Writing Spark Applications

Writing Spark Applications

- In this chapter, you will learn
  - How to write, build, configure and run Spark applications

Chapter Topics

- Spark Applications Vs. Spark Shell
- Creating The SparkContext
- Building A Spark Application (Scala And Java)
- Running A Spark Application
- Hands-On Exercise: Writing And Running A Spark Application
- Configuring Spark Properties
- Logging
- Examples
- Summary
- Review
- References
- Hands-On Exercise: Setting Log Levels

Spark Shell Vs Spark Applications

- The Spark Shell allows interactive exploration and manipulation of data
  - REPL using Python or Scala
  - But most end users don't use a shell to interact with their data
    - They use applications
- Spark applications run as independent programs
  - Python, Scala, or Java
  - e.g., ETL processing, Streaming, and so on
  - Suitable for batch jobs, long running applications, or "automated" jobs that don't require human interaction
- Technically, you can think of a shell session in Python or Scala as a Spark application

Chapter Topics

- Spark Applications Vs. Spark Shell
- Creating The SparkContext
- Building A Spark Application (Scala And Java)
- Running A Spark Application
- Hands-On Exercise: Writing And Running A Spark Application
- Configuring Spark Properties
- Logging
- Examples
- Summary
- Review
- References
- Hands-On Exercise: Setting Log Levels

The SparkContext

- Every Spark program needs a SparkContext
  - The interactive shell creates one for you
    - Named sc
  - You create your own in a Spark application
    - Named sc by convention
    - Call sc.stop() at the end of your application
      - Use this in a clustered environment to prevent the application from hanging

Example: WordCount As A Python Application

```
import sys
from pyspark import SparkContext

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print >> sys.stderr, "Usage: WordCount <file>"
        exit(-1)

sc = SparkContext()

counts = sc.textFile(sys.argv[1]) \
    .flatMap(lambda line: line.sp1it()) \
    .map(lambda word: (word, 1)) \
    .reduceByKey(lambda v1, v2: v1 + v2)

for pair in counts.take(5):
    print pair

sc.stop()
```

Example: WordCount As A Scala Application

```
import org.apache.spark.SparkContext   // Get Context
import org.apache.spark.SparkContext._ // Get Interfaces

object WordCount {
    def main(args: Array[String]) {
        if (args.length < 1) {
            System.err.println("Usage: WordCount <file>")
            System.exit(1)
        }

        val sc = new SparkContext()

        val counts = { sc.textFile(args(0))
            .flatMap(line => line.split("\\W"))
            .map(word => (word, 1))
            .reduceByKey(_ + _) }

        counts.take(5).foreach(println)

        sc.stop()
    }
```

Chapter Topics

- Spark Applications Vs. Spark Shell
- Creating The SparkContext
- Building A Spark Application (Scala And Java)
- Running A Spark Application
- Hands-On Exercise: Writing And Running A Spark Application
- Configuring Spark Properties
- Logging
- Examples
- Summary
- Review
- References
- Hands-On Exercise: Setting Log Levels

Building A Spark Application: Scala Or Java

- Scala or Java Spark applications must be compiled and assembled into JAR files
  - The application JAR file will be passed to worker nodes
- Most developers use Apache Maven to build the application
  - Maven is a package management tool for Java-based languages that lets you link to libraries in public repositories
  - http://spark.apache.org/docs/latest/building-with-maven.html
  - http://blog.cloudera.com/blog/2014/04/how-to-run-a-simple-apache-spark-app-in-cdh-5/
- Build details will differ depending on
  - Version of Hadoop (HDFS)
  - Deployment platform (Spark Standalone, YARN, Mesos)
- Development options include
  - IDEs (e.g. IntelliJ, Eclipse)
  - sbt (Simple Build Tool for Scala) installed in a JAR file on your classpath (not included with CDH)
    - Similar to Maven but with features like incremental compilation and an interactive shell
    - http://www.scala-sbt.org/
    - http://stackoverflow.com/tags/sbt

A Simple Maven Build File For A Spark Application Written In Java

```
<project>                                                    <!-- From the book: Learning Spark -->
    <groupId>com.oreilly.learningsparkexamples.mini</groupId>
    <artifactId>learning-spark-mini-example</artifactId>
    <modelVersion>4.0.0</modelVersion>
    <name>example</name>
    <packaging>jar</packaging>
    <version>0.0.1</version>
    <dependencies>
        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-core_2.10</artifactId>
            <version>1.2.0</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
    <properties>
        <java.version>1.6</java.version>
    </properties>
    <build>
        <pluginManagement>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <version>3.1</version>
                    <configuration>
                        <source>${java.version}</source>
                        <target>${java.version}</target>
                    </configuration>
                </plugin>
            </plugins>
```

Chapter Topics

- Spark Applications Vs. Spark Shell
- Creating The SparkContext
- Building A Spark Application (Scala And Java)
- Running A Spark Application
- Hands-On Exercise: Writing And Running A Spark Application
- Configuring Spark Properties
- Logging
- Examples
- Summary
- Review
- References
- Hands-On Exercise: Setting Log Levels

Running A Spark Application (1 Of 2)

- The easiest way to run a Spark Application is by using Apache's spark-submit script from the command line
  - Python:

```
$ spark-submit WordCount.py fileURL
```

  - Works with all cluster managers
    - Spark Standalone, Hadoop YARN, Apache Mesos

```
$ spark-submit --class WordCount MyJarFile.jar fileURL
```

  - In both examples above, fileURL is a reference to the input file location in the form of a URL
    - file:
    - hdfs:

Running A Spark Application (2 Of 2)

- Some key spark-submit options
  - --help
    - Explain available options
  - --master URL
    - Equivalent to MASTER environment variable for the Spark Shell for specifying the Master node
    - local[*]
      - Run locally with as many threads as cores (default)
    - local[n]
      - Run locally with n threads
    - local
      - Run locally with a single thread
    - master URL format:
      - e.g. spark://masternode:7077
  - --deploy-mode
    - Either client or cluster, indicating where the driver will run
  - --name
    - Application name to display in the UI (default is the Scala/Java class or Python program name)
  - --jars
    - Additional JAR files (Scala/Java only)
  - --pyfiles
    - Additional Python files (Python only)
  - --driver-java-options
    - Parameters to pass to the driver JVM
      - e.g. -D ...

Spark Cluster Options

- Spark can run
  - Locally on a single machine, in a single JVM
    - No distributed processing
    - Can use the local drive or HDFS
    - Analogous to local mode in Hadoop
    - Good For testing
  - On a cluster
    - Single use cluster
      - Spark Standalone
      - Dedicated Spark cluster
    - Mixed use cluster
      - Apache Hadoop YARN (Yet Another Resource Negotiator)
      - Apache Mesos
      - Both capable of managing resources across a cluster using both Spark and Hadoop
    - Other options may be available in the future

Supported Cluster Resource Managers

- Hadoop YARN
  - https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.htm
  - Included in CDH
  - Most common for production sites
  - Scalable to 7000+ nodes
  - Very configurable
  - Capable of sharing cluster resources across platforms (MapReduce, Impala, etc.)
- Spark Standalone
  - Included with Spark
  - Easy to install and run
  - Limited configurability and scalability
    - About 12 configurable properties (as opposed to over 200 for Hadoop YARN)
    - Does not share cluster resources dynamically with other applications (the way YARN does)
      - e.g. You cannot deploy both Spark and Hadoop applications in Spark Standalone, as they do not play well together
  - Useful for testing, development, or small production systems
- Apache Mesos
  - https://mesos.apache.org/
  - First platform supported by Spark
  - Scalable to “10,000s of nodes"
  - Also capable of sharing cluster resources across platforms (MapReduce, Impala, etc.)
  - Now used less often
  - Mesos is used by Apple (Siri), Twitter, Airbnb, Google, PayPal and OpenTable
    - http://www.slideshare.net/caniszczyk/apache-mesos-at-twitter-texas-linuxfest-2014

Running A Spark Application Locally

- Use spark-submit --master to specify configuration options
  - Local options
    - local[*]
      - Run locally with as many threads as cores (default)
    - local[n]
      - Run locally with n threads
    - local
      - Run locally with a single thread
- Python

```
$ spark-submit --master local[3] WordCount.py fileURL
```

```
$ spark-submit --master local[3] \
      --class WordCount MyJarFile.jar fileURL
```

Running A Spark Application On A Cluster

- Use spark-submit --master to specify configuration options
  - Cluster options
    - yarn-client
      - Run on YARN in Client Mode
    - yarn-cluster
      - Run on YARN in Cluster Mode
    - spark://masternode:port
      - Run on a cluster using Spark Standalone
    - mesos://masternode:port
      - Run on a cluster using Mesos
- Python

```
$ spark-submit --master yarn-cluster \
      WordCount.py fileURL
```

```
$ spark-submit --yarn-cluster \
      --class WordCount MyJarFile.jar fileURL
```

Chapter Topics

- Spark Applications Vs. Spark Shell
- Creating The SparkContext
- Building A Spark Application (Scala And Java)
- Running A Spark Application
- Hands-On Exercise: Writing And Running A Spark Application
- Configuring Spark Properties
- Logging
- Examples
- Summary
- Review
- References
- Hands-On Exercise: Setting Log Levels

Building And Running Scala Applications In The Hands-On Exercises

- Basic Maven projects are provided in the exercises/projects directory with two packages
  - --stubs - Starter Scala file, do your work here
  - --solution - Final exercise solution
  - Make sure the package name matches the directory you run with

Command Line	Directory Structure

`$ mvn package`
`$ spark-submit \`
`  --class stubs.CountJPGs \`
`  target/countjpgs-1.0.jar \`
`  weblogs.*`
	`+countjpgs`
	`  —pom.xml`
	`  +src`
	`  +main`
	`  +scala`
	`  +solution`
	`  -CountJPGs.scala`
	`  +stubs`
	`  —CountJPGs.scala`
	`  +target`
	`  -countjpgs-1.0.jar`

Hands-On Exercise

- Writing And Running A Spark Application
  - Write and run a Spark application to count JPG requests in a web server log
- Please refer to the Hands-On Exercise Manual

Chapter Topics

- Spark Applications Vs. Spark Shell
- Creating The SparkContext
- Building A Spark Application (Scala And Java)
- Running A Spark Application
- Hands-On Exercise: Writing And Running A Spark Application
- Configuring Spark Properties
- Logging
- Examples
- Summary
- Review
- References
- Hands-On Exercise: Setting Log Levels

Spark Application Configuration Properties

- Spark provides numerous properties for configuring your application, some of which we have already seen set from the command line
- Example properties
  - spark.master
    - URI of the master node
  - spark.app.name
    - Application name (displayed in the UI)
  - spark.local.dir
    - Storage location for local files
      - Default: /tmp
  - spark.ui.port
    - Port on which to run the Spark Application Ul
      - Default: 4040
  - spark.executor.memory
    - The amount of memory to allocate to each Executor
      - Default: 512MB
- Most are more interesting to system administrators than developers
  - You can view the properties by selecting the Environment tab for any running application in the Spark Application UI (Later)
    - http://localhost:4040/Environment
  - OR
  - http://spark.apache.org/docs/latest/configuration.html

Spark Application Configuration

- Spark applications can be configured
  - Programmatically (statically/compile time)
    - Specified at compile time in the source code
    - Executes at runtime
    - Highest precedence
  - Declaratively (dynamically/run time)
    - Specified at runtime on the command line or via a properties file declared on the command line
    - Executes at runtime
    - Second highest precedence
  - Default
    - Do nothing, and Spark looks for the site-wide default configuration file to configure the application
      - $SPARK_HOME/conf/spark-defaults.conf
      - CDH only
      - Third highest precedence
    - If none is provided, Spark's built-in defaults are used
      - Lowest precedence

Runtime Configuration Options

- spark-submit script
  - e.g., spark-submit --master spark://masternode:7077
- Properties file
  - Tab or space-separated list of properties and values
    - Not XML
  - Or load with spark-submit --properties-file filename
  - Sample properties file

```
spark.master        spark://masternode:7077
spark.local.dir     /tmp
spark.ui.port       4444
```

Setting Configuration Properties Programmatically

- Spark configuration settings are part of the SparkContext
- Configure them using a SparkConf object
- Functions available on the SparkConf object include:
  - setAppName(name)
  - setMaster(master)
  - set(property-name, value)
- Set functions return a SparkConf object to support chaining

Python Programmatic Configuration

- Note the (temporarily) required bug workaround
  - https://issues.apache.org/jira/browse/SPARK-2003

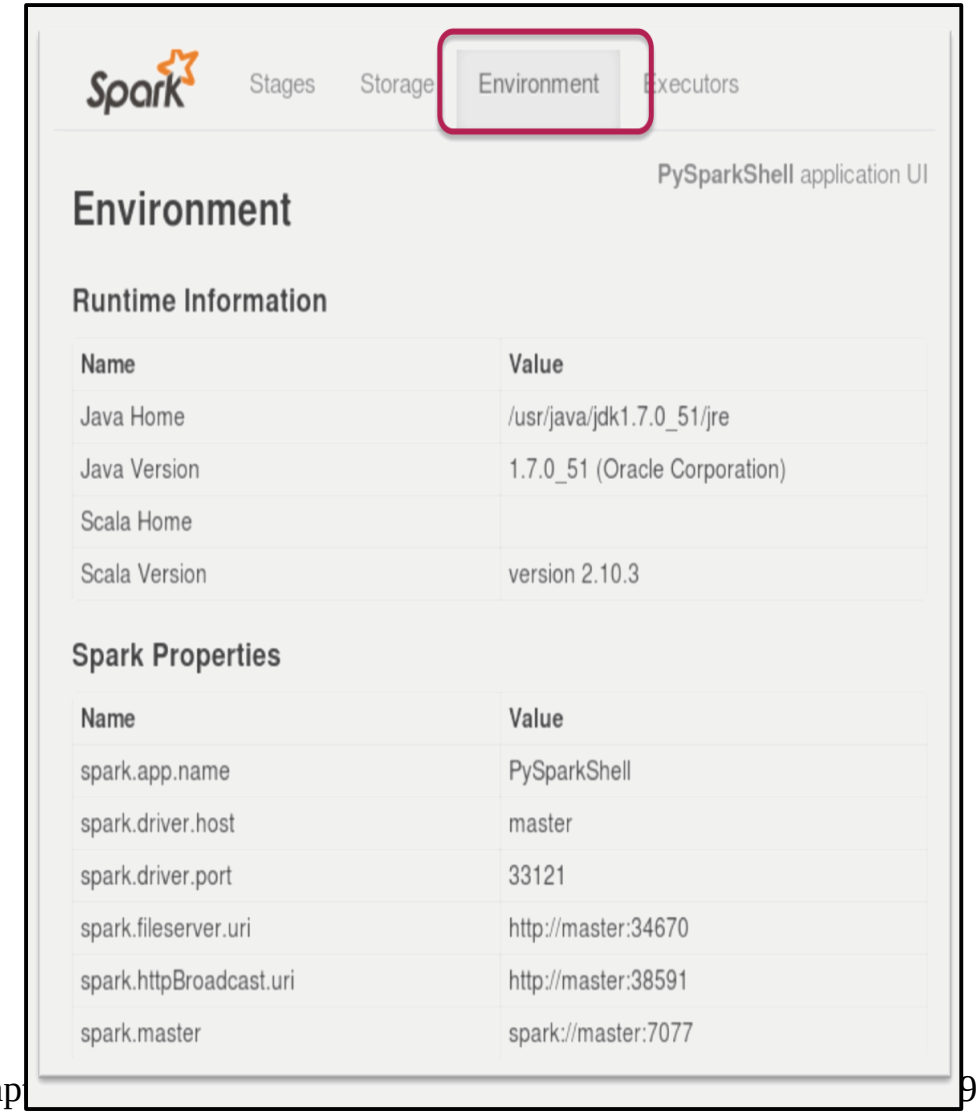
```
import sys
from pyspark import SparkContext
from pyspark import SparkConf
if __name__ == "__main__":
    if len(sys.argv) < 2:
        print >> sys.stderr, "Usage: WordCount <file>"
        exit(-1)
    sconf = SparkConf() \
        .setAppName("Word Count") \
        .set("spark.ui.port","4141")
    sc = SparkContext(conf=sconf) // Bug Workaround
    counts = sc.textFile(sys.argv[1]) \
        .flatMap(lambda line: line.split()) \
        .map(lambda w: (w, 1)) \
        .reduceByKey(lambda v1, v2: v1 + v2)
    for pair in counts.take(5):
        print pair
    sc.stop()
```

Scala Programmatic Configuration

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.SparkConf
object WordCount {
    def main(args: Array[String]) {
        if (args.length < 1) {
            System.err.println("Usage: WordCount <file>")
            System.exit(1)
        }
        val sconf = new SparkConf()
            .setAppName("Word Count")
            .set("spark.ui.port","4141")
        val sc = new SparkContext(sconf)
        val counts = sc.textFile(args(0)) {
            .flatMap(line => line.split("\\W"))
            .map(word => (word, 1))
            .reduceByKey(_ + _) }
        counts.take(5).foreach(println)
        sc.stop()
    }
}
```

Viewing Spark Properties

- View the environment while an application is running (including environment properties) from the Environment tab of the Spark Application UI



Chapter Topics

- Spark Applications Vs. Spark Shell
- Creating The SparkContext
- Building A Spark Application (Scala And Java)
- Running A Spark Application
- Hands-On Exercise: Writing And Running A Spark Application
- Configuring Spark Properties
- Logging
- Examples
- Summary
- Review
- References
- Hands-On Exercise: Setting Log Levels

Spark Logging

- Spark uses Apache Log4j for logging
  - Allows you to control logging at runtime using a properties file, in which you can
    - Enable or disable logging
    - Set logging levels
    - Select output destination
    - Format the output
  - http://logging.apache.org/log4j/1.2/
- Log4j provides several logging levels
  - Off
  - Fatal
  - Error
  - Warn
  - Info
  - Debug
  - Trace
  - All
  - For any level you choose other than Off, events of that level of severity or greater (higher in the list) will be logged

Spark Log Files

- Log file locations depend on your cluster management platform
- Spark Standalone defaults:
  - Spark daemons: /var/log/spark
  - Individual tasks (Executors): $SPARK_HOME/work on each worker node
  - Note that YARN has a nice log aggregation tool that allows you to view the logs of all worker nodes collectively, but currently Spark Standalone does not support this feature
    - Eliminates the need to have to drill down using the web UI to locate the log for a particular worker node to resolve a problem with that node

Spark Worker UI – Log File Access

Spark Worker at ip-10-236-129-42.ec2.internal:60105

ID: worker-20140121065745-ip-10-236-129-42.ec2.internal-60105
Master URL: spark://ec2-23-20-24-104.compute-1.amazonaws.com:7077
Cores: 4 (4 Used)
Memory: 13.6 GB (12.6 GB Used)

Back to Master

Running Executors 1

ExecutorID	Cores	Memory	Job Details	Logs
4	4	12.6 GB	**ID:** app-20140121220135-0003 **Name:** PageRank **User:** root	stdout stderr

Configuring Spark Logging

- Logging levels can be set for the cluster, for individual applications, or even for specific components or subsystems within an application
- For any machine, copy the log4j.properties.template to a directory in the classpath of the machine being logged
  - $SPARK_HOME/conf/log4j.properties
  - Edit as required

```
# Set everything to be logged to the console
log4j.rootCategory=DEBUG, console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
...
```

Chapter Topics

- Spark Applications Vs. Spark Shell
- Creating The SparkContext
- Building A Spark Application (Scala And Java)
- Running A Spark Application
- Hands-On Exercise: Writing And Running A Spark Application
- Configuring Spark Properties
- Logging
- Examples
- Summary
- Review
- References
- Hands-On Exercise: Setting Log Levels

Examples

- Your virtual machine comes configured with many
  - Example programs
    - /usr/lib/spark/examples/lib/spark-examples_2.10-1.0.0-cdh5.1.2-sources.jar
    - https://spark.apache.org/examples.html
    - http://spark.apache.org/docs/latest/index.html
    - https://sungsoo.github.io/2014/04/24/run-a-simple-apache-spark-app-in-cdh5.html
    - View (inside spark-examples_2.10-1.0.0-cdh5.1.2-sources.jar) /org/apache/spark/examples/SparkPi.scala
  - Simple shell scripts to compile and run the applications using maven

```
$ cd /usr/lib/spark
$ ./bin/run-example SparkPi 10
```

You would not typically use them in production

Chapter Topics

- Spark Applications Vs. Spark Shell
- Creating The SparkContext
- Building A Spark Application (Scala And Java)
- Running A Spark Application
- Hands-On Exercise: Writing And Running A Spark Application
- Configuring Spark Properties
- Logging
- Examples
- Summary
- Conclusion
- Review
- Hands-On Exercise: Setting Log Levels

Summary

- Use the Spark Shell application for interactive data exploration
- Write a Spark application to run independently
- Spark applications require a Spark Context object
- Spark applications are run using the spark-submit script
- Spark configuration parameters can be set at runtime using the spark-submit script or programmatically using a SparkConf object
- Spark uses log4j for logging
  - Configure using a log4j.properties file
- Spark is designed to run on a cluster
  - Spark includes a basic cluster management platform called Spark Standalone
  - But can also run on Hadoop YARN and Mesos
- The master distributes tasks to individual workers in the cluster
  - Tasks run in executors, as JVMs running on worker nodes
- Spark clusters work closely with HDFS
  - Tasks are assigned to workers storing the data whenever possible

Chapter Topics

- Spark Applications Vs. Spark Shell
- Creating The SparkContext
- Building A Spark Application (Scala And Java)
- Running A Spark Application
- Hands-On Exercise: Writing And Running A Spark Application
- Configuring Spark Properties
- Logging
- Examples
- Summary
- Review
- References
- Hands-On Exercise: Setting Log Levels

Review

- A SparkContext is provided for you by the Spark shell. But when writing a Spark application, you must create your own. True or False?

Review Answer

- A SparkContext is provided for you by the Spark shell. But when writing a Spark application, you must create your own. True or False?

Review

- Spark based applications written in Scala or Java must be compiled into a JAR file before they can be executed. True or False?

Review Answer

- Spark based applications written in Scala or Java must be compiled into a JAR file before they can be executed. True or False?

Review

- The program used to execute a Spark application is named _____

Review Answer

- The program used to execute a Spark application is named spark-submit

Review

- List 3 spark-submit options

Review Answer

- List 3 spark-submit options
  - --help
    - Explain available options
  - --master
    - Equivalent to MASTER environment variable for the Spark Shell
    - local[*]
      - Run locally with as many threads as cores (default)
    - local [n]
      - Run locally with n threads
    - local
      - Run locally with a single thread
    - master URL format
      - e.g. spark://masternode:7077
  - --deploy-mode
    - Either client or cluster
  - --name
    - Application name to display in the UI (default is the Scala/Java class or Python program name)
  - --jars
    - Additional JAR files (Scala/Java only)
  - --pyfiles
    - Additional Python files (Python only)
  - --driver-java-options
    - Parameters to pass to the driver JVM

Review

- What must you do to configure a specific Spark worker node for logging

Review Answer

- What must you do to configure a specific Spark worker node for logging
  - For or any machine, copy the log4j.properties.template to a directory in the classpath of the machine being logged
    - $SPARK_HOME/conf/log4j.properties
    - Edit as required

Chapter Topics

- Spark Applications Vs. Spark Shell
- Creating The SparkContext
- Building A Spark Application (Scala And Java)
- Running A Spark Application
- Hands-On Exercise: Writing And Running A Spark Application
- Configuring Spark Properties
- Logging
- Examples
- Summary
- Review
- References
- Hands-On Exercise: Setting Log Levels

References

- http://www.scala-sbt.org/documentation.html
- https://spark.apache.org/docs/latest/quick-start.html#standalone-applications
- http://spark.apache.org/docs/1.3.1/runningKonKmesos.html

Chapter Topics

- Spark Applications Vs. Spark Shell
- Creating The SparkContext
- Building A Spark Application (Scala And Java)
- Running A Spark Application
- Hands-On Exercise: Writing And Running A Spark Application
- Configuring Spark Properties
- Logging
- Examples
- Summary
- Conclusion
- Review
- Hands-On Exercise: Setting Log Levels

Hands-On Exercise:

- In this exercise you will
  - Set properties using spark-submit
  - Set properties in a property file
  - Change the logging levels in a log4j.properties file
- Please refer to the Hands-On Exercise Manual
-