Hadoop Developer

Modality: Self-Paced Learning

Duration: 27 Hours

About this course:

This Hadoop Developer Training will help you get a detailed idea about Big Data and Hadoop. Some of the topics included are introduction to the Hadoop ecosystem, understanding of HDFS and MapReduce including MapReduce abstraction. Learn to install, implement various components of Hadoop like Pig, Hive, Flume, Sqoop and YARN.

The average salary for a Hadoop Developer is **$119,250** per year.

Course Objective:

- Learn the Hadoop Architecture and Hadoop basics for beginners
- Learn what is Hadoop, HDFS and MapReduce framework
- Write MapReduce programs and deploy Hadoop clusters
- Develop applications for Big Data using Hadoop Technology
- Develop YARN programs on the Hadoop 2.X version
- Work on Big Data analytics using Hive, Pig and YARN
- Integrate MapReduce and HBase to do advanced usage and Indexing
- Learn fundamentals of Spark framework and its working
- Understand RDD in Apache Spark
- Learn Hadoop development best practices
- Job scheduling using Oozie
- Prepare for the Cloudera Spark and Hadoop Developer Certification

Audience:

- Software Developers, analytics, BI, ETL, and data warehousing professionals
- Big Data Hadoop developers, architects and testing personnel

Prerequisite:

- You don’t need prior knowledge of Apache Hadoop.

Course Outline:

Why is Big Data a Big Deal

- The Big Data Paradigm
- Resource Material
- Serial vs Distributed Computing

- What is Hadoop?
- HDFS or the Hadoop Distributed File System
- MapReduce Introduced
- YARN or Yet Another Resource Negotiator

Installing Hadoop in a Local Environment

- Hadoop Install Modes
- Hadoop Standalone mode Install
- Hadoop Pseudo-Distributed mode Install

The MapReduce "Hello World"

- The basic philosophy underlying MapReduce
- Resource Material
- MapReduce - Visualized And Explained
- MapReduce - Digging a little deeper at every step
- "Hello World" in MapReduce
- Resource Material
- The Mapper
- The Reducer
- The Job

Run a MapReduce Job

- Get comfortable with HDFS
- Run your first MapReduce Job

Juicing your MapReduce - Combiners, Shuffle and Sort and The Streaming API

- Parallelize the reduce phase - use the Combiner
- Resource Material
- Not all Reducers are Combiners
- How many mappers and reducers does your MapReduce have?
- Parallelizing reduce using Shuffle And Sort
- MapReduce is not limited to the Java language - Introducing the Streaming API
- Python for MapReduce

HDFS and Yarn

- HDFS - Protecting against data loss using replication
- Resource Material
- HDFS - Name nodes and why they're critical
- HDFS - Checkpointing to backup name node information
- Yarn - Basic components
- Resource Material
- Yarn - Submitting a job to Yarn
- Yarn - Plug in scheduling policies

- Yarn - Configure the scheduler

MapReduce Customizations For Finer Grained Control

- Setting up your MapReduce to accept command line arguments
- Resource Material
- The Tool, ToolRunner and GenericOptionsParser
- Configuring properties of the Job object
- Customizing the Partitioner, Sort Comparator, and Group Comparator

Introducing Hive

- Hadoop and Hive Install

Hadoop and Hive Install

- Resource Material
- What is Hadoop?
- Resource Material
- HDFS or the Hadoop Distributed File System

Hive Basics

- Primitive Datatypes
- Resource Material
- Collections_Arrays_Maps
- Structs and Unions
- Create Table
- Insert Into Table
- Insert into Table 2
- Alter Table
- HDFS
- HDFS CLI - Interacting with HDFS
- Code-Along: Create Table
- Code-Along: Hive CLI

Built-in Functions

- Three types of Hive functions
- Resource Material
- The Case-When statement, the Size function, the Cast function
- The Explode function
- Code-Along: Hive Built - in functions

Sub-Queries

- Quirky Sub-Queries
- Resource Material

- More on subqueries: Exists and In
- Inserting via subqueries
- Code-Along: Use Subqueries to work with Collection Datatypes
- Views
- Resource Material

Partitioning

- Indices
- Resource Material
- Partitioning Introduced
- The Rationale for Partitioning
- How Tables are Partitioned
- Using Partitioned Tables
- Dynamic Partitioning: Inserting data into partitioned tables
- Code-Along: Partitioning

Bucketing

- Introducing Bucketing
- Resource Material
- The Advantages of Bucketing
- How Tables are Bucketed
- Using Bucketed Tables
- Sampling

Windowing

- Windowing Introduced
- Resource Material
- Windowing - A Simple Example: Cumulative Sum
- Windowing - A More Involved Example: Partitioning
- Windowing - Special Aggregation Functions

Understanding MapReduce

- The basic philosophy underlying MapReduce
- Resource Material
- MapReduce - Visualized and Explained
- MapReduce - Digging a little deeper at every step

Introduction to HBase

- The problem with distributed computing
- Resource Material
- Installing HBase
- Resource Material
- The Hadoop ecosystem

- The role of HBase in the Hadoop ecosystem
- How is HBase different from RDBMS?
- HBase Data Model
- Introducing CRUD operations
- HBase is different from Hive

CRUD operations using the HBase Shell

- Example1 - Creating a table for User Notifications
- Resource Material
- Example 2 - Inserting a row
- Resource Material
- Example 3 - Updating a row
- Example 4 - Retrieving a row
- Example 5 - Retrieving a range of rows
- Example 6 - Deleting a row
- Example 7 - Deleting a table

CRUD operations using the Java API

- Example 8 - Creating a table with HBaseAdmin
- Resource Material
- Example 9 - Inserting a row using a Put object
- Example 10 - Inserting a list of Puts
- Example 11 - Retrieving data - Get and Result objects
- Example 12 - A list of Gets
- Example 13 - Deleting a row
- Example 14 - A list of Deletes
- Example 15 - Mix and match with batch operations
- Example 16 - Scanning a range of rows
- Example 17 - Deleting a table

HBase Architecture

- HBase Architecture
- Resource Material

Advanced operations - Filters and Counters

- Example 18 - Filter by Row id - RowFilter
- Resource Material
- Example 19 - Filter by column value - SingleColumnValueFilter
- Example 20 - Apply multiple conditions - Filterlist
- Example 21 - Retrieve rows within a time range
- Example 22 - Atomically incrementing a value with Counters

Where does Pig fit in?

- Pig and the Hadoop ecosystem
- Resource Material
- Install and set up
- Resource Material
- How does Pig compare with Hive?
- Pig Latin as a data flow language
- Pig with HBase

Pig Basics

- Operating modes, running a Pig script, the Grunt shell
- Resource Material
- Loading data and creating our first relation
- Scalar data types
- Complex data types - The Tuple, Bag and Map
- Partial schema specification for relations
- Displaying and storing relations - The dump and store commands

Pig Operations And Data Transformations

- Selecting fields from a relation
- Resource Material
- Built-in functions
- Evaluation functions
- Using the distinct, limit and order by keywords
- Filtering records based on a predicate

Advanced Data Transformations

- Group by and aggregate transformations
- Combining datasets using Join
- Concatenating datasets using Union
- Generating multiple records by flattening complex fields
- Resource Material
- Using Co-Group, Semi-Join and Sampling records
- The nested Foreach command
- Debug Pig scripts using Explain and Illustrate

Optimizing Data Transformations

- Parallelize operations using the Parallel keyword
- Join Optimizations: Multiple relations join, large and small relation join
- Join Optimizations: Skew join and sort-merge join
- Common sense optimizations

Introduction to Spark

- What does Donald Rumsfeld have to do with data analysis?

- Resource Material
- Why is Spark so cool?
- An introduction to RDDs - Resilient Distributed Datasets
- Built-in libraries for Spark
- Installing Spark
- Resource Material
- The PySpark Shell
- Transformations and Actions
- See it in Action: Munging Airlines Data with PySpark - I
- Resource Material
- [For Linux/Mac OS Shell Newbies] Path and other Environment Variables

Resilient Distributed Datasets

- RDD Characteristics: Partitions and Immutability
- Resource Material
- RDD Characteristics: Lineage, RDDs know where they came from
- What can you do with RDDs?
- Create your first RDD from a file
- Average distance travelled by a flight using map() and reduce() operations
- Get delayed flights using filter(), cache data using persist()
- Average flight delay in one-step using aggregate()
- Frequency histogram of delays using countByValue()
- See it in Action: Analyzing Airlines Data with PySpark - II

Advanced RDDs: Pair Resilient Distributed Datasets

- Special Transformations and Actions
- Resource Material
- Average delay per airport, use reduceByKey(), mapValues() and join()
- Average delay per airport in one step using combineByKey()
- Get the top airports by delay using sortBy()
- Lookup airport descriptions using lookup(), collectAsMap(), broadcast()
- See it in Action: Analyzing Airlines Data with PySpark - III

Advanced Spark: Accumulators, Spark Submit, MapReduce , Behind The Scenes

- Get information from individual processing nodes using accumulators
- Resource Material
- See it in Action: Using an Accumulator variable
- Long running programs using spark-submit
- See it in Action: Running a Python script with Spark-Submit
- Behind the scenes: What happens when a Spark script runs?
- Running MapReduce operations
- See it in Action: MapReduce with Spark