

PySpark with Python Fundamentals

A four-day progression from Python data handling to distributed analytics

FA-0656 · 4 days

Data & Analytics · Software Development · DevOps, Cloud & Infrastructure

Develop Python data-handling foundations, understand Spark clusters and build selected DataFrame, SQL, ML and streaming workflows in a prepared Linux lab.

Introduction

This four-day course is for learners who already understand general programming, command-line work and basic data management but are new to Python or PySpark. It begins with a focused Python refresher before moving to distributed computing and Spark.

A prepared Linux environment supports Python exercises, a small Spark configuration/submission lab and a data-processing pipeline. DataFrames, SQL, schemas, transformations and data quality receive practical attention. Machine learning and Structured Streaming extend that pipeline through selected examples.

The final day introduces application packaging, resource allocation, monitoring and ecosystem integration. It is not a complete production-platform deployment or mastery programme: YARN/Kubernetes, the wider ML algorithm catalogue and advanced operational work are comparisons or demonstrations where required by time and infrastructure.

Learning outcomes

The course teaches participants to:

- Use essential Python data structures, functions, files and selected numerical/analysis libraries.
- Distinguish parallel and distributed processing and explain driver, executor and cluster-manager responsibilities.
- Work with a prepared Linux Spark setup and submit a bounded application.

- Load and transform data with PySpark DataFrames and SQL, including schema and quality checks.
- Inspect plans and evaluate caching, partitioning and resource trade-offs.
- Build and evaluate a selected DataFrame-based ML pipeline.
- Run a prepared Structured Streaming example and explain state, event-time and delivery limitations.
- Identify packaging, access, monitoring, integration and deployment work needed for a production system.

Prerequisites

- Solid foundation in general programming concepts (variables, functions, control structures).
- Basic familiarity with command-line interfaces and terminal operations.
- Fundamental understanding of data structures and algorithms.
- Working knowledge of IT infrastructure and systems architecture principles.
- Basic understanding of data management concepts (databases, data formats).
- Experience with software development lifecycle or data analysis workflows.

A prepared, authorised Linux lab with compatible Python, Java, Spark and notebook packages, sample datasets and any required cluster/storage access. This is not an introduction to programming or Linux administration; deeper ML and cluster topics use prepared examples.

Training outline

1. Day 1 – Python Fundamentals for Data Processing

Python programming essentials

- Linux Python setup, virtual environments, package management and Jupyter notebooks.
- Variables, types, lists, dictionaries, sets, tuples, control structures and functions.
- Files and data I/O, strings, regular expressions, comprehensions and functional concepts.
- Classes, objects, inheritance and polymorphism, with a small data-oriented example.

Data-analysis libraries

- NumPy arrays, vectorised and mathematical operations.
- Pandas Series/DataFrames, loading, cleaning, transformation, aggregation and grouping.
- Selected Matplotlib/Seaborn visualisation examples and choosing an appropriate chart.

2. Day 2 — Distributed Computing and Spark Foundations

Distributed systems and Linux

- Parallel versus distributed processing, cluster components and common coordination challenges.
- Linux commands, filesystems/storage, processes and monitoring relevant to the lab.
- Cluster topologies, resource management and network communication.

Spark architecture and setup

- Spark history and ecosystem; driver/executor architecture.
- RDDs and DataFrames, with Scala/Java typed Datasets as context rather than a PySpark API.
- Inspect/configure a small standalone training cluster or prepared equivalent.
- YARN and Kubernetes integration overview; not installation of every cluster manager.
- Python/Spark API trade-offs, PySpark shell, SparkSession and notebooks.

3. Day 3 — PySpark DataFrames, SQL and Quality

DataFrames and transformations

- DataFrame/RDD comparison, schemas and inference.
- CSV, JSON and Parquet loading; DataFrames from Python collections or RDDs where supported.
- Selection, filtering, projection, grouping, joins and unions.
- Arrays, maps, structs and UDF examples, including built-in-function alternatives.

SQL, plans and data quality

- SparkSession, temporary views and tables; selected SQL and window-function queries.
- Caching/persistence, partitioning, bucketing and execution-plan interpretation.
- Schema validation and cleansing in a bounded data-processing pipeline.

4. Day 4 — Selected Applications and Deployment Considerations

DataFrame-based MLlib

- pyspark.ml versus legacy RDD-based pyspark.mllib.

- Feature extraction, transformation, scaling and normalisation.
- Estimator/transformer pipelines and evaluation with appropriate train/validation separation.
- One selected classification or regression lab; clustering and recommendation as guided examples.

Structured Streaming and operations

- Streaming concepts, batch versus incremental processing and legacy DStream context.
- Prepared Structured Streaming source/sink example with state and event-time windows.
- Packaging and submission, resource allocation, logging and monitoring.
- Database/warehouse and messaging integration concepts, subject to connector and access requirements.

Lab review and follow-on work

- Review the small cluster setup and data-processing pipeline.
- Evaluate the selected ML example and inspect a bounded streaming application.
- Review correctness, observed performance, integration constraints and production-readiness gaps.