

# Spark Fundamentals

*Configuring and monitoring Spark applications*

---

# Contents

**CONFIGURING AND MONITORING SPARK APPLICATIONS ..... 3**

    1.1      CONFIGURING SPARK APPLICATIONS..... 4

    1.2      MONITORING SPARK APPLICATIONS ..... 5

    SUMMARY ..... 9

---

## Configuring and monitoring Spark applications

This lab exercise will show you where you can specify configurations for your Spark environment. It will introduce you to some monitoring tools and methods for tuning Spark applications.

After completing this hands-on lab, you should be able to:

- Understand how to configure, monitor and tune Spark applications.

Allow 15 minutes to complete this section of lab.

## 1.1 Configuring Spark applications

Spark properties control most application settings and are configured separately for each application. These properties can be set directly on the SparkConf object and passed to your SparkContext. You can also set properties by providing it at runtime. For example, submitting a job using the spark-submit command and passing in arguments to that command. In this section, you will set some common properties.

- \_\_\_1. **REFERENCE ONLY:** This doesn't work in the docker image, but is shown here for reference. Change the logging from INFO to ERROR so that only the pertinent information shows up, otherwise the console can be quite verbose. Open up and edit the \$SPARK\_HOME/conf/log4j.properties file. If that file does not exist, make a copy of the log4j.properties.template one and rename it to log4j.properties.

```
biadmin@ibmclass:~/Desktop> cat /opt/ibm/biginsights/spark/conf/log4j.properties
# Set everything to be logged to the console
log4j.rootCategory=ERROR, console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{1}:
%n
```

Next, launch a Spark shell with two cores. You should notice that you no longer see INFO messages on the console.

```
$SPARK_HOME/bin/spark-shell --master local[2]
```

This launches a shell that is only going to utilize two cores on the node, which is the minimum number of cores needed for parallelism. Go ahead and quit out of the Spark shell.

- \_\_\_2. The *spark-submit* command supports loading configurations dynamically. In fact, this is probably the more common usage of loading configurations. You have seen this used in previous lab exercises when you ran sample applications as well as the ones you developed.
- \_\_\_3. Additionally, the spark-submit will also read configurations options from conf/spark-defaults.conf. Let's look at it now:

```
cat $SPARK_HOME/conf/spark-defaults.conf.template
```

```

biadmin@ibmclass:/opt/ibm/biginsights/bin> cat /opt/ibm/biginsights/spark/conf/spark-defaults.conf
# Default system properties included when running spark-submit.
# This is useful for setting default environmental settings.

# Example:
# spark.master                spark://master:7077
# spark.eventLog.enabled      true
# spark.eventLog.dir          hdfs://namenode:8021/directory
# spark.serializer            org.apache.spark.serializer.KryoSerializer
# spark.driver.memory         5g
# spark.executor.extraJavaOptions -XX:+PrintGCDetails -Dkey=value -Dnumbers="one two three"
spark.master yarn-client
spark.yarn.jar hdfs://ibmclass.localdomain.com:9000/biginsights/spark/assembly/target/scala-2.10/spark-assembly-1.1.0-10-hadoop2.4.1.jar
spark.eventLog.enabled true
spark.eventLog.dir hdfs://ibmclass.localdomain.com:9000/biginsights/logs/spark/historyServer
spark.yarn.historyServer.address ibmclass.localdomain.com:18080
#spark.metrics.conf /opt/ibm/biginsights/spark/conf/metrics.properties

```

If you want to use the defaults, make a copy of the template file.

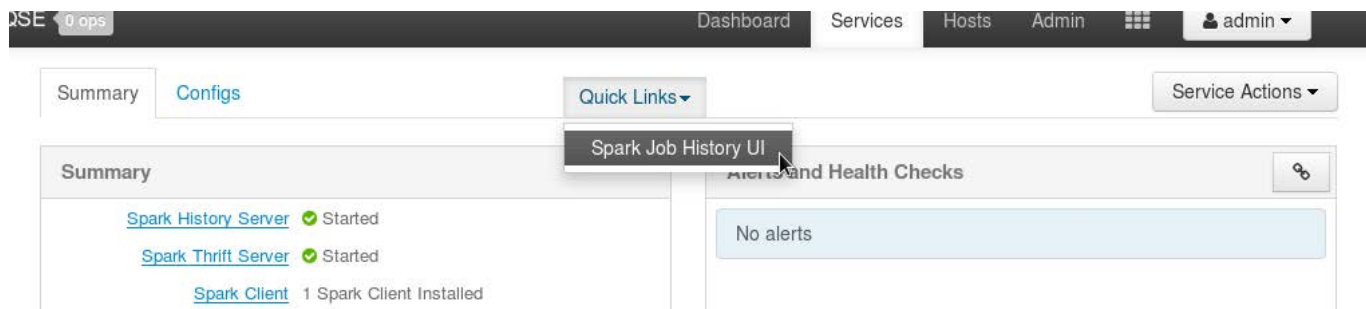
**Note:** Properties set directly on the SparkConf take highest precedence, then flags passed to spark-submit or spark-shell, then options in the spark-defaults.conf file.

## 1.2 Monitoring Spark applications (REFERENCE for BigInsights v4 QSE only)

- \_\_\_1. To view the history server, you can find the port by going to the Ambari console and accessing Spark on the Cluster Status tab:

The screenshot displays the Ambari console interface. At the top, the navigation bar includes 'Ambari', 'BI4\_QSE', '0 ops', 'Dashboard', 'Services', 'Hosts', 'Admin', and a user profile 'admin'. The left sidebar lists services: HDFS, MapReduce2, YARN, Nagios, Ganglia, Hive, HBase, Pig, Sqoop, Oozie, ZooKeeper, Spark (selected), and Flume. The main content area is divided into 'Summary' and 'Configs' tabs. The 'Summary' tab shows the status of Spark components: 'Spark History Server' (Started), 'Spark Thrift Server' (Started), and 'Spark Client' (1 Spark Client Installed). On the right, the 'Alerts and Health Checks' section indicates 'No alerts'.

- \_\_\_2. To access the history server, click on the Quick Links:



## Spark 1.2.1 History Server

Event log directory: <https://rvm.svl.ibm.com:8020/lop/apps/4.0.0.0/spark/logs/history-server>

Showing 1-20 of 24

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated
<a href="#">local-1428016196985</a>	Spark shell	2015/04/02 16:09:55	2015/04/02 16:11:46	1.9 min	virtuser	2015/04/02 16:11:47
<a href="#">local-1428014463211</a>	Spark shell	2015/04/02 15:41:01	2015/04/02 15:45:07	4.1 min	virtuser	2015/04/02 15:45:08
<a href="#">local-1428014379296</a>	Spark shell	2015/04/02 15:39:37	2015/04/02 15:40:48	1.2 min	virtuser	2015/04/02 15:40:49
<a href="#">local-1428014074278</a>	Spark shell	2015/04/02 15:34:32	2015/04/02 15:39:25	4.9 min	virtuser	2015/04/02 15:39:26
<a href="#">local-1428013325947</a>	Spark shell	2015/04/02 15:22:04	2015/04/02 15:34:23	12 min	virtuser	2015/04/02 15:34:24
<a href="#">local-1428013146932</a>	PythonPi	2015/04/02 15:19:05	2015/04/02 15:19:08	3 s	virtuser	2015/04/02 15:19:10
<a href="#">local-1428013023691</a>	WordCount	2015/04/02 15:17:02	2015/04/02 15:17:05	3 s	virtuser	2015/04/02 15:17:07
<a href="#">local-1428011545168</a>	Spark shell	2015/04/02 14:52:23	2015/04/02 15:11:46	19 min	virtuser	2015/04/02 15:11:47
<a href="#">local-1428006794414</a>	PySparkShell	2015/04/02 13:33:12	2015/04/02 15:10:05	1.6 h	virtuser	2015/04/02 15:10:06
<a href="#">local-1428012098375</a>	Spark Pi	2015/04/02 15:01:36	2015/04/02 15:01:39	4 s	virtuser	2015/04/02 15:01:41
<a href="#">local-1428007889665</a>	Spark shell	2015/04/02 13:51:28	2015/04/02 14:52:14	1.0 h	virtuser	2015/04/02 14:52:16

The history server lists the applications that ran recently. Your screenshot will be different from what you see here. The app will only show up after it has finished running.

- \_\_\_3. Let's go through a simple example to see how caching affects the storage. Create a RDD from the README.md file.

```
val readme = sc.textFile("/tmp/README.md")
```

- \_\_\_4. Cache the readme RDD.

```
readme.cache()
```

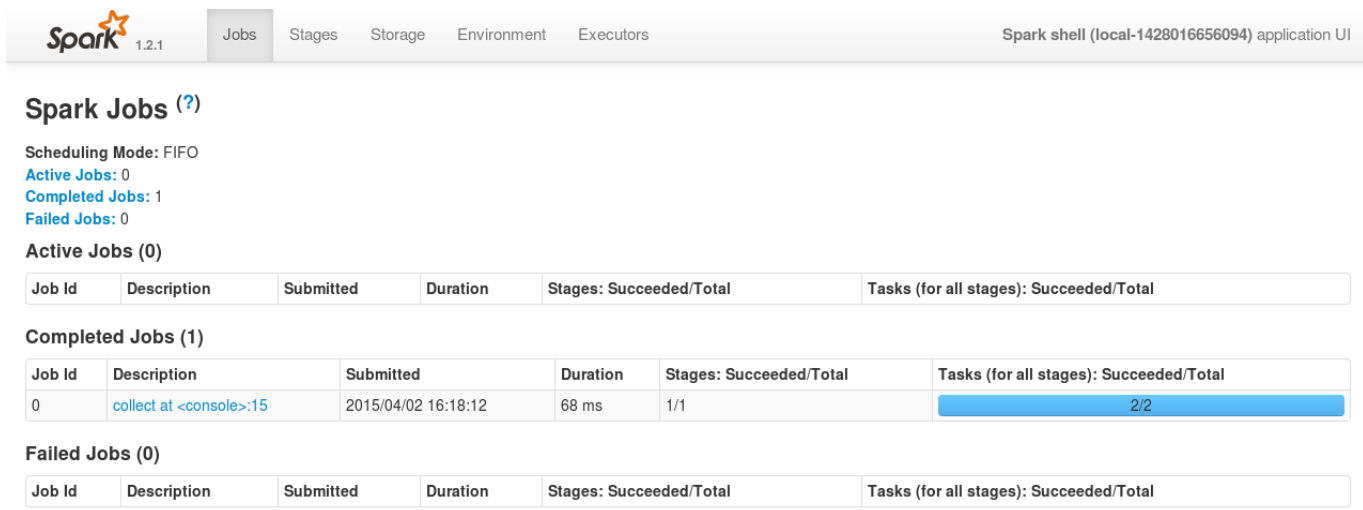
- \_\_\_5. Print the information out onto the console. Remember that transformations do not get applied until some action takes place. This is why we're going printing it out to the console using the collect() action.

```
readme.collect()
```

- \_\_\_6. Quit out of the Spark shell so the entry appears in the history server page.

- \_\_\_7. CTRL + D

- \_\_8. Refresh the history server page to see the new listing of the Spark shell that you just used to run the simple example. Click on the link of that shell for more information.



The screenshot shows the 'Spark Jobs' page. The top navigation bar includes 'Jobs', 'Stages', 'Storage', 'Environment', and 'Executors'. The page title is 'Spark Jobs (?)'. Below the title, it shows 'Scheduling Mode: FIFO', 'Active Jobs: 0', 'Completed Jobs: 1', and 'Failed Jobs: 0'. Under 'Active Jobs (0)', there is an empty table. Under 'Completed Jobs (1)', there is a table with one row: Job Id 0, Description 'collect at <console>:15', Submitted '2015/04/02 16:18:12', Duration '68 ms', Stages: Succeeded/Total '1/1', and Tasks (for all stages): Succeeded/Total '2/2'. Under 'Failed Jobs (0)', there is an empty table.

**Spark Jobs (?)**

Scheduling Mode: FIFO  
 Active Jobs: 0  
 Completed Jobs: 1  
 Failed Jobs: 0

**Active Jobs (0)**

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
--------	-------------	-----------	----------	-------------------------	---

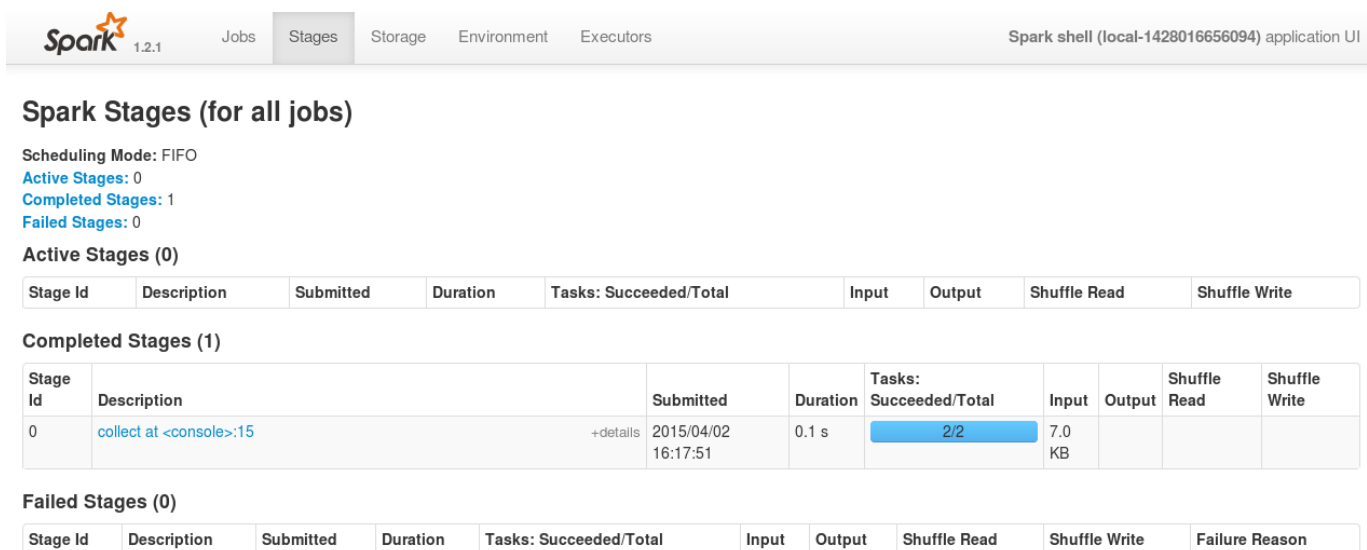
**Completed Jobs (1)**

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	collect at <console>:15	2015/04/02 16:18:12	68 ms	1/1	2/2

**Failed Jobs (0)**

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
--------	-------------	-----------	----------	-------------------------	---

- \_\_9. Click the Stages link on the menu:



The screenshot shows the 'Spark Stages' page. The top navigation bar includes 'Jobs', 'Stages', 'Storage', 'Environment', and 'Executors'. The page title is 'Spark Stages (for all jobs)'. Below the title, it shows 'Scheduling Mode: FIFO', 'Active Stages: 0', 'Completed Stages: 1', and 'Failed Stages: 0'. Under 'Active Stages (0)', there is an empty table. Under 'Completed Stages (1)', there is a table with one row: Stage Id 0, Description 'collect at <console>:15', Submitted '2015/04/02 16:17:51', Duration '0.1 s', Tasks: Succeeded/Total '2/2', Input '7.0 KB', Output, Shuffle Read, and Shuffle Write. Under 'Failed Stages (0)', there is an empty table.

**Spark Stages (for all jobs)**

Scheduling Mode: FIFO  
 Active Stages: 0  
 Completed Stages: 1  
 Failed Stages: 0

**Active Stages (0)**

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
----------	-------------	-----------	----------	------------------------	-------	--------	--------------	---------------

**Completed Stages (1)**

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	collect at <console>:15	2015/04/02 16:17:51	0.1 s	2/2	7.0 KB			

**Failed Stages (0)**

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write	Failure Reason
----------	-------------	-----------	----------	------------------------	-------	--------	--------------	---------------	----------------

- \_\_10. Click on the collect stage to view the details of that run:

## Details for Stage 0

Total task time across all tasks: 0.1 s

Input: 7.0 KB

► Show additional metrics

### Summary Metrics for 2 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	67 ms	67 ms	67 ms	67 ms	67 ms
GC Time	0 ms	0 ms	0 ms	0 ms	0 ms
Input	2.3 KB	2.3 KB	4.7 KB	4.7 KB	4.7 KB

### Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input	Output	Shuffle Read	Shuffle Write	Shuffle Spill (Memory)	Shuffle Spill (Disk)
<driver>	localhost:44643	0.2 s	2	0	2	7.0 KB	0.0 B	0.0 B	0.0 B	0.0 B	0.0 B

### Tasks

\_\_11. There are four sections at the top: Click on the Storage section.

## Storage

RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size in Tachyon	Size on Disk
<a href="#">/tmp/README.md</a>	Memory Deserialized 1x Replicated	2	100%	15.5 KB	0.0 B	0.0 B

\_\_12. You can see that the README.MD file was cached along with the size in memory and the number of cached partitions. If you click on the link, you can see in more detail which blocks and executors that file is distributed:

## RDD Storage Info for /tmp/README.md

Storage Level: Memory Deserialized 1x Replicated

Cached Partitions: 2

Total Partitions: 2

Memory Size: 15.5 KB

Disk Size: 0.0 B

### Data Distribution on 1 Executors

Host	Memory Usage	Disk Usage
localhost:44643	15.5 KB (265.1 MB Remaining)	0.0 B

### 2 Partitions

Block Name	Storage Level	Size in Memory	Size on Disk	Executors
rdd_1_0	Memory Deserialized 1x Replicated	8.2 KB	0.0 B	localhost:44643
rdd_1_1	Memory Deserialized 1x Replicated	7.3 KB	0.0 B	localhost:44643

A lot of the monitoring tools can be used to help with tuning. For example, the previous task that you just did to cache a RDD is actually a good method to use for figuring out how much memory each partition is consuming.



## Summary

Having completed this exercise, you should have a basic understanding of how to specify various Spark configurations through the SparkConf object, as parameters through the spark-submit or spark-shell commands, or by specifying defaults in the spark-defaults.conf file. You also launched the history server Web UI to check out the Spark application that you ran.

## NOTES

## NOTES



---

© Copyright IBM Corporation 2015.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo and [ibm.com](http://ibm.com) are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).



Please Recycle

---