Open for Innovation ®

KNIME

KNIME® Spark Executor

Installation Guide

Version 1.7.1

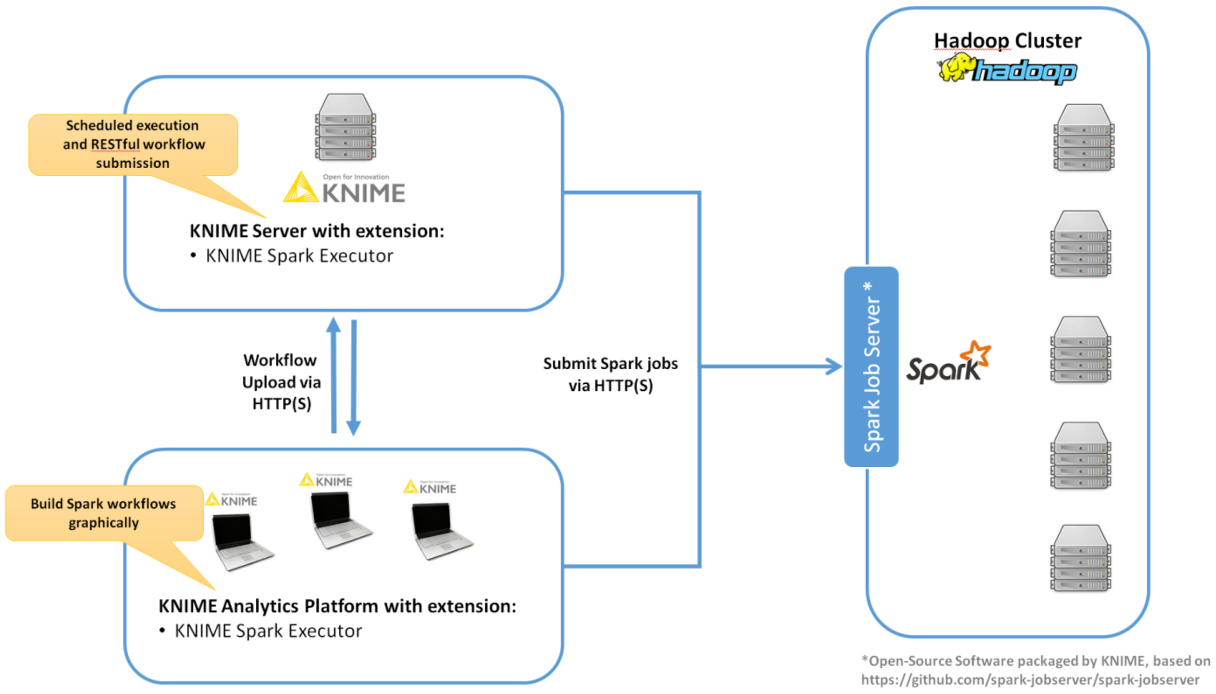
TABLE OF CONTENTS

INTRODUCTION

This document describes the installation procedure of the KNIME® Spark Executor to be used with KNIME® Analytics Platform 3.3 and KNIME® Server 4.4.

As depicted below, KNIME Spark Executor consists of:

- an *extension* for KNIME Analytics Platform/KNIME Server
- a service called *Spark Job Server*, that needs to be installed on an edge node of your Hadoop cluster or a node that can execute the *spark-submit* command.


Scheduled execution and RESTful workflow submission
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
Spark Job Server *
Spark
*Open-Source Software packaged by KNIME, based on https://github.com/spark-jobserver/spark-jobserver

SUPPORTED HADOOP DISTRIBUTIONS

- **Supported <u>without</u> Kerberos security:**
  - Hortonworks HDP 2.2 with Spark 1.2
  - Hortonworks HDP 2.3.0 with Spark 1.3
  - Hortonworks HDP 2.3.4 with Spark 1.5
  - Hortonworks HDP 2.4.x with Spark 1.6
  - Hortonworks HDP 2.5.x with Spark 1.6
  - Cloudera CDH 5.3 with Spark 1.2
  - Cloudera CDH 5.4 with Spark 1.3
  - Cloudera CDH 5.5 with Spark 1.5
  - Cloudera CDH 5.6 with Spark 1.5
  - Cloudera CDH 5.7 with Spark 1.6
  - Cloudera CDH 5.8 with Spark 1.6
  - Cloudera CDH 5.9 with Spark 1.6
- **Supported <u>with</u> Kerberos security:**
  - Hortonworks HDP 2.4.2 and HDP 2.5.x with Spark 1.6
  - Cloudera CDH 5.7, CDH 5.8 and CDH 5.9 with Spark 1.6

SUPPORTED KNIME SOFTWARE VERSIONS

KNIME Spark Executor is compatible with the following versions of KNIME software:

KNIME Analytics Platform 3.3

KNIME Server 4.4

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 Spark Job Server documentation is available in the doc folder of the GitHub repository.

KNIME packages the official Spark Job Server and - if necessary – adapts it for the Hadoop distributions supported by KNIME Spark Executor. These packages can be downloaded on the KNIME Spark Executor product website.

VERSION

The packaged versions of the Spark Job Server provided by KNIME and described in this document are based on the Spark Job Server release 0.6.2.1. They have been tested with the supported Hortonworks and Cloudera versions on CentOS 6 Linux.

REQUIREMENTS

The Spark Job Server runs the so-called Spark driver program and executes Spark jobs submitted via REST. Therefore, it must be installed on a machine that

- runs on Linux (RHEL 6.x/7.x recommended)
- has full network connectivity to all of your Hadoop cluster nodes,
- can be connected to via HTTP (default port TCP/8090) from KNIME Analytics Platform and/or KNIME Server,
- has the Spark, Hadoop and Hive libraries installed and properly configured for your cluster.

This can for example be the Hadoop master node or a cluster edge node.

HOW TO INSTALL

1. Locate the Spark Job Server package that matches your Hadoop distribution on the KNIME Spark Executor product website under **Installation Steps > KNIME Analytics Platform X.Y > Supplementary download links for the Spark Job Server.**

2. Download the file on the machine where you want to install Spark Job Server.

3. The recommend installation procedure is to log in as root on that machine and install the Job Server as follows (replace xxx with the version of your download):

```
root@host$ useradd -d /opt/spark-job-server/ -M -r -s /bin/false
spark-job-server
root@host$ su -l -c "hdfs dfs -mkdir -p /user/spark-job-server ; hdfs
dfs -chown -R spark-job-server /user/spark-job-server" hdfs
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
root@host$ chown -R spark-job-server:spark-job-server /opt/spark-job-
server spark-job-server-xxx/
```

4. If you are installing on RedHat Enterprise Linux (RHEL), a boot script is provided and can be installed as follows:

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

`For RHEL 6.x:`

Execute the following command to make Spark Job Server start during system boot:

```
root@host$ chkconfig --levels 2345 spark-job-server on
```

`For RHEL 7.x:`

Execute the following commands to make Spark Job Server start during system boot:

```
root@host$ systemctl daemon-reload
root@host$ systemctl enable spark-job-server
```

The boot script will run the Job Server as the `spark-job-server` 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-job-server /opt/spark-job-server/
root@host$ mkdir -p /var/log/spark-job-server
root@host$ chown -R spark-job-server /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.

     Note: yarn-cluster mode is currently not supported by Spark Job Server.
   - **Settings for predefined contexts.** Under `context-settings`, you can predefine Spark settings for the default Spark context. Please note that these settings can be overwritten by the configuration of the KNIME extension. 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).

HOW TO INSTALL ON A KERBEROS-SECURED CLUSTER

In a Kerberos secured cluster, Job Server requires a Ticket Granting Ticket (TGT) in order to access Hadoop services and provide user impersonation. To set this up, please first follow the installation steps in the previous section. Then proceed with the following steps:

1. Via kadmin, create a service principal and a keytab file for the `spark-job-server` Linux user. By default this is assumed to be `spark-job-server/host@REALM`, where
   - `host` ist the fully qualified hostname (FQDN) of the machine where you are installing Job Server,
   - `REALM` is the Kerberos realm of your cluster.

2. Upload the keytab file to the machine where Job Server is installed and limit its accessibility to only the `spark-job-server` system user:

```
root@host$ chown spark-job-server:spark-job-server /path/to/keytab
root@host$ chmod go= /path/to/keytab
```

3. Now you have to tell Job Server about the keytab file and, optionally, about the service principal you have created. In `/opt/spark-job-server/settings.sh` uncomment and edit the following lines:

   ```
   export JOBSERVER_KEYTAB=/path/to/keytab
   export JOBSERVER_PRINCIPAL=user/host@REALM
   ```

   Note: You only need to set the principal, if it is different from the assumed default principal `spark-job-server/$(hostname -f)/<default realm from /etc/krb5.conf>`

4. In `/opt/spark-job-server/environment.conf` set the following properties:

   ```
   spark {
     jobserver {
       context-per-jvm = true
     }
   }
   shiro {
     authentication = on
     config.path = "shiro.ini"
     use-as-proxy-user = on
   }
   ```

   The effect of these settings is that Job Server will authenticate all of its users, and each user will have its own Spark context, that can access Hadoop resources in the name of this user.

5. Configure the authentication mechanism of Job Server in `/opt/spark-job-server/shiro.ini`. You can find two example templates in `/opt/spark-job-server/shiro.ini.xxx.template`. Further instructions on configuring authentication can be found in the README file of the Job Server github page.

6. Add the following properties to the core-site.xml of your Hadoop cluster:

   ```
   hadoop.proxyuser.spark-job-server.hosts = *
   hadoop.proxyuser.spark-job-server.groups = *
   ```

   This should be done either via Ambari (on HDP) or Cloudera Manager (on CDH). A restart of the affected Hadoop services (HDFS, YARN) is required.

MAINTENANCE

STARTING THE SPARK JOB SERVER

On RHEL 6, start the server via the boot-script:

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

On RHEL 7 and higher start the server via systemd:

```
root@host$ systemctl start spark-job-server
```

Important Note: It is not recommend to start the server with the `/opt/spark-job-server/server_start.sh script.`

You can verify that the Spark Job Server has correctly started via the WebUI (see Spark Job Server Web UI).

STOPPING THE SPARK JOB SERVER

On RHEL 6, stop the server via the boot-script:

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

On RHEL 7 and higher start the server via systemd:

```
root@host$ systemctl stop spark-job-server
```

Important Note: It is not recommend to stop the server with the `/opt/spark-job-server/server_stop.sh script.`

CLEANUP JOB HISTORY

It is advisable to re-start the Spark Job Server every once in a while, and clean-up its 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

REQUEST TO SPARK JOB SERVER FAILED, BECAUSE THE AMOUNT OF UPLOADED DATA WAS TOO LARGE

Spark Job Server limits how much data can be submitted in a single REST request. For Spark nodes that submit large amounts of data to Spark Job Server, e.g. a large MLlib model, this can result in a request failure with an error that says Request to "Spark Job Server failed, because the amount of uploaded data was too large". This problem can be addressed by increasing the maximum content length in Job Server's webserver. Add and adjust the following section in `environment.conf`:

```
spray.can.server {
      request-chunk-aggregation-limit = 200m
}

spray.can.server.parsing {
      max-content-length = 200m
}
```

SPARK JOB EXECUTION FAILED BECAUSE NO FREE JOB SLOTS WERE AVAILABLE ON SPARK JOB SERVER

Spark Job Server limits how much jobs can run at the same time within the same context. This limit can be changed by adjusting the following setting in `environment.conf`:

```
spark {
  jobserver {
    max-jobs-per-context = 100
  }
}
```

KNIME SPARK EXECUTOR EXTENSION SETUP

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

REQUIREMENTS

Required Software:
A compatible version of KNIME Analytics Platform or KNIME Server (see beginning of this document).

Network Connectivity Requirements:
KNIME Spark Executor extension (the client) needs to be able to make a network connection to the Spark Jobserver service. There are two options to make this connection:

- **Direct connection (recommended):** Client → Spark Jobserver (default port 8090)

- **Proxied connection:** Client → HTTP/HTTPS/SOCKS Proxy → Spark Jobserver (default port 8090). Currently, only proxies that do not require any authentication are supported. Note that KNIME does not provide the proxy software itself.

INSTALLATION

The extension can be installed via the KNIME Update Manager:

1. Go to **File > Install KNIME Extensions ...**
2. Open the category **KNIME.com Big Data Extensions (licenses required)**.
3. Select the **KNIME Spark Executor** extension.
4. Click on Next and follow the subsequent dialog steps to install the extension.

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 → KNIME Analytics Platform X.Y > 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 using 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 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 is allowed to run before being considered failed, and, respectively, in which intervals the status of a job shall be polled.
4. **Spark version:** Please choose the Spark version of the Hadoop cluster you are connecting to.
5. **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.
6. **Delete Spark objects on dispose:** KNIME workflows with Spark nodes create objects such as RDDs during execution. This setting specifies whether those objects shall be deleted when closing a workflow.
7. **Spark job log level:** The Spark jobs triggered by KNIME nodes often create some log messages on the Spark Job Server. Log messages with a log level equal or above the one

specified here will also be display inside KNIME Analytics Platform, which can be useful when debugging workflows.

8. **Override Spark settings:** Custom settings for the remote Spark context, e.g. the amount of memory to allocate per Spark executor. These settings override those from Job Server's *environment.conf.*

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