KNIME® Spark Executor

Installation Guide

Version 1.0.2

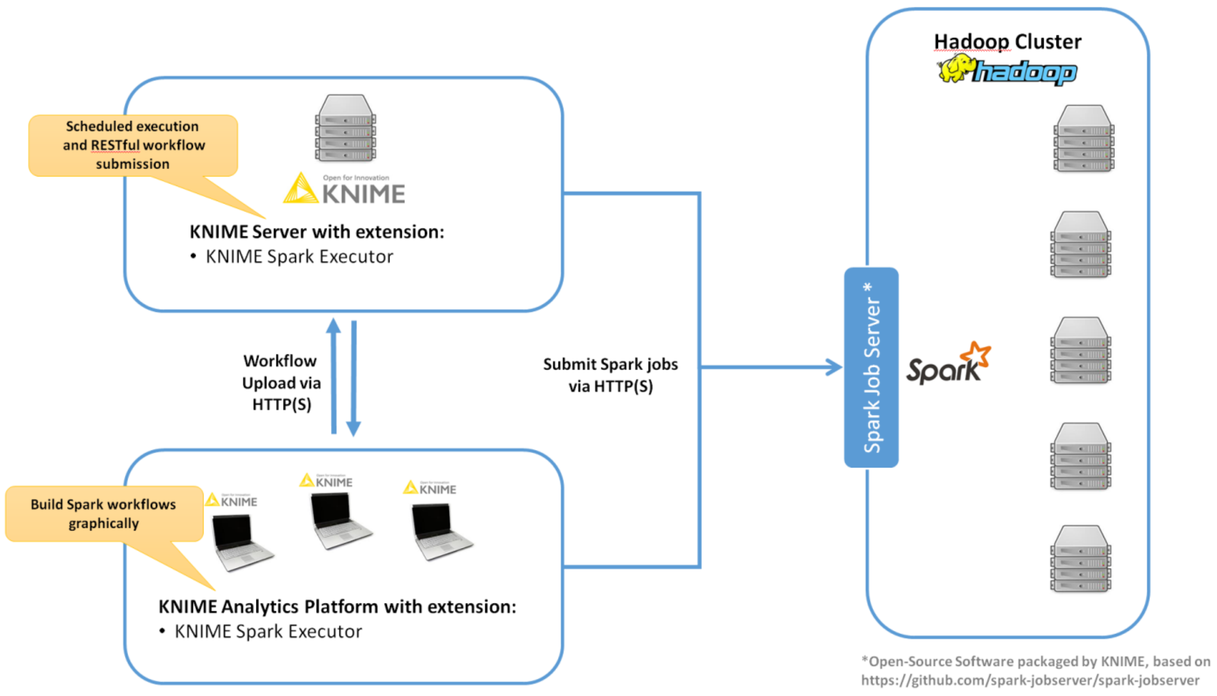
TABLE OF CONTENTS

INTRODUCTION

This document describes the installation procedure of the KNIME® Spark Executor to be used with KNIME® Analytics Platform 2.12 and KNIME® Server 4.1.

As depicted below, the KNIME Spark Executor consists of ...

- an extension for KNIME Analytics Platform/KNIME Server
- a service called Spark job server, that needs to installed on an edge node of your Hadoop cluster.


Scheduled execution and RESTful workflow submission
KNIME
KNIME Server with extension:
• KNIME Spark Executor
Workflow Upload via HTTP(S)
Submit Spark jobs via HTTP(S)
Build Spark workflows graphically
KNIME Analytics Platform with extension:
• KNIME Spark Executor
Hadoop Cluster
hadoop
Spark Job Server *
Spark
*Open-Source Software packaged by KNIME, based on https://github.com/spark-jobserver/spark-jobserver

SUPPORTED HADOOP DISTRIBUTIONS

Hortonworks HDP 2.2 with Spark 1.2.x

Hortonworks HDP 2.3 with Spark 1.3.x

Cloudera CDH 5.3 with Spark 1.2.x

Cloudera CDH 5.4 with Spark 1.3.x

SUPPORTED KNIME SOFTWARE VERSIONS

KNIME Analytics Platform 2.12

KNIME Server 4.1

SPARK JOB SERVER SETUP

This section describes how to install the Spark job server on a Linux machine. The KNIME Spark Executor requires the Spark job server to execute and manage Spark jobs.

BACKGROUND

The Spark job server provides a RESTful interface for submitting and managing Apache Spark jobs, jars, and job contexts.

MORE INFORMATION

The Spark job server was originally developed at Ooyala, but the main development repository is now on GitHub. For more information please consult the GitHub repository at https://github.com/spark-jobserver/spark-jobserver/, including licensing conditions, contributors, mailing lists and additional documentation.

In particular, the readme of the job server contains dedicated sections about "HTTPS / SSL Configuration" and "Authentication". The GitHub repository also contains a general Troubleshooting and Tips as well as YARN Tips section. All job server documentation is available in the doc folder of the GitHub repository.

KNIME provides packaged versions of the Spark job server, that can be downloaded on the KNIME.com commercial product download site. Each version is compatible with one of the Hadoop distributions and their respective Spark versions listed above.

VERSION

The packaged versions of the Spark job server provided by KNIME and described in this document are based on the Spark job server of October 06 2015[1]. They have been tested with Hortonworks HDP 2.2/2.3 and Cloudera CDH 5.3/5.4 on CentOS 6 Linux.

INSTALLATION

The Spark job server must be installed on a (Linux) machine that is co-located in the same network as your Hadoop / Spark installation. It can be installed on the Hadoop master server or any other server that has unrestricted access to the Hadoop installation.

1. Download the packaged Spark job server that matches your Hadoop distribution on the KNIME Spark Executor product website under Installation Steps → **<Your KNIME Analytics Platform version> → Supplementary download links for the Spark job server**.

2. Upload the downloaded file to the machine where you want to install the Spark Job Server.

3. The recommend installation procedure is to log in as root on that machine and install the job server under `/opt` (replace xxx with the version of your download):
```
root@host$ cp /path/to/spark-job-server-xxx.tar.gz /opt
root@host$ cd /opt
root@host$ tar xzf spark-job-server-xxx.tar.gz
root@host$ ln -s spark-job-server-xxx spark-job-server
```

4. If you are installing to RedHat Enterprise Linux (RHEL) 6.x[2], a boot script is provided and can be installed as follows:

[1] GitHub revision c99571914c5947eea58ec0381eb1113148827782

```
root@host$ ln -s /opt/spark-job-server/spark-job-server-init.d \
/etc/init.d/spark-job-server
root@host$ chkconfig --levels 2345 spark-job-server on
```

The boot script will run the spark-job-server as the `spark` system user. If you have installed the job server to a different location, or wish to run the server with a different user, you will have to change the JSDIR and USER variables in the boot script.

5. If you are not using the above boot script, you will have to configure the logging directory and filesystem permissions by hand:

```
root@host$ chown –R spark /opt/spark-job-server/
root@host$ mkdir –p /var/log/spark-job-server
root@host$ chown –R spark /var/log/spark-job-server
```

6. Edit `/opt/spark-job-server/environment.conf` as appropriate. The most important settings are:
   - master: Set...
     - `master = yarn-client` for running Spark in YARN-client mode
     - `master = spark://localhost:7077` for stand-alone mode
     - `master = local[4]` for local debugging.
   - Settings for predefined contexts. Under `context-settings`, you can predefine Spark settings for the default Spark context. Please note that these settings can be partly overwritten by the configuration of the client-side extension. Examples of overwritten settings are `num-cpu-cores` and `memory-per-node`. Under `contexts` you can predefine Spark settings for non-default Spark contexts.

7. Edit settings.sh as appropriate:
   - SPARK_HOME, please change if Spark is not installed under the given location.
   - LOG_DIR, please change if you want to log to a non-default location.

8. Edit log4j-server.properties as appropriate (not necessary unless you wish to change the defaults).

MAINTENANCE

STARTING THE SPARK JOB SERVER

If you have installed the boot-script for RHEL 6.x, start the server via the boot-script:

```
root@host$ /etc/init.d/spark-job-server start
```

Otherwise, start the Spark job server via:

```
root@host$  /opt/spark-job-server/server_start.sh
```

STOPPING THE SPARK JOB SERVER

[2] or any other RedHat-based distribution, that still uses init.d scripts , for example CentOS 6.x

If you have installed the boot-script for RHEL 6.x, stop the server via the boot-script:

```
root@host$ /etc/init.d/spark-job-server stop
```

Otherwise, start the Spark job server via:

```
root@host$  /opt/spark-job-server/server_stop.sh
```

CLEANUP JOB HISTORY

It might be advisable to re-start the Spark job server every once in a while and clean-up the rootdir. Remove either the entire directory or only the jar files under `/tmp/spark-jobserver` , or whichever file system locations you have set in `environment.conf.`

SPARK JOB SERVER WEB UI

Point your browser to http://<server>:<port> to check out the status of the Spark job server. The default port is 8090. Three different tabs provide information about active and completed jobs, contexts and jars.

TROUBLESHOOTING

JOB SERVER FAILS TO RESTART

At times, the Spark job server cannot be restarted when large tables were serialized from KNIME to Spark. It fails with a message similar to java.io.UTFDataFormatException: encoded string too long: 6653559 bytes. In that case it is advisable to delete `/tmp/spark-jobserver`, or whichever file system locations you have set in `environment.conf.`

SPARK COLLABORATIVE FILTERING NODE FAILS

If your Spark Collaborative Filtering node fails with a "Job canceled because SparkContext was shutdown" exception the cause might be missing native libraries on the cluster. If you find the error message JAVA.LANG.UNSATISFIEDLINKERROR: ORG.JBLAS.NATIVEBLAS.DPOSV in your job server log the native JBlas library is missing on your cluster. To install the missing lib execute the following command as root on all cluster nodes:

RedHat-based systems: `yum install libgfortran`

Debian-based systems: `apt-get install libgfortran3`

For detailed instructions on how to install the missing libraries go to the JBlas Github page. For information about the MLlib dependencies see the Dependencies section of the MLlib Guide.

The issue is described in https://spark-project.atlassian.net/browse/SPARK-797

EXTENSION SETUP

This section describes how to install the client-side extension of the KNIME Spark Executor in the KNIME Analytics Platform or the KNIME Server. The extension provides all the necessary KNIME nodes to create workflows that execute on Apache Spark.

INSTALLATION

The extension can be installed via the KNIME Update Manager:

1. Go to File → **Install KNIME Extensions** ...
2. Open the category KNIME.com Extension Store (licenses required).
3. Depending on your Spark version, select either ...
   a. the KNIME Big Data Executor for Spark version 1.2 extension,
   b. or the KNIME Big Data Executor for Spark version 1.3 extension,
4. Click on Next and follow the subsequent dialog steps to install the extension.

Please note that the Spark version (1.2 or 1.3) of the extension needs to match the one of your Hadoop distribution. Only one the two extensions can be installed at any time.

If you don't have direct internet access you can also install the extension from a zipped update site:

1. Download the zipped update site from the KNIME Spark Executor product website under Installation Steps → **<Your KNIME Analytics Platform version>** → Extension for KNIME Analytics Platform/KNIME Server.
2. Then register the update site in the KNIME Analytics Platform via **File → Preferences → Install/Update → Available Software Sites**. Then follow the installation steps for the KNIME Update Manager (see above).

Note that to actually use the client-side extension requires a license, which you can purchase via the KNIME Store.

CONFIGURATION

After installing the client-side extension, you have to configure it to work with your environment e.g. your Spark job server configuration.

To setup the extension within KNIME Analytics Platform, open ***File → Preferences → KNIME → Spark*** and adapt the following settings to your environment:

1. Job server URL: This is the IP address or DNS name of the Linux server the Spark job server is installed on.
2. Job server port: The port the Spark job server is listening for job requests as defined in the environment.conf file. The default port is 8090.
3. Context name: The name of the Spark context. This should be the same for all users of the same Spark job server. If you want to use multiple contexts you should install a Spark job server for each context.
4. Number of CPU cores per node: The number of cores to reserve for the Spark context (only works for standalone mode, as this is equivalent to the Spark configuration parameter spark.cores.max).

5. Memory per node: The amount of memory per node to reserve for the Spark context. Corresponds to the Spark configuration parameter spark.executor.memory.

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