data Connectors* extension provides several nodes to connect to HDFS:

- *HDFS Connection* directly communicates with HDFS during data transfer, i.e. the NameNode and all DataNodes. This node requires **direct network connectivity (no proxies, no firewall)** between KNIME Analytics Platform/KNIME Server and the Hadoop cluster, which is often not the case. Restricted network connectivity typically results in timout errors during data transfer.
- *WebHDFS Connection* uses HTTP to directly connect to HDFS, i.e. the NameNode and all DataNodes. It is possible to connect through a HTTP proxy, but still all cluster nodes need to be reachable through the proxy.
- *Secured WebHDFS Connection* uses HTTPS (SSL encrypted) to directly connect to HDFS.
- *httpFS Connection* (**recommended**) uses HTTP to connect to a httpFS service on a cluster frontend/edge node. The httpFS service serves as an intermediary between the internal cluster network and KNIME Analytics Platform/KNIME Server.
- *httpFS Connection* (**recommended**) uses HTTPS (SSL encrypted) to connect to a httpFS service on a cluster frontend/edge node.

The above connector nodes can be used together with the *KNIME File Handling Nodes* extension to upload, download or list files and perform other file operations.

Please consult the example workflows, which are available in KNIME Analytics Platform when connecting to the



The Hadoop preferences allow to specify typical Hadoop configuration files (`core-site.xml` and `hdfs-site.xml`) if necessary.

Furthermore, an SSL trustore can be configured if your HDFS services are using SSL encryption with corporate or self-signed SSL certificates. A keystore can also be defined if required by your Hadoop cluster.

Spark

KNIME Extension for Apache Spark provides a set of over 60 nodes nodes to create and execute Apache Spark applications.

Figure 4. All Spark nodes

The first step in any Spark workflow is to create a Spark context, which represents the connection to a Spark cluster. The Spark context also reserves resources (CPU cores and memory) in your cluster to be exclusively used by your workflow. Hence, a Spark context

should be created at the beginning of a workflow and destroyed at the end, in order to release the resources.

There are several nodes to create a Spark context:

- Create Spark Context (Livy) (recommended)
- Create Spark Context (Jobserver) (deprecated)
- *Create Local Big Data Environment* (requires *KNIME Extension for Local Big Data Environments*)

Create Spark Context (Livy)

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

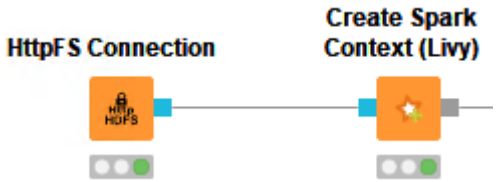


Figure 5. Create Spark Context (Livy) node

Requirements

- **Apache Livy service** Livy needs to be installed as a service in your cluster. Please consult the Apache Livy setup section for more details.
- **Network Connectivity:** The node initiates a HTTP(S) connection to Livy (default port TCP/8998). Currently, only HTTP(S) proxies that do not require any authentication are supported.
- **Authentication**: If Livy requires Kerberos authentication, then KNIME Analytics Platform needs to be set up accordingly.
- **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. 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

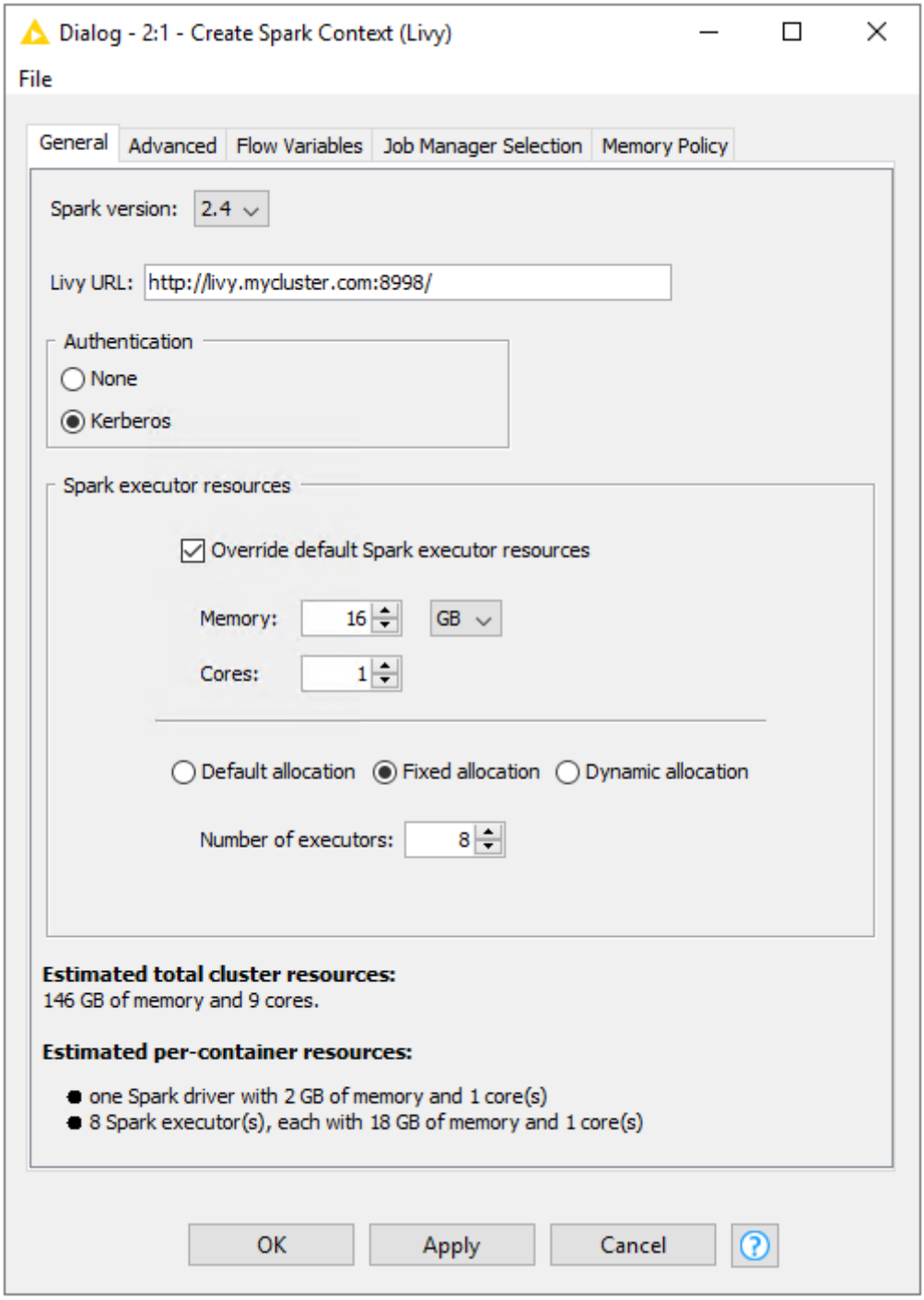


Figure 6. Create Spark Context (Livy): General settings tab

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.

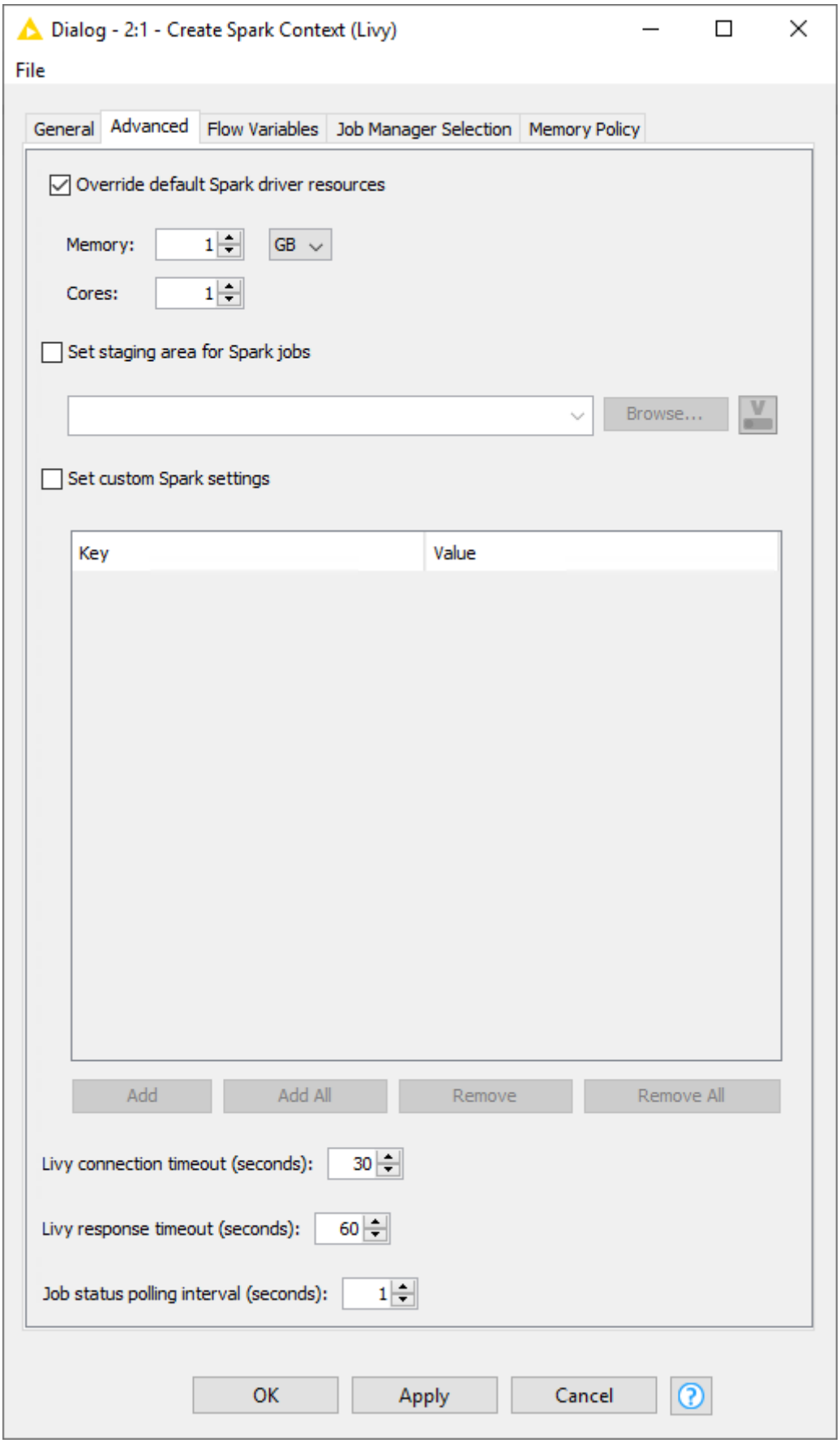


Figure 7. Create Spark Context (Livy): Advanced settings tab

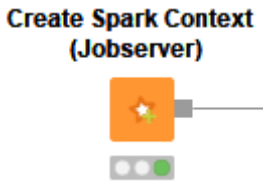
The second tab provides the advanced settings that are sometimes useful when working

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 (Jobserver)

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



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.

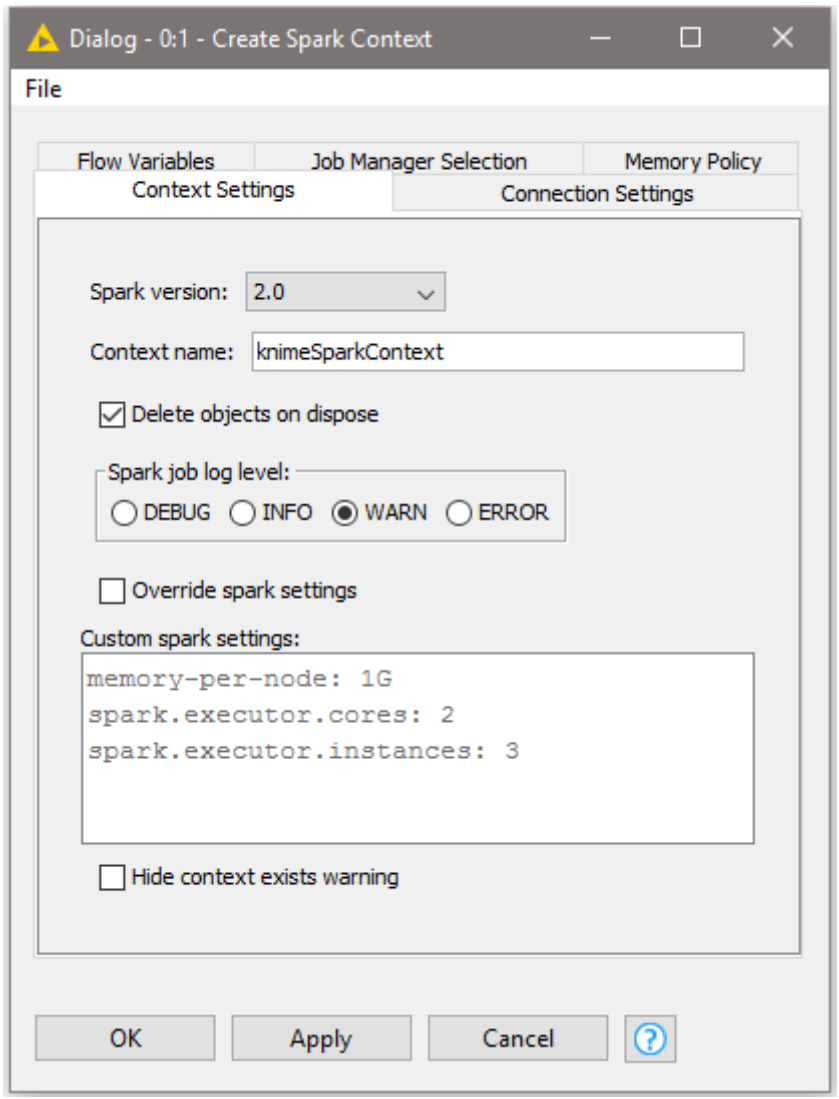


Figure 8. Create Spark Context: Context Settings tab

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.

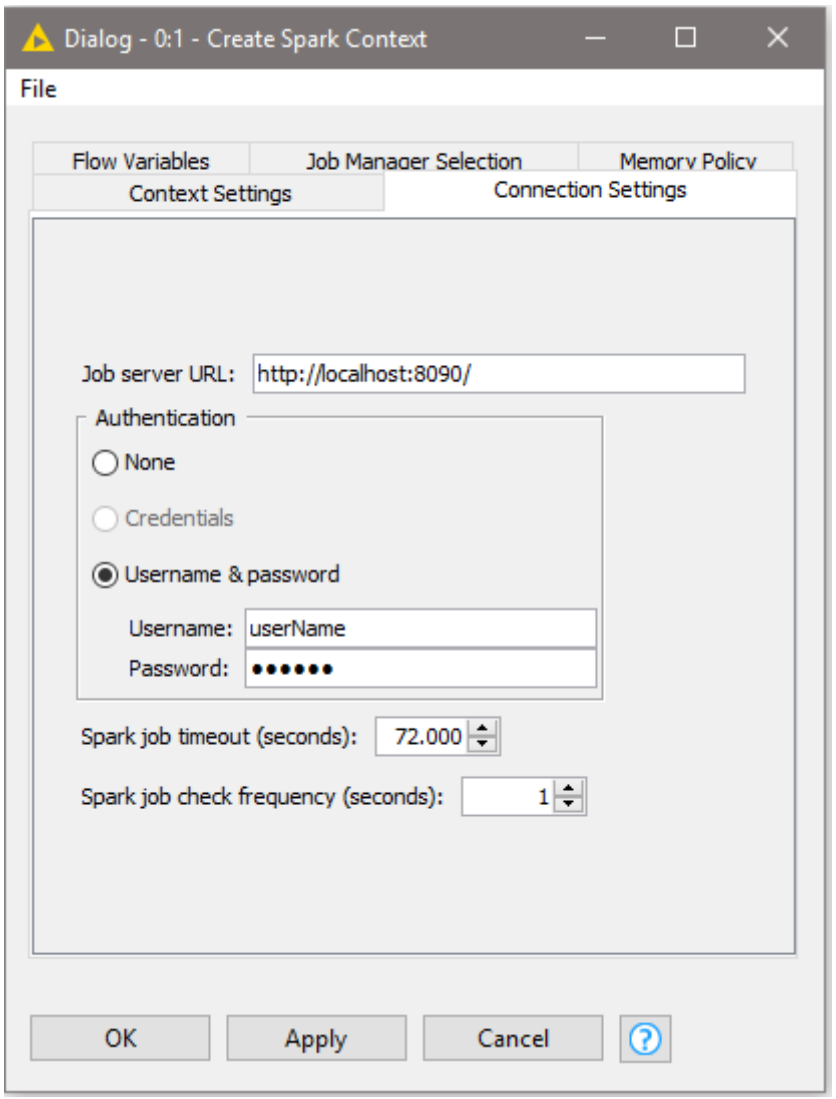


Figure 9. Create Spark Context: Connection Settings tab

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]).

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.

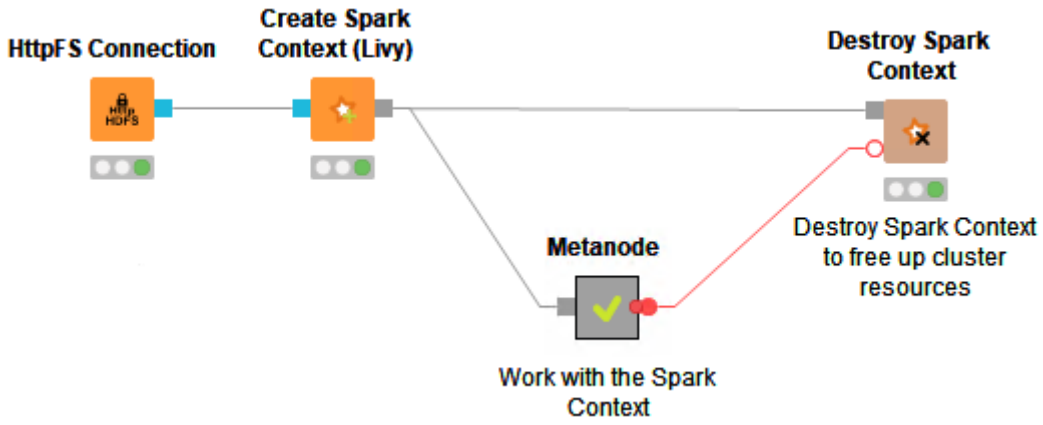
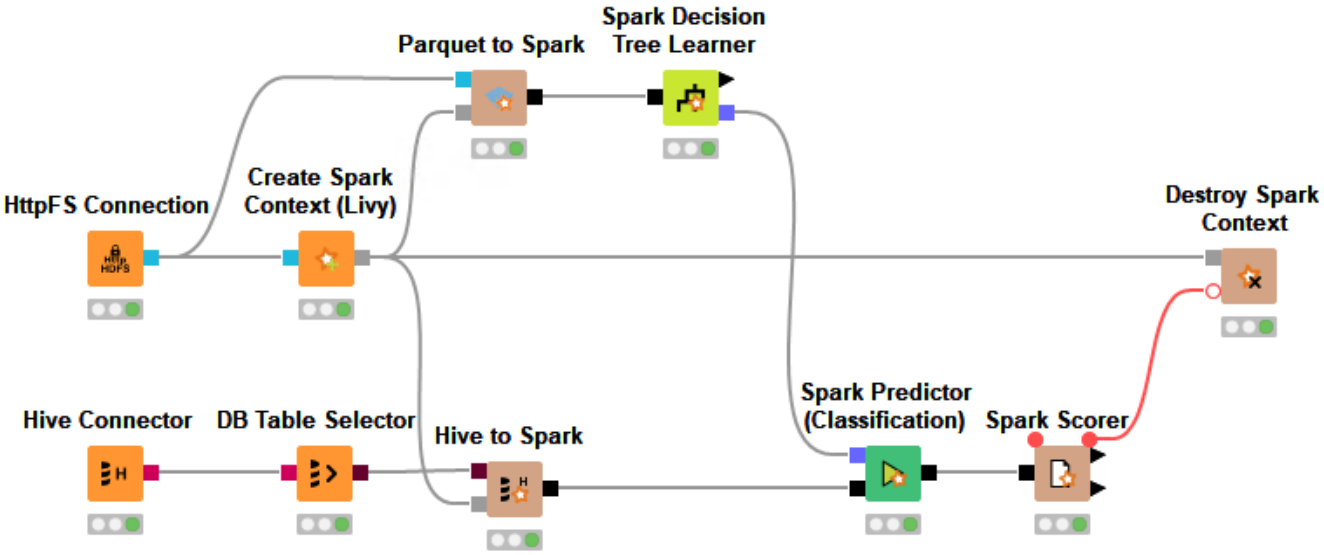


Figure 10. How to use the Destroy Spark Context node

Proxy settings

If your network requires you to connect to Livy or 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.

Example workflow



The above example workflow first creates a Spark context (*Create Spark Context (Livy)*) and then reads training data from a Parquet file stored in HDFS (*Parquet to Spark*). It then trains a decision tree model (*Spark Decision Tree Learner*) on that data. Additionally, it reads a Hive table with test data into Spark (*Hive to Spark*), uses the previously learned model to perform predictions (*Spark Predictor*), and determines the accuracy of the model given the test data (*Spark Scorer*).

For more examples consult the example workflows, which are available in KNIME Analytics Platform in the KNIME Explorer View, when connecting to the EXAMPLES server. Under *01_Big_Data → 02_Spark_Executor* you will find a variety of example workflows that demonstrate how to use the Spark nodes.

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