Hi - Welcome to Spark Fundamentals. Introduction to Spark.

Objectives:
After completing this lesson, you should be able to explain the purpose of Spark and understand why and when you would use Spark. You should be able to list and describe the components of the Spark unified stack. You will be able to understand the basics of the Resilent Distributed Dataset, Spark's primary data abstraction. Then you will see how to download and install Spark standalone to test it out yourself. You will get an overview of Scala and Python to prepare for using the two Spark shells.

Slide 3:
There is an explosion of data. No matter where you look, data is everywhere. You get data from social media such as Twitter feeds, Facebook posts, SMS, and a variety of others. The need to be able to process those data as quickly as possible becomes more important than ever. How can you find out what your customers want and be able to offer it to them right away? You do not want to wait hours for a batch job to complete. You need to have it in minutes or less.

MapReduce has been useful, but the amount of time it takes for the jobs to run is no longer acceptable in most situations. The learning curve to writing a MapReduce job is also difficult as it takes specific programming knowledge and the know-how. Also, MapReduce jobs only work for a specific set of use cases. You need something that works for a wider set of use cases.

Apache Spark was designed as a computing platform to be fast, general-purpose, and easy to use. It extends the MapReduce model and takes it to a whole other level.

The speed comes from the in-memory computations. Applications running in memory allows for a much faster processing and response. Spark is even faster than MapReduce for complex applications on disks.

This generality covers a wide range of workloads under one system. You can run batch application such as MapReduce types jobs or iterative algorithms that builds upon each other. You can also run interactive queries and process streaming data with your application. In a later slide, you'll see that there are a number of libraries which you can easily use to expand beyond the basic Spark capabilities.

The ease of use with Spark enables you to quickly pick it up using simple APIs for Scala, Python and Java. As mentioned, there are additional libraries which you can use for SQL, machine learning, streaming, and graph processing. Spark runs on Hadoop clusters such as Hadoop YARN or Apache Mesos, or even as a standalone with its own scheduler.

Slide 4:
You may be asking, why would I want to use Spark and what would I use it for? As you know, Spark is related to MapReduce in a sense that it expands on its capabilities.

Like MapReduce, Spark provides parallel distributed processing, fault tolerance on commodity hardware, scalability, etc. Spark adds to the concept with aggressively cached in-memory distributed computing, low latency, high level APIs and stack of high level tools described on the next slide. This saves time and money.

There are two groups that we can consider here who would want to use Spark: Data Scientists and Engineers. You may ask, but aren't they similar? In a sense, yes, they do have overlapping skill sets, but for our purpose, we'll define data scientist as those who need to analyze and model the data to obtain insight. They would have techniques to transform the data into something they can use for data analysis. They will use Spark for its ad-hoc analysis to run interactive queries that will give them results immediately. Data scientists may also have experience using SQL, statistics, machine learning and some programming, usually in Python, MatLab or R. Once the data scientists have obtained insights on the data and later someone determines that there's a need develop a production data processing application, a web application, or some system to act upon the insight, the person called upon to work on it would be the engineers.

Engineers would use Spark's programming API to develop a system that implement business use cases. Spark parallelize these applications across the clusters while hiding the complexities of distributed systems programming and fault tolerance. Engineers can use Spark to monitor, inspect and tune applications.

For everyone else, Spark is easy to use with a wide range of functionality. The product is mature and reliable.