Graph Analytics & Data Engineering Training Programme

Technical Stacks: Neo4j | Apache Spark | Apache Airflow | NeoDash | Bloom | Power BI

Duration: 15 Days

Course Overview

This training programme is designed for data engineers seeking to advance into production-grade graph analytics, large-scale data processing, automated pipelines, and professional graph and BI visualisation.

Why Graph Databases and Not Just Relational Databases?

Relational databases are excellent for structured, tabular data, but struggle when relationships between data points are as important as the data itself. Graph databases address this by:

- Treating relationship traversal as a first-class operation, making them fast regardless of data volume
- Replacing complex multi-JOIN queries with simple, readable Cypher pattern matches
- Allowing the schema to evolve without altering tables or migrating data
- Running graph algorithms such as PageRank, community detection, and shortest path directly on the data structure

This training operationalises that advantage by building end-to-end pipelines that move data from Oracle and PostgreSQL into Neo4j, enrich it with graph analytics, and surface insights through automated pipelines, dashboards, and AI-powered query interfaces.

Course Objectives

- Understand the core concepts of Neo4j graph databases, including advanced querying, graph algorithms, and vector-based ML integration.
- Process and transform large-scale datasets using Apache Spark and integrate Spark pipelines with Neo4j for scalable graph ingestion.
- Build production-grade Apache Airflow pipelines with scalable executors, monitoring, and automated data ingestion workflows from Oracle and PostgreSQL.
- Design and deploy interactive graph dashboards and visual network investigations using NeoDash and Neo4j Bloom for KPI reporting, data exploration, and query-free graph analysis.
- Connect Power BI to the Neo4j graph layer for enterprise-grade reporting and dynamic graph-powered visualisations.
- Integrate graph intelligence into end-to-end pipelines connecting relational databases to a graph analytics and AI layer.

Learning Outcomes

Upon completion of this training, participants will be able to:

- Design, query, and optimise advanced Neo4j graph models using variable-length paths, APOC procedures, and GDS algorithms for network analysis and community detection.
- Build and deploy a Graph RAG pipeline using Neo4j as a vector and knowledge graph store integrated with a Python LLM backend.
- Use Apache Spark to process and transform large datasets and integrate Spark pipelines with Neo4j using the Neo4j Spark Connector.
- Configure and operate Apache Airflow in production, designing modular DAGs that automate Oracle and Postgres-to-Neo4j pipelines with retry logic, SLA monitoring, and alerting.

- Build interactive dashboards and conduct visual graph investigations using NeoDash and Neo4j Bloom for KPI reporting, data exploration, and query-free network analysis.
- Connect Power BI to the Neo4j graph layer via FastAPI middleware for enterprise reporting with dynamic Cypher-bound filters.

Week 1: Neo4j – Advanced Analytics & Graph RAG

Day 1 – Graph Modelling & Advanced Cypher

- Graph modelling principles, node labels, relationship types, properties, and schema design patterns
- Variable-length paths, UNWIND, WITH chaining, and APOC procedures
- Query profiling and optimisation with EXPLAIN and PROFILE
- **Lab:** Design a graph schema for RAG document storage; write and optimise Cypher queries against the model

Day 2 – Graph Data Science Algorithms

- GDS plugin setup and graph projection
- Centrality algorithms: PageRank, Betweenness Centrality, and Degree Centrality
- Community detection with Louvain and Label Propagation; writing results back as node properties
- **Lab:** Run centrality and community detection on a projected graph; export enriched scores as node properties for use in the RAG retrieval layer

Day 3 – Vector Indexes & Hybrid Graph Search

- Neo4j Vector Index setup, embedding storage, and similarity search
- Hybrid queries combining vector search with Cypher graph traversal
- **Lab:** Generate and store embeddings on graph nodes; build a hybrid query that retrieves semantically and structurally relevant context

Day 4 – Graph RAG: Retrieval Pipeline

- RAG architecture: retriever, context assembler, and generator
- Subgraph extraction, relevance ranking using GDS scores, and context formatting for LLM prompts
- **Lab:** Build the retrieval component — accept a natural language question, run a hybrid query, rank results using PageRank scores, and format subgraph output as LLM-ready context

Day 5 – Graph RAG: LLM Integration & Deployment

- LLM connection via LangChain or LlamaIndex, prompt engineering, and RAG evaluation
- Packaging the full pipeline as a FastAPI endpoint
- **Lab:** Connect the retrieval layer to an LLM; design the system prompt; deploy as a FastAPI endpoint, and evaluate response quality against varied questions

Week 2: Apache Spark & Apache Airflow - Data Processing & Pipelines

Day 6 – Apache Spark Fundamentals & Neo4j Integration

- Spark architecture: Driver, Executors, DataFrames, and SparkSession
- DataFrame transformations, joins, aggregations, and reading and writing Parquet and CSV
- Neo4j Spark Connector; setup, configuration, reading from, and writing to Neo4j

- **Lab:** Set up a local Spark environment; transform a raw dataset into a graph-ready structure; load nodes and relationships into Neo4j using the Spark Connector

Day 7 – Spark with Oracle & PostgreSQL

- Reading from Oracle and PostgreSQL using JDBC in Spark
- Partitioned reads for large tables, write strategies, and incremental loading patterns
- Transforming relational data into graph-ready node and relationship structures
- **Lab:** Extract data from both Oracle and PostgreSQL using Spark JDBC; apply transformations; write the processed output as Parquet, and verify the schema

Day 8 – Airflow Architecture & Production Setup

- Airflow components: Scheduler, Workers, Metadata DB, and Executors
- CeleryExecutor vs LocalExecutor vs KubernetesExecutor
- Docker Compose setup with Redis broker
- **Lab:** Configure a CeleryExecutor cluster using Docker Compose; verify worker communication and task execution

Day 9 – Production DAG Design, Error Handling & Alerting

- TaskGroups, dynamic DAG generation, XCom, custom operators, and sensors
- Retry logic, timeout policies, SLA configuration, and email or Slack alerting on failure
- **Lab:** Build a dynamic DAG with a custom sensor; add retry logic, SLA breach handling, and failure email alerts

Day 10 – Oracle & Postgres-to-Neo4j Pipeline with Spark & Airflow

- Orchestrating Spark jobs from Airflow using SparkSubmitOperator
- Extracting from Oracle with cx_Oracle and PostgreSQL with psycopg2, transforming with Spark, loading into Neo4j
- Monitoring, logging with StatsD and Prometheus, and CI/CD deployment strategies
- **Lab:** Build a full end-to-end Airflow DAG, extract from Oracle and PostgreSQL, process with Spark, load into Neo4j; set up a Grafana dashboard for task monitoring

Week 3: Visualisation - NeoDash, Bloom & Power BI

Day 11 – NeoDash Fundamentals

- Widget types, Cypher-driven charts, parameterised filters, and report layout
- Global dashboard parameters and linking widgets across reports
- **Lab:** Build a graph overview dashboard with KPI cards, bar charts, and a graph visualisation widget driven by dynamic Cypher queries

Day 12 – Advanced NeoDash & FastAPI Integration

- Drill-down interactivity, map widgets, and linked parameters between reports
- Connecting NeoDash to the Graph RAG API built in Week 1
- Serving Neo4j results through a FastAPI middleware layer for external consumers
- **Lab:** Add drill-down interactivity to the Day 11 dashboard; wire the dashboard to the FastAPI endpoint and verify data flows from Neo4j through the API

Day 13 – Neo4j Bloom

- Perspectives, search phrases, scene setup, and styling rules
- Rule-based highlighting, path expansion, and sharing investigations
- **Lab:** Build a Bloom perspective on the loaded graph; define search phrases and style nodes by property values; investigate a graph scenario using path expansion and rule-based highlighting

Day 14 – Power BI & Neo4j Integration

- Connecting Power BI to Neo4j via the FastAPI middleware layer and Python scripting
- Dynamic filtering with Cypher parameter binding and building graph-powered Power BI reports
- Refreshing Power BI reports from live Neo4j graph data
- **Lab:** Connect Power BI to the FastAPI endpoint; build a report with dynamic filters that bind to Cypher parameters and refresh from live graph data

Day 15 – End-to-End Integration & System Review

- Full pipeline walkthrough, Oracle and PostgreSQL ingestion, Spark processing, Airflow orchestration, Neo4j graph analytics, Graph RAG API, NeoDash dashboard, Bloom investigation, and Power BI reporting
- Testing strategies, unit tests for DAGs, integration tests, and data quality checks across the stack
- **Lab:** Run a full end-to-end integration test across all components; verify data flows correctly from source databases through to all visualisation and AI layers; review and document the complete system architecture