Hi. Welcome to the Spark Fundamentals course. This lesson will cover Spark configuration, monitoring and tuning.

Objectives:
After completing this lesson, you should be able describe the cluster overview. Configure Spark by modifying Spark properties, environmental variables or logging properties. Monitor Spark and its applications using the Web UIs, metrics and various other external tools. Also covered in this lesson would be some performance tuning considerations.

Slide 3:
There are three main components of a Spark cluster. You have the driver, where the SparkContext is located within the main program. To run on a cluster, you would need some sort of cluster manager. This could be either Spark's standalone cluster manager, Mesos or Yarn. Then you have your worker nodes where the executor resides. The executors are the processes that run computations and store the data for the application. The SparkContext sends the application, defined as JAR or Python files to each executor. Finally, it sends the tasks for each executor to run.

Several things to understand about this architecture.
Each application gets its own executor. The executor stays up for the entire duration of the application. The benefit of this is that the applications are isolated from each other, on the scheduling side and running on different JVMs. However, this means that you cannot share data across applications. You would need to externalize the data if you wish to share data between thedifferent applications.

Spark applications don't care about the underlying cluster manager. As long as it can acquire executors and communicate with each other, it can run on any cluster manager.

Because the driver program schedules tasks on the cluster, it should run close to the worker nodes on the same local network. If you like to send remote requests to the cluster, it is better to use a RPC and have it submit operations from nearby.

There are currently three supported cluster managers that we have mentioned before. Sparks comes with a standalone manager that you can use to get up and running. You can use Apache Mesos, a general cluster manager that can run and service Hadoop jobs. Finally, you can also use Hadoop YARN, the resource manager in Hadoop 2. In the lab exercise, you will be using BigInsights with Yarn to run your Spark applications.

Slide 4:
Spark configuration.
There are three main locations for Spark configuration. You have the Spark properties, where the application parameters can be set using the SparkConf object or through Java system properties.

Then you have the environment variables, which can be used to set per machine settings such as IP address. This is done through the conf/spark-env.sh script on each node.

Finally, you also have your logging properties, which can be configured through log4j.properties. You can choose to override the default configuration directory, which is currently under the SPARK_HOME/conf directory. Set the SPARK_CONF_DIR environment variable and provide your custom configuration files under that directory.

In the lab exercise, the Spark shell can be verbose, so if you wish, change it from INFO to ERROR in the log4j.properties to reduce all the information being printed on the console.

Slide 5:
There are two methods of setting Spark properties. The first method is by passing application properties via the SparkConf object. As you know, the SparkConf variable is used to create the SparkContext object. In the example shown on this slide, you set the master node as local, the appName as "CountingSheep", and you allow 1GB for each of the executor processes.

The second method is to dynamically set the Spark properties. Spark allows you to pass in an empty SparkConf when creating the SparkContext as shown on the slide.

You can then either supply the values during runtime by using the command line options --master or the --conf. You can see the list of options using the --help when executing the spark-submit script.
On the slide here, you give the app name of My App and telling it to run on the local system with four cores. You set the spark.shuffle.spill to false and the various java options at the end. Finally you supply the application JAR file after all the properties have been specified.

You can find a list of all the properties on the spark.apache.org website.

Another way to set Spark properties is to provide your settings inside the spark-defaults.conf file. The spark-submit script will read in the configurations from this file. You can view the Spark properties on the application web UI at the port 4040 by default.

One thing I'll add is that properties set directly on the SparkConf take highest precedence, then flags passed to spark-submit or spark-shell is second and finally options in the spark-defaults.conf file is the lowest priority.