Open for Innovation
KNIME

KNIME Amazon Web Services Integration User Guide

KNIME AG, Zurich, Switzerland
Version 5.5 (last updated on)

Table of Contents

Overview 1
Create an Amazon EMR cluster 1
Connect to S3 5
Apache Hive 6
    Register the Amazon JDBC Hive driver 6
    Hive Connector 7
    HDFS 8
Amazon Athena 9
    Connect to Amazon Athena 9
    Create an Athena table 11
Execute Spark jobs on an EMR cluster 12
    Create Spark Context (Livy) node 12

Overview

KNIME Analytics Platform includes a set of nodes to interact with Amazon Web Services (AWS™). They allow you to create connections to Amazon services, such as Amazon EMR, or Amazon S3.

The KNIME Amazon Cloud Connectors Extension is available on KNIME Hub.

Create an Amazon EMR cluster

This section describes a step-by-step guide on how to create an EMR cluster.

> The following guide aims to create a standard EMR Spark cluster for testing and education purposes. Please modify the settings and configurations according to your needs.

The following prerequisites are necessary before launching an EMR cluster:

- An Amazon AWS account. To sign up please follow the instructions provided in the AWS documentation.
- An Amazon S3 bucket. The bucket is needed to exchange data between KNIME and Spark and to store the cluster log files. To create an Amazon S3 bucket, please follow the AWS documentation.

After all the prerequisites are fulfilled, you can create the EMR cluster:

1. In the AWS web console, go to EMR
2. Click the button *Create cluster* at the top of the page

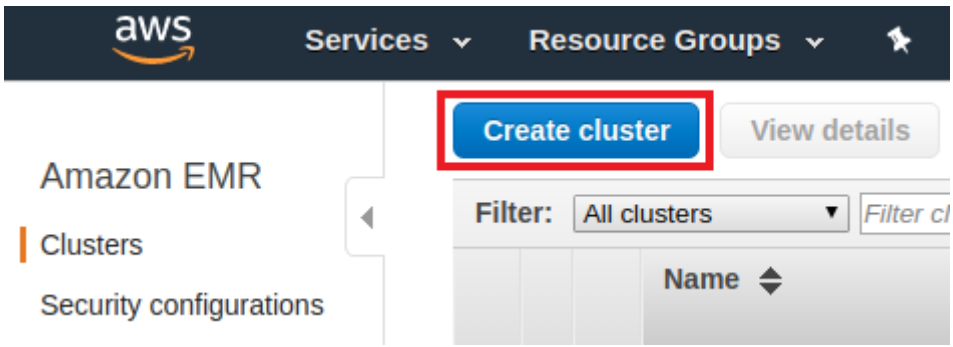

Figure 1. Create cluster button

3. While in the cluster creation page, navigate to the *Advanced options*

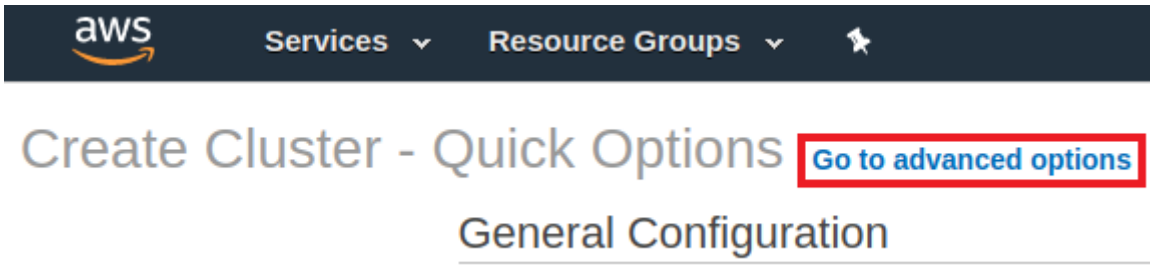

Figure 2. Advanced options

4. Under *Software Configuration*, choose the software to be installed within the cluster. If you want to use Livy and KNIME Spark nodes, install Livy and Spark by checking the corresponding checkboxes.

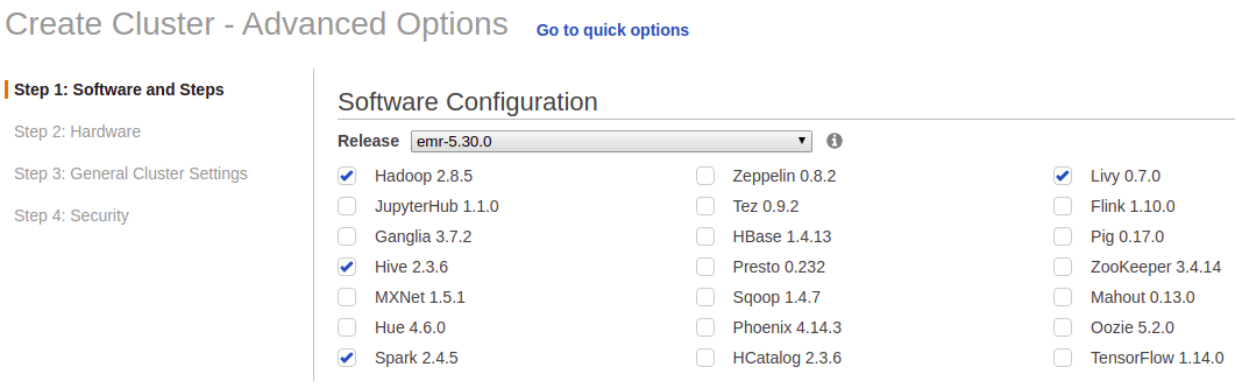

Figure 3. Software configuration

Under *Edit software settings*, you can override the default configurations of applications, such as Spark. In the example below, the spark property *maximizeResourceAllocation* is set to *true* to allow the executors to use the maximum resources possible on each node in a cluster. Please note that this feature works only on a pure Spark cluster (without Hive running in parallel).

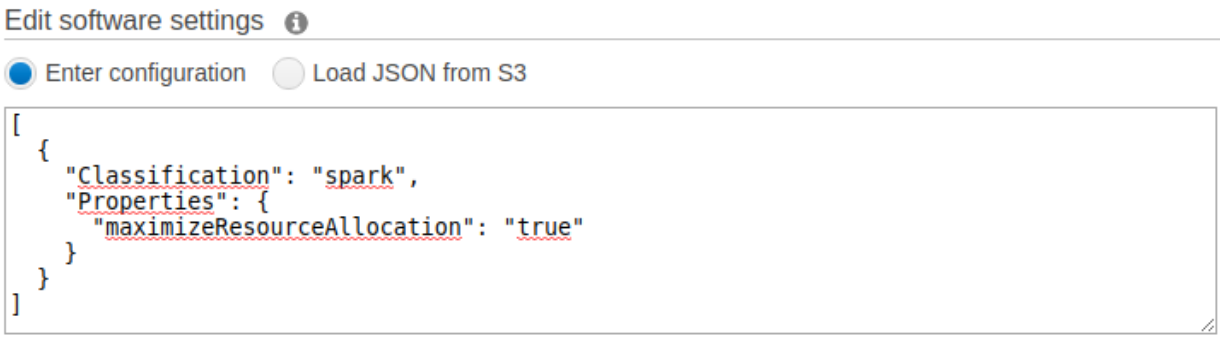

Figure 4. How to maximize resources on a Spark cluster

5. Under *Hardware Configuration*, you can specify the EC2 instance types, number of EC2 instances to initialize in each node, and the purchasing option, depending on your budget. For a standard cluster, it is enough to use the default configuration. The rest of the settings you can keep by default values, or adjust them according to your needs.

For more information on the hardware and network configuration, please check the AWS documentation. For a more in-depth guidance about the optimal number of instances and other related things, please check the corresponding guidelines in the AWS documentation as well.

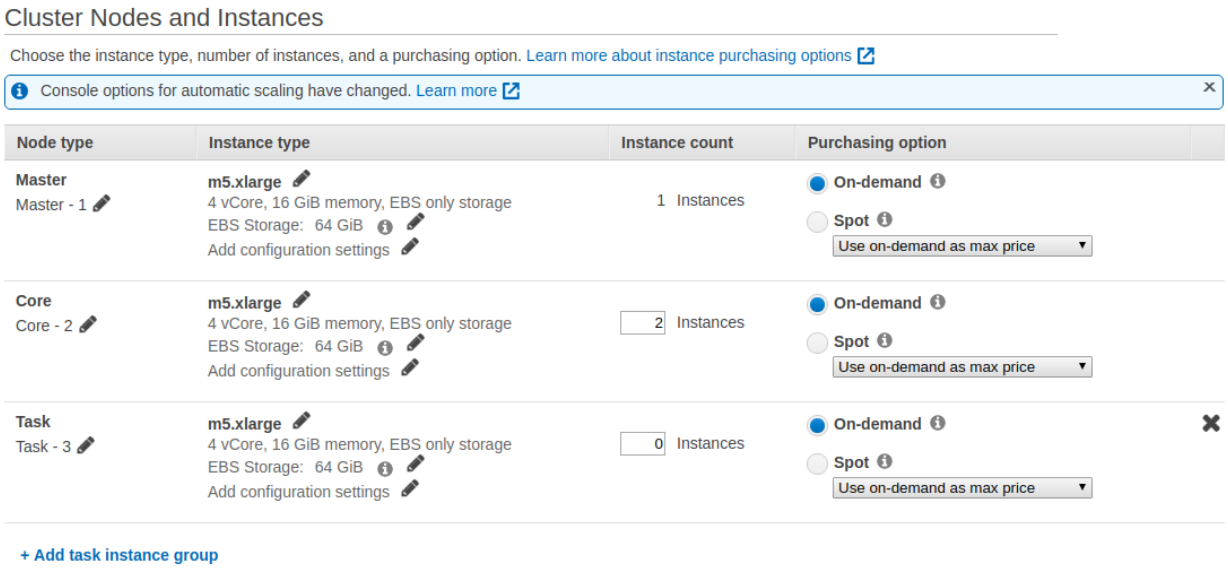

Figure 5. Hardware configuration

6. Under *General Options*, enter the cluster name. *Termination Protection* is enabled by default and is important to prevent accidental termination of the cluster. To terminate the cluster, you must disable termination protection.
7. Under *Security options*, there is an option to specify the EC2 key pair. You can proceed without an EC2 key pair, but if you do have one and you want to SSH into the EMR cluster later, you can provide it here.

   Further down the page, you can also specify the EC2 security group. It acts as a virtual firewall around your cluster and controls all inbound and outbound traffic of your cluster nodes. A default EMR-managed security group is created automatically for your new cluster, and you can edit the network rules in the security group after the cluster is created. Follow the instructions in the AWS documentation on how to work with EMR-managed security groups.

   If needed, add your IP to the *Inbound* rules to enable access to the cluster.

   To make some AWS services accessible from KNIME Analytics Platform, you need to enable specific ports of the EMR master node. For example, Hive is accessible via port 10000.

8. Click *Create cluster* and the cluster will be launched. It might take a few minutes until all the resources are available. You know the cluster is ready when there is a Waiting sign

beside the cluster name (see Figure 6).

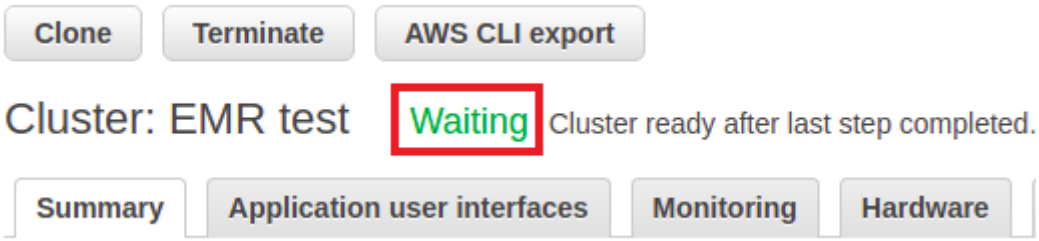

Figure 6. Cluster is ready

Connect to S3

You will need the Amazon Authentication node and Amazon S3 Connector node to create a connection to Amazon S3 from within KNIME Analytics Platform. For more details, please check out the new KNIME File Handling Guide.

> You can check whether a connection can be successfully established by clicking on the *Test connection* button in the configuration dialog of the Amazon Authentication node. A new pop-up window will appear showing the connection information in the format of *s3://accessKeyId@region* and whether a connection is successfully created.

After the connection to Amazon S3 is established, you can then use a variety of the KNIME file handling nodes to manage files on Amazon S3 (see Figure 7).

> The KNIME file handling nodes are available in the node repository under *IO*.

> For more information on Amazon S3, please check out the AWS Documentation.

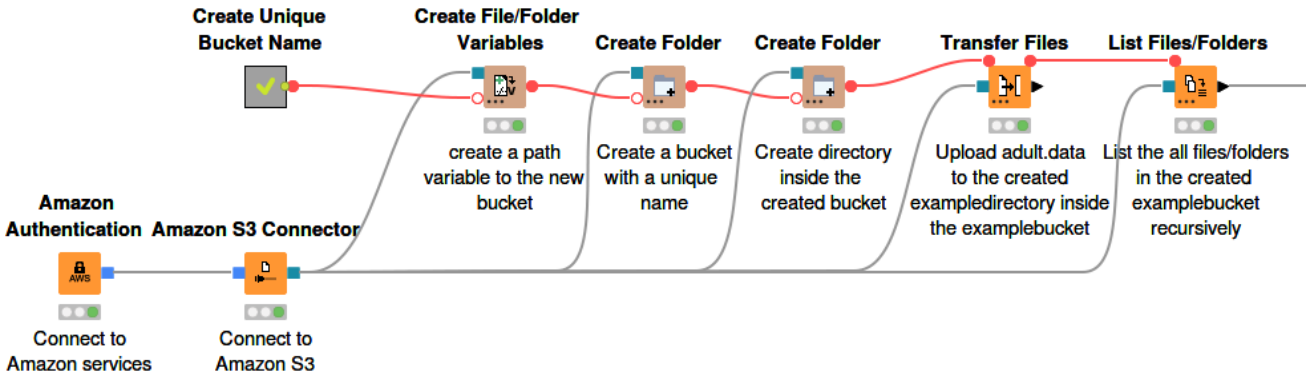

Figure 7. Example usage of Amazon Authentication node and Amazon S3 Connector node

> An example workflow on how to connect and work with remote files on S3 is available on KNIME Hub.

Apache Hive

This section describes how to establish a connection to Hive on EMR in KNIME Analytics Platform.

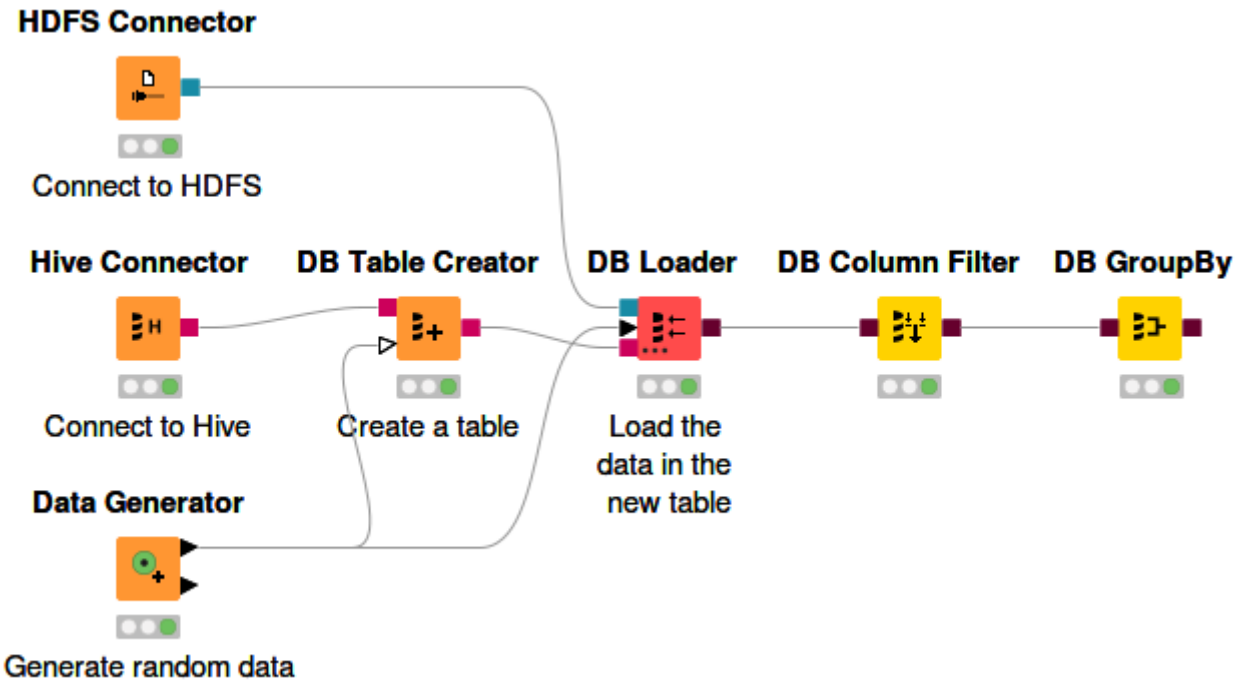

Figure 8. Connect to Hive and create a Hive table

In Figure 8, an example workflow is shown on how to connect to Hive and create a Hive table.

Register the Amazon JDBC Hive driver

To register the Amazon JDBC Hive driver in KNIME Analytics Platform:

1. Download the driver from the AWS website
2. Extract the `.zip` file and the desired driver version
3. Follow the instruction in the Database documentation on how to register an external JDBC driver in KNIME.

> For more information about the Amazon JDBC Hive driver, please check the AWS documentation.

Hive Connector

The Hive Connector node creates a connection via JDBC to a Hive database. The output of this node is a database connection that can be used with the standard KNIME database nodes.

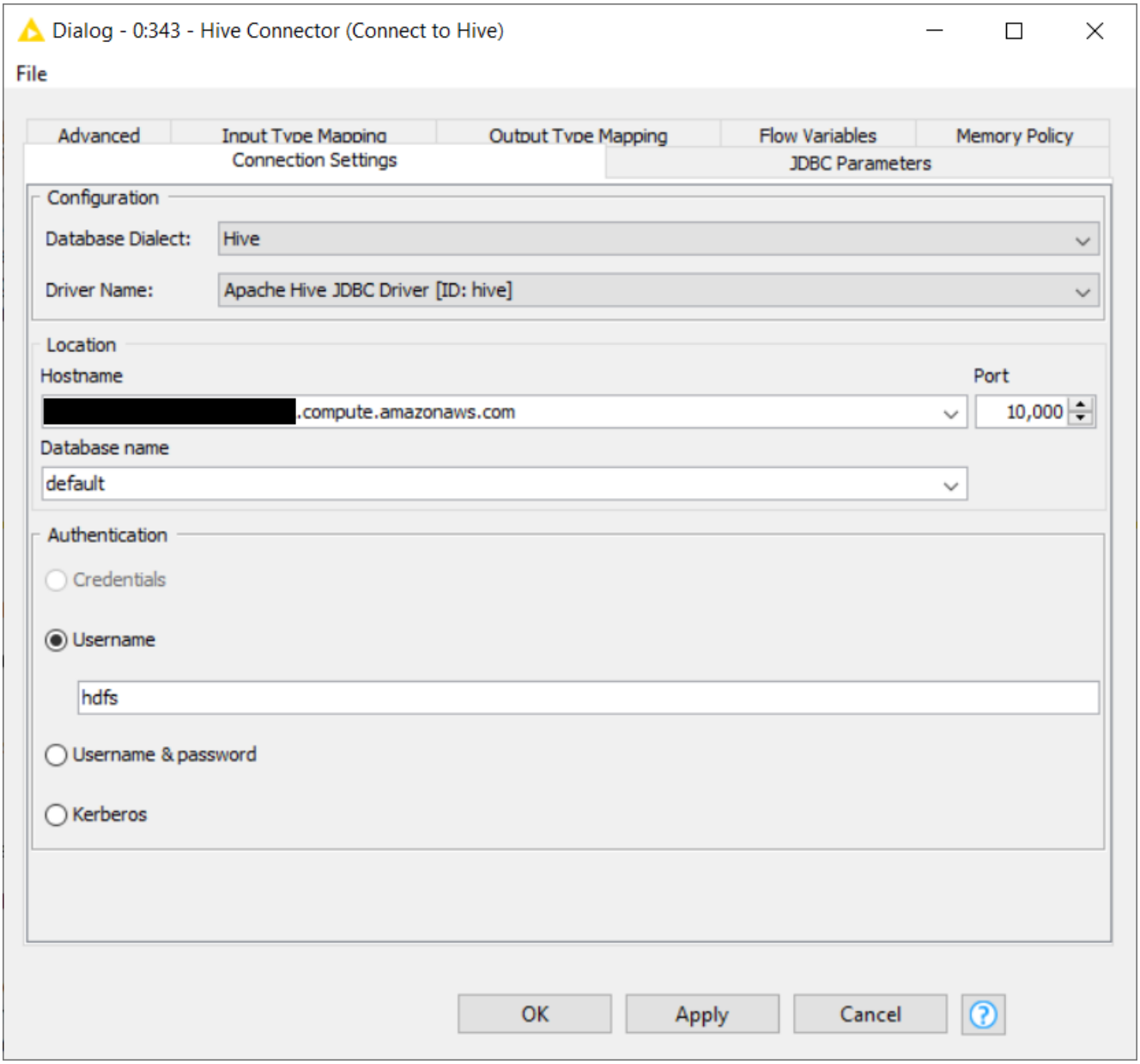

Figure 9. Hive Connector configuration dialog

Inside the node configuration dialog, you have to specify:

- Database dialect and driver name. The driver name is the name given to the driver when registering the driver (see previous section on how to register the Hive driver).
- Server hostname (or IP address), the port, and a database name
- Authentication mechanism. By default, the username *hdfs* can be used as username without a password.

For more information about the advanced options inside the Connector node, please check the KNIME database documentation.

HDFS

To upload or work with remote files on the EMR cluster, it is recommended to use the HDFS Connector node (Figure 8). Amazon EMR 5.x can use *hdfs* or *hadoop* as HDFS administrator user.

Amazon Athena

This section describes Amazon Athena and how to connect to it, as well as create an Athena table, via KNIME Analytics Platform.

Amazon Athena is a query service where users are able to run SQL queries against their data that are located on Amazon S3. In Athena, databases and tables contain basically the metadata for the underlying source data. For each dataset, a corresponding table needs to be created in Athena. The metadata contains information such as the location of the dataset in Amazon S3, and the structure of the data, e.g. column names, data types, and so on.

The KNIME Amazon Athena Connector Extension is available on KNIME Hub.

It is very important to note that Athena only reads your data on S3, you can't add or modify it.

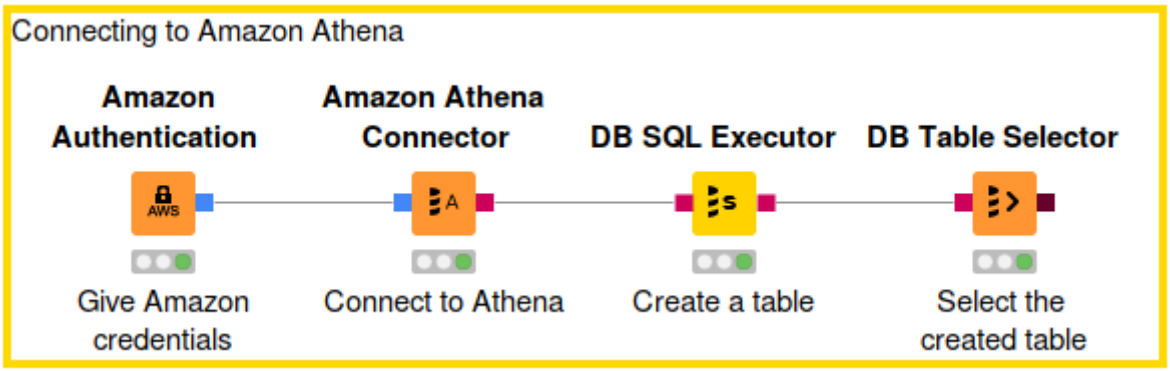

Figure 10. Connect to Athena and create an Athena table

Connect to Amazon Athena

Due to license restriction you need to download the Athena JDBC driver from Amazon and register it once prior connecting to Athena. To download the driver please click here and download the latest version of the JDBC driver **without** the AWS SDK e.g. *AthenaJDBC42-2.0.35.1001.jar*. Once downloaded register the driver via the KNIME preference page with *athena* as database type as described in the Database Extension Guide.

To connect to Amazon Athena via KNIME Analytics Platform:

1. Use the Amazon Authentication node to create a connection to AWS services. In the node configuration dialog please provide the AWS access key ID and secret access key. For more information about AWS access keys, see the AWS documentation.

2. The Amazon Athena Connector node creates a connection to Athena through the built-in Athena JDBC driver. Please provide the following information in the node configuration dialog:
    a. The hostname of the Athena server. It has the format of *athena.<REGION_NAME>.amazonaws.com*. For example: *athena.eu-west-1.amazonaws.com*.
    b. Name of the S3 staging directory to store the query result. For example, *s3://aws-athena-query-results-eu-west-1/*.

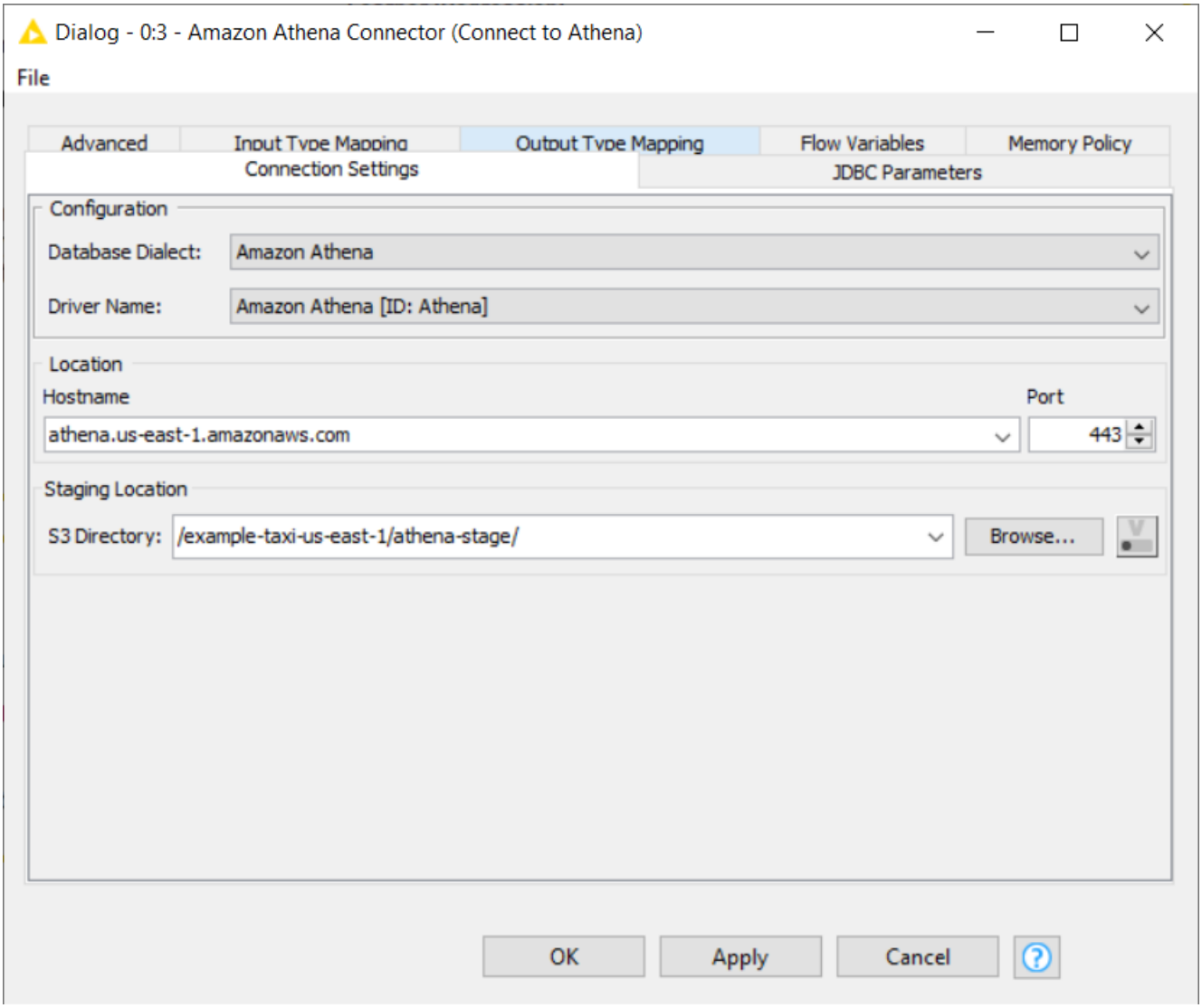

Figure 11. Athena Connector node

After executing this node, a connection to Athena will be established. But before you can start querying data located in S3, you have to create the corresponding Athena table.

Create an Athena table

Creating an Athena table in KNIME Analytics Platform requires a SQL statement, where you have to build your own *CREATE TABLE* statement. The example below shows a *CREATE TABLE* statement to create a table for the Amazon CloudFront log dataset which is a part of the public example Athena dataset made available at *s3://athena-examples-<YOUR-REGION>/cloudfront/plaintext/*. After building your own *CREATE TABLE* statement, copy the statement to the node configuration dialog of DB SQL Executor node.

```
CREATE EXTERNAL TABLE IF NOT EXISTS cloudfront_logs (
  `Date` DATE,
  Time STRING,
  Location STRING,
  Bytes INT,
  RequestIP STRING,
  Method STRING,
  Host STRING,
  Uri STRING,
  Status INT,
  Referrer STRING,
  os STRING,
  Browser STRING,
  BrowserVersion STRING
) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
 "input.regex" = "^(?!#)([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^
]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^
]+)\\s+[^\(]+[\(]([^\;]+).*\%20([^\/]+)[\/](.*)$"
) LOCATION 's3://athena-examples-<YOUR-REGION>/cloudfront/plaintext/';
```

Once the DB SQL Executor node is executed, the corresponding Athena table that contains metadata of the data files is created. Now you can query the files using the standard KNIME database nodes.

> If you are not familiar with SQL and prefer to do it interactively, you can also create the table using the Athena web console. This way, you can even let AWS Glue Crawlers to detect the file schema (column names, column types, among other things) automatically instead of entering them manually. Follow the tutorial in the Athena documentation for a more in-depth explanation.

An example workflow to demonstrate the usage of the Athena Connector node to connect to Amazon Athena from within KNIME Analytics Platform is available on KNIME Hub (see Figure 10).

Execute Spark jobs on an EMR cluster

This section describes how to configure and run a Spark job on an EMR cluster from within KNIME Analytics Platform. Before running a Spark job on an EMR cluster, a Spark context has to be created. To create a Spark context via Livy, use the Create Spark Context (Livy) node.

Create Spark Context (Livy) node

The Create Spark Context (Livy) node creates a Spark context via Apache Livy. The node has a remote connection port (blue) as input. The idea is that this node needs to have access to a remote file system to store temporary files between KNIME and the Spark context.

A wide array of file systems are supported, such as HDFS, webHDFS, httpFS, Amazon S3, Azure Blob Store, and Google Cloud Storage. However, please note that using, e.g HDFS is complicated on a remote cluster because the storage is located on the cluster, hence any data that is stored there will be lost as soon as the cluster is terminated.

The recommended and easy way is to use Amazon S3. Please check the previous section Connect to S3 on how to establish a connection to Amazon S3.

The other connector nodes are available under *IO > Connectors* inside the node repository.

Open the node configuration dialog of the Create Spark Context (Livy) node. In this window you have to provide some information, the most important are:

- The Spark version. The version has to be the same as the one used by Livy. Otherwise the node will fail. You can find the Spark version in the cluster summary page, or in the *Software configuration* step during cluster creation (see Figure 3) on the Amazon EMR web console.
- The Livy URL including the protocol and port e.g. *http://localhost:8998*. You can find the URL in the cluster summary page on the Amazon EMR web console (see Figure 12). Then simply attach the default port 8998 to the end of the URL.

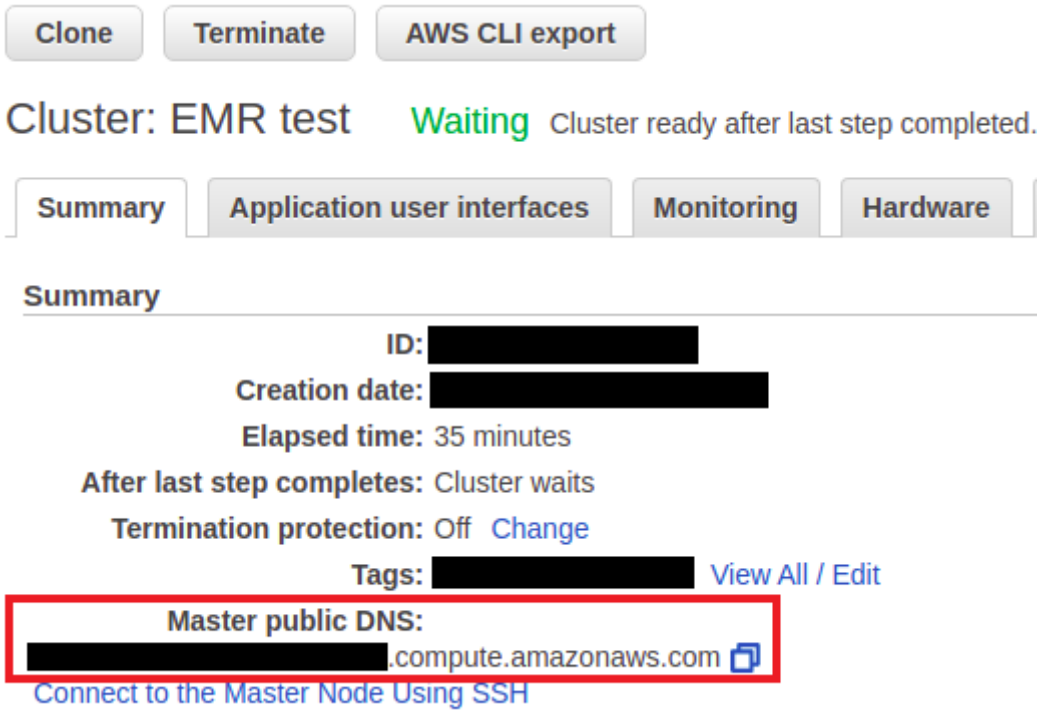

Figure 12. The Livy URL on the cluster summary page

- Select the authentication method. Usually no authentication is required, but if, for example, you have setup a Kerberos authentication on the cluster, you can also use it in KNIME. If that is the case, then you have to set up Kerberos in KNIME Analytics Platform first. Please check the KNIME Kerberos documentation for more details.
- Under *Spark executor resources* section, it is possible to manually set the resources, i.e amount of memory, and number of cores, for each Spark executor. There are three possible Spark executor allocation strategies, default, fixed, and dynamic.
- Under *Advanced* tab, there is an option to set the staging area for Spark jobs. For Amazon S3, it is mandatory to provide a staging directory. Additionally, there is also an option to override the default Spark driver resources (the amount of memory and cores the Spark driver process will allocation), and to specify custom Spark settings.

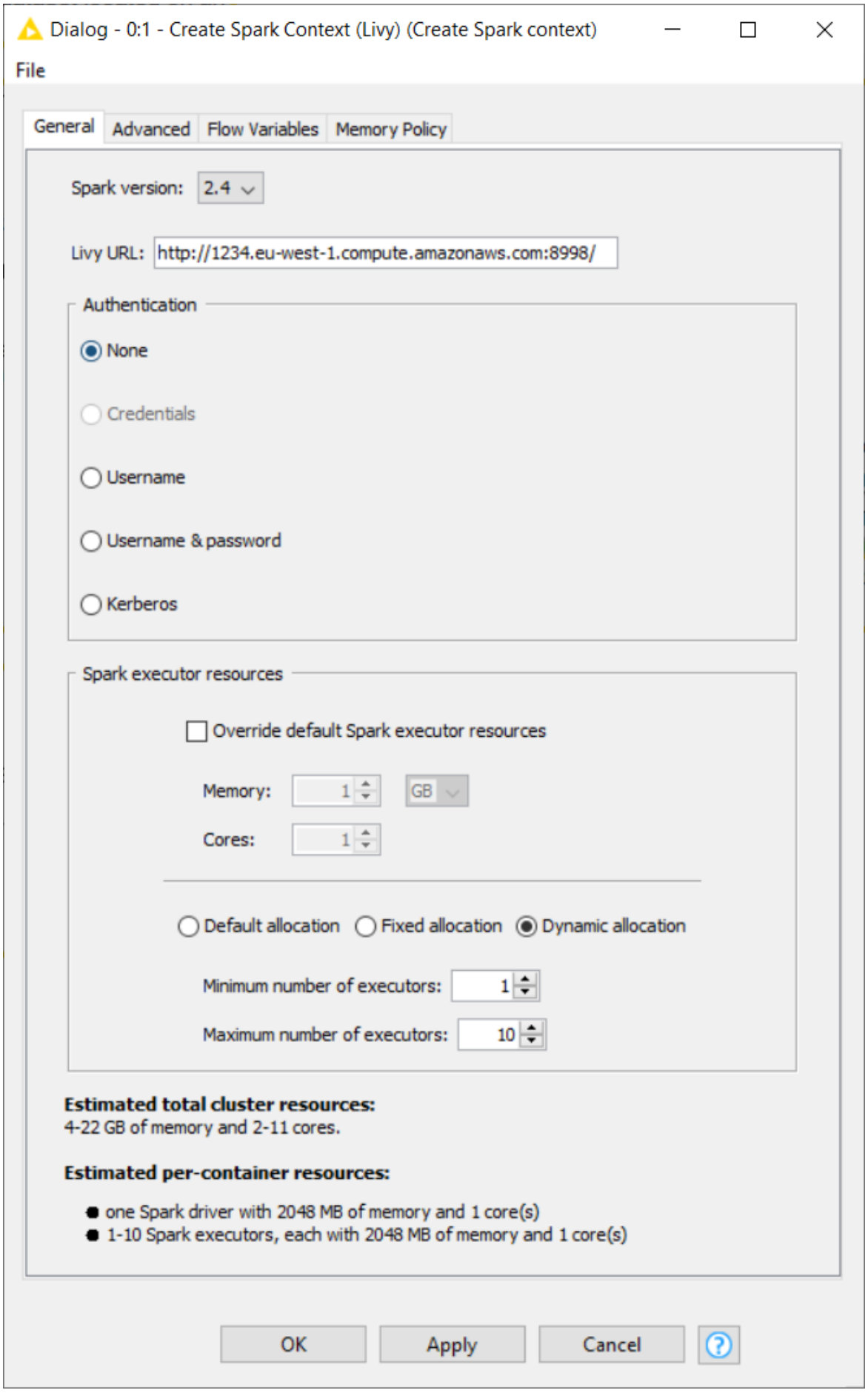

Figure 13. Create Spark Context (Livy) node

After the Create Spark Context (Livy) node is executed, the output Spark node (grey) will contain the newly created Spark context. It allows executing Spark jobs via KNIME Spark nodes.

For a more in-depth explanation on how to read and write data between a remote file system and Spark DataFrame via KNIME Analytics Platform, please check out the KNIME Databricks documentation.

Figure 14 shows a simple example where a Random Forest algorithm is employed to train a prediction model on a dataset, all executed on an EMR Spark cluster.

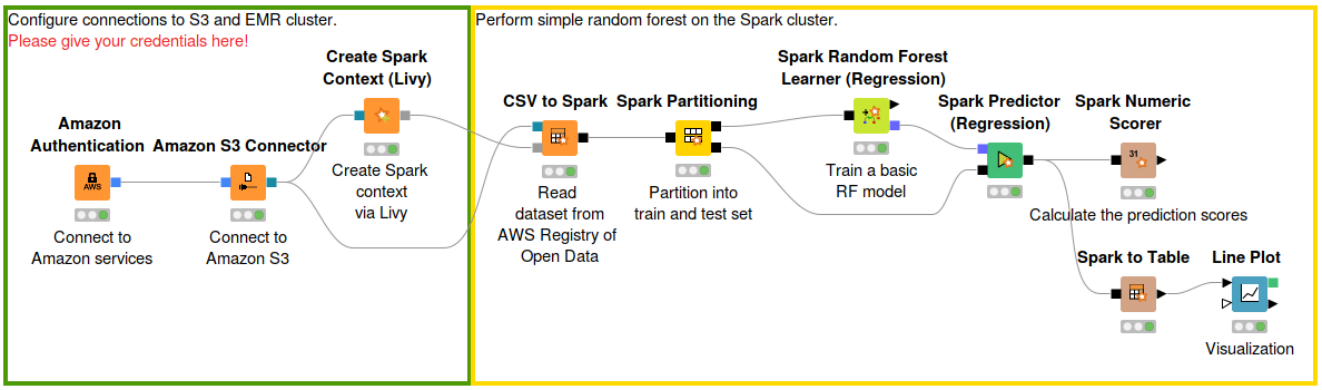

Figure 14. Train a machine learning model on a Spark EMR cluster

An example workflow to demonstrate the usage of Amazon EMR from within KNIME Analytics Platform is available on KNIME Hub.

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