How to stop INFO messages displaying on spark console?

Asked 9 years, 10 months ago Modified 11 months ago Viewed 276k times

228

I'd like to stop various messages that are coming on spark shell.

I tried to edit the `log4j.properties` file in order to stop these message.

Here are the contents of `log4j.properties`

```
# Define the root logger with appender file
log4j.rootCategory=WARN, 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}: %m%n

# Settings to quiet third party logs that are too verbose
log4j.logger.org.eclipse.jetty=WARN
log4j.logger.org.eclipse.jetty.util.component.AbstractLifeCycle=ERROR
log4j.logger.org.apache.spark.repl.SparkIMain$exprTyper=INFO
log4j.logger.org.apache.spark.repl.SparkILoop$SparkILoopInterpreter=INFO
```

But messages are still getting displayed on the console.

Here are some example messages

```
15/01/05 15:11:45 INFO SparkEnv: Registering BlockManagerMaster
15/01/05 15:11:45 INFO DiskBlockManager: Created local directory at /tmp/spark-local-
20150105151145-b1ba
15/01/05 15:11:45 INFO MemoryStore: MemoryStore started with capacity 0.0 B.
15/01/05 15:11:45 INFO ConnectionManager: Bound socket to port 44728 with id =
ConnectionManagerId(192.168.100.85,44728)
15/01/05 15:11:45 INFO BlockManagerMaster: Trying to register BlockManager
15/01/05 15:11:45 INFO BlockManagerMasterActor$BlockManagerInfo: Registering block
manager 192.168.100.85:44728 with 0.0 B RAM
15/01/05 15:11:45 INFO BlockManagerMaster: Registered BlockManager
15/01/05 15:11:45 INFO HttpServer: Starting HTTP Server
15/01/05 15:11:45 INFO HttpBroadcast: Broadcast server star
```

How do I stop these?

apache-spark log4j spark-submit

Share Edit Follow

edited Oct 31, 2018 at 8:43
lospejos
1,976 ● 4 ● 19 ● 38

asked Jan 5, 2015 at 14:04
Vishwas
7,057 ● 7 ● 45 ● 70

22 Answers

Sorted by: Highest score (default)

199

Edit your `conf/log4j.properties` file and change the following line:

```
log4j.rootCategory=INFO, console
```

to

```
log4j.rootCategory=ERROR, console
```

Another approach would be to :

Start spark-shell and type in the following:

```
import org.apache.log4j.Logger
import org.apache.log4j.Level

Logger.getLogger("org").setLevel(Level.OFF)
Logger.getLogger("akka").setLevel(Level.OFF)
```

You won't see any logs after that.

Other options for Level include: `all`, `debug`, `error`, `fatal`, `info`, `off`, `trace`, `trace_int`, `warn`

Details about each can be found in the documentation.

Share Edit Follow

edited Aug 4, 2016 at 18:01
lambda
3,385 ● 1 ● 27 ● 32

answered Jan 7, 2015 at 8:48
AkhlD
2,596 ● 2 ● 17 ● 15

19 I think that OFF is too restrictive. WARN or ERROR may fit better here. – snowindy Aug 29, 2015 at 9:39

2 Add that in your projects Main class. – AkhlD Dec 10, 2015 at 11:22

3 Great answer. Any way to do the same from PySpark programmatically? – Tagar Sep 13, 2016 at 19:56

2 The programmatic part of this does not work. Instead see this answer from @cantdutchthis stackoverflow.com/a/37836847/1056563 – WestCoastProjects Jun 17, 2018 at 18:41

@AkhlD: I have added these 3 in my code. Still I can see INFO and WARN messages in log `log4j.rootCategory=ERROR, console` And ``` val rootLogger = Logger.getRootLogger() rootLogger.setLevel(Level.OFF) Logger.getLogger("org.apache.spark").setLevel(Level.OFF) Logger.getLogger("org.spark-project").setLevel(Level.OFF) Logger.getLogger("org").setLevel(Level.OFF) Logger.getLogger("akka").setLevel(Level.OFF) ``` And `ssc.sparkContext.setLogLevel("ERROR")` – Nikhil Redij May 28, 2020 at 6:50

Right after starting `spark-shell` type ;

```
sc.setLogLevel("ERROR")
```

195

you could put this in preload file and use like:

```
spark-shell ... -I preload-file ...
```

In Spark 2.0 (Scala):

```
spark = SparkSession.builder.getOrCreate()
spark.sparkContext.setLogLevel("ERROR")
```

API Docs :
https://spark.apache.org/docs/2.2.0/api/scala/index.html#org.apache.spark.sql.SparkSession

For Java:

```
spark = SparkSession.builder.getOrCreate();
spark.sparkContext().setLogLevel("ERROR");
```

Share Edit Follow

edited Jul 16, 2022 at 9:16
Eugene Lopatkin
2,651 ● 1 ● 24 ● 36

answered Jun 15, 2016 at 13:27
cantdutchthis
34.2k ● 17 ● 76 ● 116

How would you set this property in a program? – Alex Raj Kaliamoorthy Jul 25, 2016 at 19:34

This is only avaibale for spark.sql.SparkSession or also avaibale for JavaSparkContext ?? – SharpLu Aug 7, 2017 at 13:18

1 This is the only answer which worked for me without created a separate log4j. thanks! – abhihello123 Apr 28, 2018 at 10:33

3 It works for me, however I'm still getting a couple of messages at the beginning of my test. Any idea? – user2241239 Jun 22, 2018 at 12:56

1 In spark 2.3.1, this reduces my messages by half, but I still get lots of INFO – Toby Eggitt Jun 29, 2018 at 18:03

91

All the methods collected with examples

Intro

Actually, **there are many ways to do it**. Some are harder from others, but it is up to you which one suits you best. I will try to showcase them all.

#1 Programatically in your app

Seems to be the easiest, but you will need to recompile your app to change those settings. Personally, I don't like it but it works fine.

Example:

```
import org.apache.log4j.{Level, Logger}

val rootLogger = Logger.getRootLogger()
rootLogger.setLevel(Level.ERROR)

Logger.getLogger("org.apache.spark").setLevel(Level.WARN)
Logger.getLogger("org.spark-project").setLevel(Level.WARN)
```

You can achieve much more just using `log4j` API.
Source: [Log4J Configuration Docs, Configuration section]

#2 Pass `log4j.properties` during `spark-submit`

This one is very tricky, but not impossible. And my favorite.

Log4J during app startup is always looking for and loading `log4j.properties` file from classpath.

However, when using `spark-submit` Spark Cluster's classpath has precedence over app's classpath! This is why putting this file in your fat-jar will not override the cluster's settings!

> Add `-Dlog4j.configuration=<location of configuration file>` to `spark.driver.extraJavaOptions` *(for the driver)* or `spark.executor.extraJavaOptions` *(for executors)*.
>
> Note that if using a file, the `file:` protocol should be explicitly provided, **and the file needs to exist locally on all the nodes**.

To satisfy the last condition, you can either upload the file to the location available for the nodes (like `hdfs`) or access it locally with driver if using `deploy-mode client`. Otherwise:

> upload a custom `log4j.properties` using spark-submit, by adding it to the `--files` list of files to be uploaded with the application.

Source: Spark docs, Debugging

Steps:

Example `log4j.properties`:

```
# Blacklist all to warn level
log4j.rootCategory=WARN, 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}: %m%n

# Whitelist our app to info :)
log4j.logger.com.github.atais=INFO
```

Executing `spark-submit`, for cluster mode:

```
spark-submit \
    --master yarn \
    --deploy-mode cluster \
    --conf "spark.driver.extraJavaOptions=-Dlog4j.configuration=file:log4j.properties" \
    --conf "spark.executor.extraJavaOptions=-Dlog4j.configuration=file:log4j.properties"
\
    --files "/absolute/path/to/your/log4j.properties" \
    --class com.github.atais.Main \
    "SparkApp.jar"
```

Note that you must use `--driver-java-options` if using `client` mode. Spark docs, Runtime env

Executing `spark-submit`, for client mode:

```
spark-submit \
    --master yarn \
    --deploy-mode client \
    --driver-java-options "-
Dlog4j.configuration=file:/absolute/path/to/your/log4j.properties" \
    --conf "spark.executor.extraJavaOptions=-Dlog4j.configuration=file:log4j.properties"
\
    --files "/absolute/path/to/your/log4j.properties" \
    --class com.github.atais.Main \
    "SparkApp.jar"
```

Notes:

1. Files uploaded to `spark-cluster` with `--files` will be available at root dir, so there is no need to add any path in `file:log4j.properties`.
2. Files listed in `--files` must be provided with absolute path!
3. `file:` prefix in configuration URI is mandatory.

#3 Edit cluster's `conf/log4j.properties`

This changes **global logging configuration file**.

> update the `$SPARK_CONF_DIR/log4j.properties` file and it will be automatically uploaded along with the other configurations.

Source: Spark docs, Debugging

To find your `SPARK_CONF_DIR` you can use `spark-shell`:

```
atais@cluster:~$ spark-shell
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.1.1
      /_/

scala> System.getenv("SPARK_CONF_DIR")
res0: String = /var/lib/spark/latest/conf
```

Now just edit `/var/lib/spark/latest/conf/log4j.properties` (with example from method #2) and all your apps will share this configuration.

#4 Override configuration directory

If you like the solution #3, but want to customize it per application, you can actually copy `conf` folder, edit it contents and specify as the root configuration during `spark-submit`.

> To specify a different configuration directory other than the default `"SPARK_HOME/conf"`, you can set `SPARK_CONF_DIR`. Spark will use the configuration files (`spark-defaults.conf`, `spark-env.sh`, `log4j.properties`, *etc*) from this directory.

Source: Spark docs, Configuration

Steps:

1. Copy cluster's `conf` folder (more info, method #3)
2. Edit `log4j.properties` in that folder (example in method #2)
3. Set `SPARK_CONF_DIR` to this folder, before executing `spark-submit`, example:

```
export SPARK_CONF_DIR=/absolute/path/to/custom/conf

spark-submit \
    --master yarn \
```

```
--deploy-mode cluster \
--class com.github.atais.Main \
"SparkApp.jar"
```

Conclusion

I am not sure if there is any other method, but I hope this covers the topic from A to Z. If not, feel free to ping me in the comments!

Enjoy your way!

Share Edit Follow

edited May 12, 2021 at 18:47

Milimetric
13.5k ● 4 ● 48 ● 57

answered Apr 9, 2019 at 15:31

Atais
11.2k ● 7 ● 73 ● 115

2 This should be the accepted answer. It offers much details and sums up a lot more use cases than the others. (Without encouraging to disable the logs.) – belgacea Nov 7, 2019 at 16:24

@Atais - You should add below So if you are like me and find that the answers above didn't help, then maybe you too have to remove the '.template' suffix from your log4j conf file and then the above works perfectly! – oneday Jan 17, 2020 at 6:17

2 Additional note on the programmatic approach- The level has to be set before the SparkContext is created – Arunraj Nair Feb 28, 2020 at 23:11

@ArunrajNair should not be the case, because logging is a separate feature, not connected to SparkContext. – Atais Apr 28, 2020 at 8:36

53

Thanks @AkhlD and @Sachin Janani for suggesting changes in `.conf` file.

Following code solved my issue:

1) Added `import org.apache.log4j.{Level, Logger}` in import section

2) Added following line after creation of spark context object i.e. after `val sc = new SparkContext(conf)`:

```
val rootLogger = Logger.getRootLogger()
rootLogger.setLevel(Level.ERROR)
```

Share Edit Follow

edited Feb 17, 2016 at 9:13

Mark Butler
4,381 ● 2 ● 41 ● 39

answered Dec 16, 2015 at 7:31

Vishwas
7,057 ● 7 ● 45 ● 70

15 Tried this but still getting the logging outputs. – horatio1701d Jun 28, 2016 at 19:36

1 I like this solution as having no permission to access conf/ – Jim Ho Sep 26, 2016 at 6:39

this will change only the log level of the current java process, after Spark has been initialized – Thomas Decaux Aug 24, 2021 at 8:04

35

Use below command to change log level while submitting application using spark-submit or spark-sql:

```
spark-submit \
--conf "spark.driver.extraJavaOptions=-Dlog4j.configuration=file:<file path>/log4j.xml"
\
--conf "spark.executor.extraJavaOptions=-Dlog4j.configuration=file:<file
path>/log4j.xml"
```

Note: replace `<file path>` where `log4j` config file is stored.

Log4j.properties:

```
log4j.rootLogger=ERROR, console

# set the log level for these components
log4j.logger.com.test=DEBUG
log4j.logger.org=ERROR
log4j.logger.org.apache.spark=ERROR
log4j.logger.org.spark-project=ERROR
log4j.logger.org.apache.hadoop=ERROR
log4j.logger.io.netty=ERROR
log4j.logger.org.apache.zookeeper=ERROR

# add a ConsoleAppender to the logger stdout to write to the console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.layout=org.apache.log4j.PatternLayout
# use a simple message format
log4j.appender.console.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L -
%m%n
```

log4j.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">
   <appender name="console" class="org.apache.log4j.ConsoleAppender">
    <param name="Target" value="System.out"/>
    <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L
- %m%n" />
    </layout>
  </appender>
    <logger name="org.apache.spark">
        <level value="error" />
    </logger>
```

```
    <logger name="org.spark-project">
        <level value="error" />
    </logger>
    <logger name="org.apache.hadoop">
        <level value="error" />
    </logger>
    <logger name="io.netty">
        <level value="error" />
    </logger>
    <logger name="org.apache.zookeeper">
        <level value="error" />
    </logger>
   <logger name="org">
        <level value="error" />
    </logger>
    <root>
        <priority value ="ERROR" />
        <appender-ref ref="console" />
    </root>
</log4j:configuration>
```

Run code snippet Expand snippet

Switch to FileAppender in log4j.xml if you want to write logs to file instead of console. `LOG_DIR` is a variable for logs directory which you can supply using `spark-submit --conf "spark.driver.extraJavaOptions=-D`.

```
<appender name="file" class="org.apache.log4j.DailyRollingFileAppender">
        <param name="file" value="${LOG_DIR}"/>
        <param name="datePattern" value="'.'yyyy-MM-dd"/>
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%d [%t] %-5p %c %x - %m%n"/>
        </layout>
    </appender>
```

Run code snippet Expand snippet

Another important thing to understand here is, when job is launched in distributed mode (deploy-mode cluster and master as yarn or mesos) the log4j configuration file should exist on driver and worker nodes (`log4j.configuration=file:<file path>/log4j.xml`) else log4j init will complain-

> log4j:ERROR Could not read configuration file [log4j.properties].
> java.io.FileNotFoundException: log4j.properties (No such file or directory)

Hint on solving this problem-

Keep log4j config file in distributed file system(HDFS or mesos) and add external configuration using log4j PropertyConfigurator. or use sparkContext addFile to make it available on each node then use log4j PropertyConfigurator to reload configuration.

Share Edit Follow

edited Oct 12, 2018 at 3:19

answered May 2, 2017 at 22:01

Rahul Sharma

5,814 ● 10 ● 59 ● 95

2 This is one of the few examples that doesn't clobber all `org` logs that are traditionally from the default logger. – deepelement Aug 25, 2017 at 20:30

1 This works very well, but what is the `Log4j.properties` file for? It doesn't seem to be used. Are you simply documenting the properties set in the XML file? – vy32 May 27, 2018 at 3:41

1 You can use either of them. – Rahul Sharma May 28, 2018 at 5:18

3 I have had success with the above - I use `--files` in the spark-submit command to make log4j.properties available on all nodes. – Ben Watson Aug 14, 2018 at 12:23

1 This is the only solution that worked for me and it does not involve any code change. Create a file `Log4.properties` under `main/resources` in case that it does not exist – Yeikel Dec 25, 2018 at 23:52

19

You set disable the Logs by setting its level to OFF as follows:

```
Logger.getLogger("org").setLevel(Level.OFF);
Logger.getLogger("akka").setLevel(Level.OFF);
```

or edit log file and set log level to off by just changing the following property:

```
log4j.rootCategory=OFF, console
```

Share Edit Follow

edited Jan 6, 2015 at 5:10

answered Jan 5, 2015 at 14:13

Sachin Janani

1,319 ● 1 ● 17 ● 34

In which file do I set above properties?? – Vishwas Jan 6, 2015 at 4:48

You can add these lines in your Driver program @Vishwas – Sachin Janani Jan 6, 2015 at 4:49

I have added same but still logs appears on console – Vishwas Jan 6, 2015 at 5:52

Have you change the property log4j.rootCategory=OFF. I have tested these at my end and its working fine – Sachin Janani Jan 6, 2015 at 6:02

5 This makes zero difference for me on Spark 2.3.1 – Toby Eggitt Jun 29, 2018 at 18:04

I just add this line to all my pyspark scripts on top just below the import statements.

```
SparkSession.builder.getOrCreate().sparkContext.setLogLevel("ERROR")
```

15

example header of my pyspark scripts

```
from pyspark.sql import SparkSession, functions as fs
SparkSession.builder.getOrCreate().sparkContext.setLogLevel("ERROR")
```

Share Edit Follow

answered May 16, 2018 at 12:32

Gajendra D Ambi
4,187 ● 31 ● 33

20 This works, but it doesn't stop the 58 lines of INFO messages that appear during the creation of the Spark context. – vy32 May 27, 2018 at 3:40

1 Note that this is for Spark 2.x – Yeikel Jan 1, 2019 at 16:31

Because those messages are written by the Hadoop cluster, and not the spark session – Guarneer Oct 24, 2023 at 9:30

Answers above are correct but didn't exactly help me as there was additional information I required.

14

I have just setup Spark so the log4j file still had the '.template' suffix and wasn't being read. I believe that logging then defaults to Spark core logging conf.

So if you are like me and find that the answers above didn't help, then maybe you too have to remove the '.template' suffix from your log4j conf file and then the above works perfectly!

http://apache-spark-user-list.1001560.n3.nabble.com/disable-log4j-for-spark-shell-td11278.html

Share Edit Follow

answered May 30, 2015 at 6:00

curtisp
2,305 ● 3 ● 32 ● 65

Please check this answer, stackoverflow.com/a/51554118/2094086 hope you're looking for the same. – Gaurav Adurkar Jul 27, 2018 at 8:41

This works well. Just remove the .template from the log4j fle and set log4j.rootCategory=Error, console – Vishal Kamlapure Apr 21, 2022 at 15:06

Adding the following to the PySpark did the job for me:

6

```
self.spark.sparkContext.setLogLevel("ERROR")
```

self.spark is the spark session (`self.spark = spark_builder.getOrCreate()`)

Share Edit Follow

answered Nov 30, 2020 at 5:26

Farshad Javadi
419 ● 6 ● 7

5

tl;dr

> For Spark Context you may use:
>
> ```
> sc.setLogLevel(<logLevel>)
> ```
>
> where `loglevel` can be ALL, DEBUG, ERROR, FATAL, INFO, OFF, TRACE or WARN.

Details-

Internally, `setLogLevel` calls `org.apache.log4j.Level.toLevel(logLevel)` that it then uses to set using `org.apache.log4j.LogManager.getRootLogger().setLevel(level)`.

> You may directly set the logging levels to `OFF` using:
>
> ```
> LogManager.getLogger("org").setLevel(Level.OFF)
> ```

You can set up the default logging for Spark shell in `conf/log4j.properties`. Use `conf/log4j.properties.template` as a starting point.

Setting Log Levels in Spark Applications

In standalone Spark applications or while in Spark Shell session, use the following:

```
import org.apache.log4j.{Level, Logger}

Logger.getLogger(classOf[RackResolver]).getLevel
Logger.getLogger("org").setLevel(Level.OFF)
Logger.getLogger("akka").setLevel(Level.OFF)
```

Disabling logging(in log4j):

Use the following in `conf/log4j.properties` to disable logging completely:

```
log4j.logger.org=OFF
```

Reference: Mastering Spark by Jacek Laskowski.

Share Edit Follow

answered Nov 12, 2017 at 6:41

Ani Menon

28k ● 16 ● 109 ● 127

5

In Python/Spark we can do:

```
def quiet_logs( sc ):
  logger = sc._jvm.org.apache.log4j
  logger.LogManager.getLogger("org"). setLevel( logger.Level.ERROR )
  logger.LogManager.getLogger("akka").setLevel( logger.Level.ERROR )
```

The after defining Sparkcontaxt 'sc' call this function by : quiet_logs(sc)

Share Edit Follow

answered Apr 4, 2016 at 21:22

Anupam Mahapatra

827 ● 9 ● 4

I'd love to find a programatic way that works without messing with the log4j file -- but when I try that, I still get warnings like WARN org.apache.spark.scheduler.TaskSetManager: Lost task 612.1 in stage 0.0 (TID 2570 ..., executor 15): TaskKilled (another attem – MrCartoonology Sep 4, 2019 at 18:29

5

Simply add below param to your spark-shell OR spark-submit command

```
--conf "spark.driver.extraJavaOptions=-Dlog4jspark.root.logger=WARN,console"
```

Check exact property name (log4jspark.root.logger here) from log4j.properties file. Hope this helps, cheers!

Share Edit Follow

edited Dec 16, 2018 at 20:13

answered Jul 27, 2018 at 8:40

Gaurav Adurkar

864 ● 8 ● 9

3 To set this from the command-line would have be awesome. But this didn't work for me. – swdev Mar 20, 2019 at 2:51

4

Simple to do on the command line...

`spark2-submit --driver-java-options="-Droot.logger=ERROR,console"` ..other options..

Share Edit Follow

edited Mar 2, 2018 at 22:16

dkackman

15.5k ● 14 ● 72 ● 127

answered Mar 2, 2018 at 21:53

1472580

163 ● 1 ● 8

what is `spark2-submit` ? – vy32 May 27, 2018 at 3:40

spark2-submit is used for Spark2. – Nephilim Jul 13, 2018 at 5:47

2

An interesting idea is to use the RollingAppender as suggested here: http://shzhangji.com/blog/2015/05/31/spark-streaming-logging-configuration/ so that you don't "polute" the console space, but still be able to see the results under $YOUR_LOG_PATH_HERE/${dm.logging.name}.log.

```
    log4j.rootLogger=INFO, rolling

log4j.appender.rolling=org.apache.log4j.RollingFileAppender
log4j.appender.rolling.layout=org.apache.log4j.PatternLayout
log4j.appender.rolling.layout.conversionPattern=[%d] %p %m (%c)%n
log4j.appender.rolling.maxFileSize=50MB
log4j.appender.rolling.maxBackupIndex=5
log4j.appender.rolling.file=$YOUR_LOG_PATH_HERE/${dm.logging.name}.log
log4j.appender.rolling.encoding=UTF-8
```

Another method that solves the cause is to observe what kind of loggings do you usually have (coming from different modules and dependencies), and set for each the granularity for the logging, while turning "quiet" third party logs that are too verbose:

For instance,

```
    # Silence akka remoting
log4j.logger.Remoting=ERROR
log4j.logger.akka.event.slf4j=ERROR
log4j.logger.org.spark-project.jetty.server=ERROR
log4j.logger.org.apache.spark=ERROR
log4j.logger.com.anjuke.dm=${dm.logging.level}
log4j.logger.org.eclipse.jetty=WARN
log4j.logger.org.eclipse.jetty.util.component.AbstractLifeCycle=ERROR
log4j.logger.org.apache.spark.repl.SparkIMain$exprTyper=INFO
log4j.logger.org.apache.spark.repl.SparkILoop$SparkILoopInterpreter=INFO
```

Share Edit Follow

edited Oct 13, 2015 at 15:31

answered Oct 13, 2015 at 8:24


marilena.oita
984 ● 8 ● 14

2

for Spark 3.5.0: go to `/spark-3.5.0-bin-hadoop3/conf` folder and run

```
cp log4j2.properties.template log4j2.properties
```

Then in the new file modify `rootLogger.level` from `info` to `warn`

Share Edit Follow

answered Nov 16, 2023 at 14:03


Galuoises
3,233 ● 31 ● 44

1

If you don't have the ability to edit the java code to insert the `.setLogLevel()` statements and you don't want yet more external files to deploy, you can use a brute force way to solve this. Just filter out the INFO lines using grep.

```
spark-submit --deploy-mode client --master local <rest-of-cmd> | grep -v -F "INFO"
```

Share Edit Follow

answered Mar 20, 2019 at 2:39


swdev
3,051 ● 2 ● 26 ● 38

0

If anyone else is stuck on this,

nothing of the above worked for me. I had to remove

```
implementation group: "ch.qos.logback", name: "logback-classic", version: "1.2.3"
implementation group: 'com.typesafe.scala-logging', name: "scala-logging_$scalaVersion",
version: '3.9.2'
```

from my build.gradle for the logs to disappear. TLDR: Don't import any other logging frameworks, you should be fine just using `org.apache.log4j.Logger`

Share Edit Follow

answered Jul 19, 2019 at 18:07


Falco Winkler
1,170 ● 1 ● 18 ● 24

---

1. Adjust conf/log4j.properties as described by other `log4j.rootCategory=ERROR, console`

0

2. Make sure while executing your spark job you pass `--file` flag with log4j.properties file path
3. If it still doesn't work you might have a jar that has log4j.properties that is being called before your new log4j.properties. Remove that log4j.properties from jar (if appropriate)

Share Edit Follow

edited Jun 3, 2023 at 18:19
Koedlt
5,893 ● 9 ● 21 ● 45

answered Jan 1, 2016 at 22:28
user5736891

0

Another way of stopping logs completely is:

```
import org.apache.log4j.Appender;
import org.apache.log4j.BasicConfigurator;
import org.apache.log4j.varia.NullAppender;

public class SomeClass {

    public static void main(String[] args) {
        Appender nullAppender = new NullAppender();
        BasicConfigurator.configure(nullAppender);

        {...more code here...}

    }
}
```

This worked for me. An NullAppender is

> An Appender that ignores log events. (https://logging.apache.org/log4j/2.x/log4j-core/apidocs/org/apache/logging/log4j/core/appender/NullAppender.html)

Share Edit Follow

answered Apr 21, 2020 at 12:16
gandalf
11 ● 3

0

In addition to all the above posts, here is what solved the issue for me.

Spark uses slf4j to bind to loggers. If log4j is not the first binding found, you can edit log4j.properties files all you want, the loggers are not even used. For example, this could be a possible SLF4J output:

> SLF4J: Class path contains multiple SLF4J bindings. SLF4J: Found binding in [jar:file:/C:/Users/~/.m2/repository/org/slf4j/slf4j-simple/1.6.6/slf4j-simple-1.6.6.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: Found binding in

> [jar:file:/C:/Users/~/.m2/repository/org/slf4j/slf4j-log4j12/1.7.19/slf4j-log4j12-1.7.19.jar!/org/slf4j/impl/StaticLoggerBinder.class] SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation. SLF4J: Actual binding is of type [org.slf4j.impl.SimpleLoggerFactory]

So here the SimpleLoggerFactory was used, which does not care about log4j settings.

Excluding the slf4j-simple package from my project via

```
<dependency>
        ...
        <exclusions>
            ...
            <exclusion>
                <artifactId>slf4j-simple</artifactId>
                <groupId>org.slf4j</groupId>
            </exclusion>
        </exclusions>
    </dependency>
```

resolved the issue, as now the log4j logger binding is used and any setting in log4j.properties is adhered to. F.Y.I. my log4j properties file contains (besides the normal configuration)

```
log4j.rootLogger=WARN, stdout
...
log4j.category.org.apache.spark = WARN
log4j.category.org.apache.parquet.hadoop.ParquetRecordReader = FATAL
log4j.additivity.org.apache.parquet.hadoop.ParquetRecordReader=false
log4j.logger.org.apache.parquet.hadoop.ParquetRecordReader=OFF
```

Hope this helps!

Share Edit Follow

answered May 14, 2018 at 10:23

daniel.wirtz
1,030 ● 1 ● 9 ● 11

this work for me. For anyone who use sbt just add "exclude("ch.qos.logback", "logback-classic")" – Hoang Minh Quang FX15045 Jun 6, 2023 at 2:24

0

This one worked for me. For only ERROR messages to be displayed as `stdout`, `log4j.properties` file may look like:

```
# Root logger option
log4j.rootLogger=ERROR, stdout
# Direct log messages to stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
```

```
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n
```

> NOTE: Put `log4j.properties` file in `src/main/resources` folder to be effective. And if `log4j.properties` doesn't exist (meaning `spark` is using `log4j-defaults.properties` file) then you can create it by going to `SPARK_HOME/conf` and then `mv log4j.properties.template log4j.properties` and then proceed with above said changes.

Share Edit Follow

answered Feb 22, 2019 at 6:48

RPaul
169 ● 1 ● 5

```
sparkContext.setLogLevel("OFF")
```

0

Share Edit Follow

edited Dec 17, 2016 at 3:43

Danh
6,006 ● 7 ● 33 ● 45

answered Dec 16, 2016 at 15:53

Art
1,330 ● 13 ● 26

1 I did both- removed the .template suffix from log4j.properties and set the level to ERROR, and `val rootLogger = Logger.getRootLogger() rootLogger.setLevel(Level.ERROR)` It worked – Sam-T Jan 4, 2017 at 16:18