AWS News Blog

New – Apache Spark on Amazon EMR

by Jeff Barr | on 16 JUN 2015 | in Amazon EMR, Launch, News | Permalink | Share

My colleague Jon Fritz wrote the guest post below to introduce a powerful new feature for Amazon EMR. We updated it on May 16, 2017 in order to describe some new console features and to account for the availability of Spark 2.0.

— Jeff;

I'm happy to announce that Amazon EMR now supports Apache Spark. Amazon EMR is a web service that makes it easy for you to process and analyze vast amounts of data using applications in the Hadoop ecosystem, including Hive, Pig, HBase, Presto, Impala, and others. We're delighted to officially add Spark to this list. Although many customers have previously been installing Spark using custom scripts, you can now launch an Amazon EMR cluster with Spark directly from the Amazon EMR Console, CLI, or API.

Apache Spark: Beyond Hadoop MapReduce

We have seen great customer successes using Hadoop MapReduce for large scale data processing, batch reporting, ad hoc analysis on unstructured data, and machine learning. Apache Spark, a newer distributed processing framework in the Hadoop ecosystem, is also proving to be an enticing engine by increasing job performance and development velocity for certain workloads.

By using a directed acyclic graph (DAG) execution engine, Spark can create a more efficient query plan for data transformations. Also, Spark uses in-memory, fault-tolerant resilient distributed datasets (RDDs), keeping intermediates, inputs, and outputs in memory instead of on disk. These two elements of functionality can result in better performance for certain workloads when compared to Hadoop MapReduce, which will force jobs into a sequential map-reduce framework and incurs an I/O cost from writing intermediates out to disk. Spark's performance enhancements are particularly applicable for iterative workloads, which are common in machine learning and low-latency querying use cases.

Additionally, Spark natively supports Scala, Python, and Java APIs, and it includes libraries for SQL, popular machine learning algorithms, graph processing, and stream processing. With many tightly integrated development options, it can be easier to create and maintain applications for Spark than to work with the various abstractions wrapped around the Hadoop MapReduce API.

Introducing Spark on Amazon EMR

Today, we are introducing support for Apache Spark in Amazon EMR. You can quickly and easily create scalable, managed Spark clusters on a variety of Amazon Elastic Compute Cloud (EC2) instance types from the Amazon EMR console, AWS Command Line Interface (CLI) or directly using the Amazon EMR API. As an engine running in the Amazon EMR container, Spark can take advantage of Amazon EMR FS (EMRFS) to directly access data in Amazon Simple Storage Service (S3), push logs to Amazon S3, utilize EC2 Spot capacity for lower costs, and can leverage

Amazon EMR's integration with AWS security features such as IAM roles, EC2 security groups, and S3 encryption at rest (server-side and client-side). Even better, there is no additional charge to run Spark in Amazon EMR.

Spark includes Spark SQL for low-latency, interactive SQL queries, MLlib for out-of-the-box scalable, distributed machine learning algorithms, Spark Streaming for building resilient stream processing applications, and GraphX for graph processing. You can also install Ganglia on your Amazon EMR clusters for additional Spark monitoring. You can send workloads to Spark by submitting Spark steps to the EMR Step API for batch jobs, or interacting directly with the Spark API or Spark Shell on the master node of your cluster for interactive workflows.

Example Customer Use Cases

Even before today's launch, many customers have been working with Spark on Amazon EMR, using bootstrap actions for installation. Here are a few examples:

- The Washington Post is using Spark to power a recommendation engine to show additional content to their readers.
- Yelp, a consumer application that connects users with local businesses, leverages the machine learning libraries included in MLlib with Spark to increase the click-through rates of display advertisements.
- Hearst Corporation uses Spark Streaming to quickly process clickstream data from over 200 web properties. This allows them to create a real-time view of article performance and trending topics.
- Krux uses Spark in its Data Management Platform to process log data stored in Amazon S3 using EMRFS.

Analytics With Spark – A Quick Example

To show an example of how quickly you can start processing data using Spark on Amazon EMR, let's ask a few questions about flight delays and cancellations for domestic flights in the US.

The Department of Transportation has a public data set outlining flight information since 1987. I downloaded it, converted the file format from CSV to the columnar Parquet format (for better performance), and uploaded it to a public, read-only S3 bucket (**s3://us-east-1.elasticmapreduce.samples/flightdata/input**). The data set is around 4 GB compressed (79 GB uncompressed) and contains 162,212,419 rows, so it makes sense to use a distributed framework like Spark for querying. Specifically, I would like to know the 10 airports with the most departures, the most flight delays over 15 minutes, the most flight delays over 60 minutes, the and most flight cancellations. I also want to know the number of flight cancellations by yearly quarter, and the 10 most popular flight routes.

I translated these questions into SQL queries, and wrote a Spark application in Scala to run these queries. You can download the Scala code from **s3://us-east-1.elasticmapreduce.samples/flightdata/sparkapp/FlightSample.scala;** I have an excerpt below:

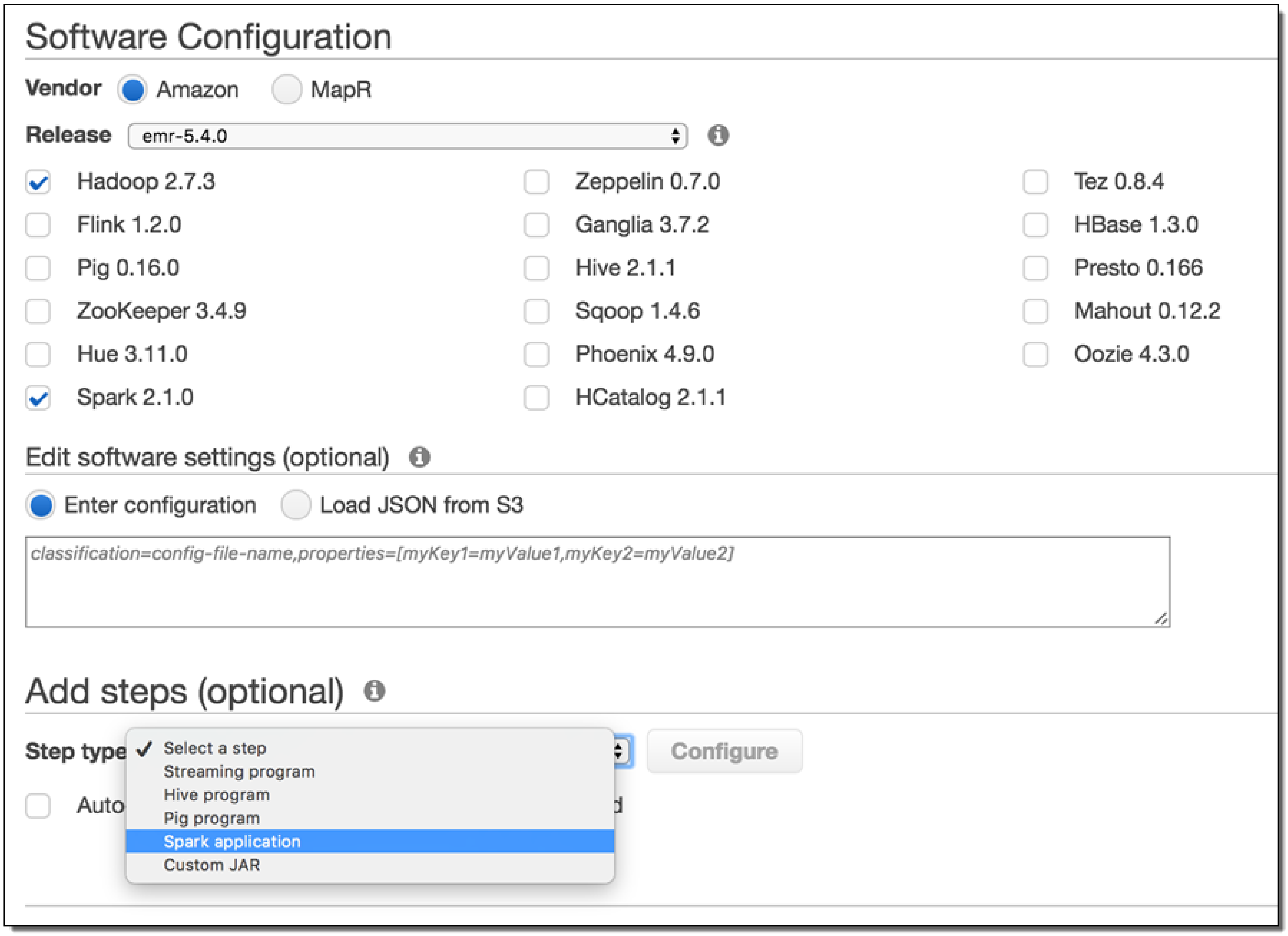
Scala
```
val df = spark.read.parquet("s3://us-east-1.elasticmapreduce.samples/flightdata/input/

//Parquet files can also be registered as tables and then used in SQL statements.
df.createOrReplaceTempView("flights")

//Top 10 airports with the most departures since 2000
val topDepartures = session.sql("SELECT origin, count(*) AS total_departures \
  FROM flights WHERE year >= '2000' \
  GROUP BY origin \
  ORDER BY total_departures DESC LIMIT 10")
```

Note that the application creates a table "flights" which is an in-memory DataFrame, and each SQL query uses that table to reduce the I/O cost for the query. Also, Spark in EMR uses EMRFS to directly access data in S3 without needing to copy it into HDFS first. Notice the input dataset and output location in S3. I compiled the application into a JAR, and you can download it at **https://s3.amazonaws.com/us-east-1.elasticmapreduce.samples/flightdata/sparkapp/flightsample-1.0.jar**.

Let's launch a three node **m3.xlarge** Amazon EMR cluster with Spark to execute this application. Make sure you are launching your cluster in the US East region, because the sample dataset for this example is located in that region. Go to the **Create Cluster – Advanced Options** page in the Amazon EMR console. Scroll down to the **Software Configuration** section to add Spark as an application:


Software Configuration

Vendor [● Amazon] ○ MapR

Release emr-5.4.0

✓ Hadoop 2.7.3	☐ Zeppelin 0.7.0	☐ Tez 0.8.4
☐ Flink 1.2.0	☐ Ganglia 3.7.2	☐ HBase 1.3.0
☐ Pig 0.16.0	☐ Hive 2.1.1	☐ Presto 0.166
☐ ZooKeeper 3.4.9	☐ Sqoop 1.4.6	☐ Mahout 0.12.2
☐ Hue 3.11.0	☐ Phoenix 4.9.0	☐ Oozie 4.3.0
✓ Spark 2.1.0	☐ HCatalog 2.1.1	

Edit software settings (optional)

[● Enter configuration] ○ Load JSON from S3

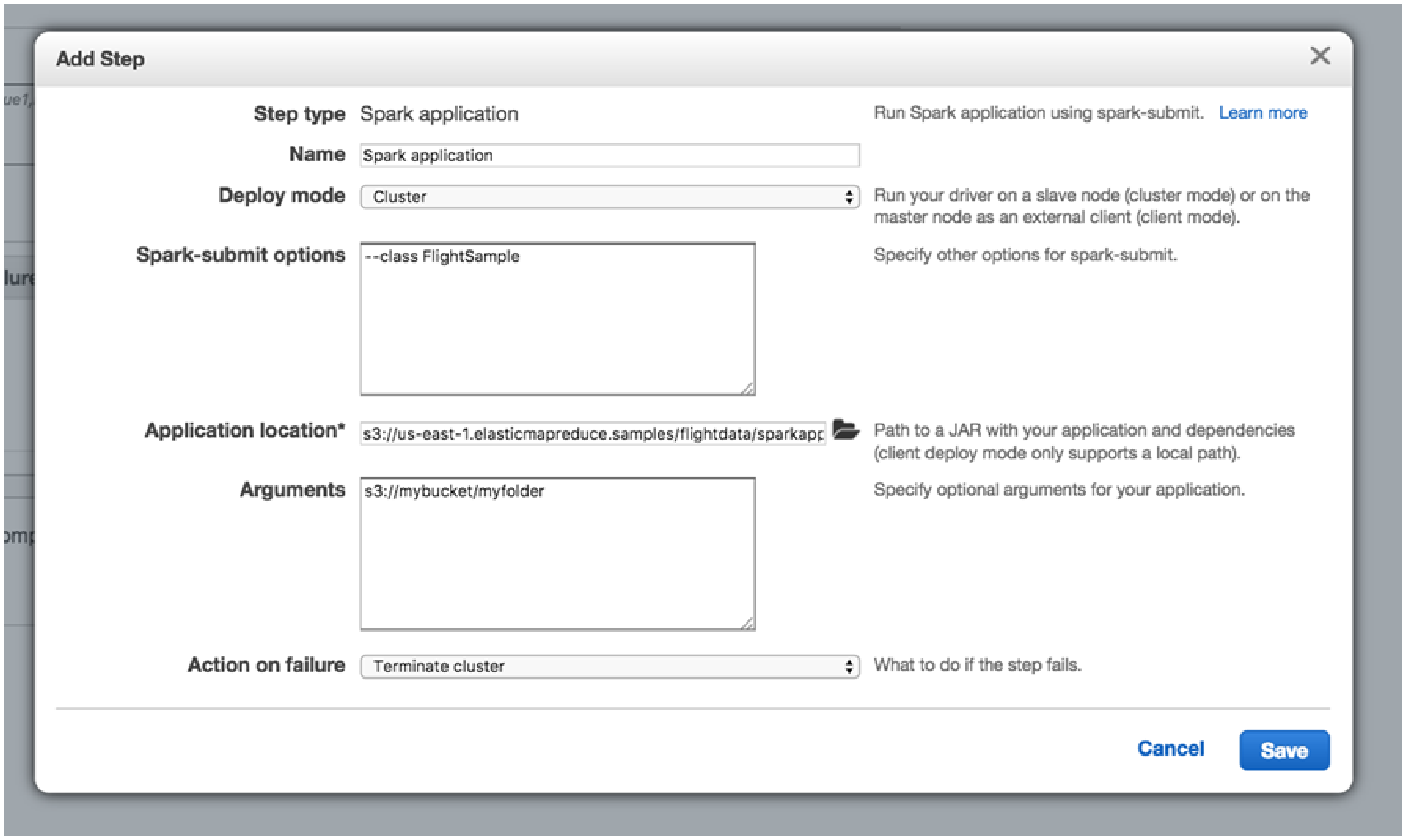
classification=config-file-name,properties=[myKey1=myValue1,myKey2=myValue2]

Add steps (optional)

Step type

By default, Amazon EMR configures Spark to use dynamic allocation of executors and sets the default RAM and cores per executor based on the Core Group instance type. However, you can always overwrite these settings at runtime by passing extra parameters to the actual `spark-submit` command.

Next, scroll to the Steps section near the bottom of the page to add your Spark job. Add a Spark application step:



Add Step

Step type	Spark application	Run Spark application using spark-submit. Learn more
Name	Spark application	
Deploy mode	Cluster	Run your driver on a slave node (cluster mode) or on the master node as an external client (client mode).
Spark-submit options	--class FlightSample	Specify other options for spark-submit.
Application location*	s3://us-east-1.elasticmapreduce.samples/flightdata/sparkapp	Path to a JAR with your application and dependencies (client deploy mode only supports a local path).
Arguments	s3://mybucket/myfolder	Specify optional arguments for your application.
Action on failure	Terminate cluster	What to do if the step fails.

In the modal, select Cluster for Deploy Mode. For Application location, enter **s3://us-east-1.elasticmapreduce.samples/flightdata/sparkapp/flightsample-1.0.jar**. For Arguments, add an S3 path to a bucket in your account where you would like Spark to write the output. And, change Action on failure to Terminate cluster. Then click **Add**.

Then, check the **Auto-terminate cluster after the last step is completed** option at the bottom of the page, so the cluster will automatically shut down once your Spark application is finished running.

Click **Next** to advance to Step 2 in the cluster creation wizard. Review the hardware selection for your cluster. By default, you should see 1 master node and 2 core nodes specified with the m3.xlarge instance type. We will use this instance type and cluster size for this demo.

Click **Next** to advance to Step 3. Uncheck the **Logging and Termination Protection** options under **General Options**. Click **Next** to advance to Step 4 (there are no settings to change on this page), and then continue and click **Create Cluster**.

Amazon EMR will launch the cluster, run your Spark application, and terminate the cluster when the job is complete. You can monitor progress of your job from the **Cluster Details** page in the EMR console. Once the job is complete, you can view the output of the queries in the S3 bucket you indicated. I won't ruin the surprise of the results, either – but definitely bring something to read if you're flying from Chicago O'Hare.

Launch an Amazon EMR Cluster with Spark Today!

For more information about Spark on Amazon EMR, visit the Spark on Amazon EMR page.

— Jon Fritz, Senior Product Manager