1-Month Expert Architecture Training

Event-Driven Architecture, Polyglot Persistence, Big Data, AI, and Multitenacy

Prepared by: Sami Mbarki
Solution Architect, Java Expert 17 September 2025
ISYGO Consulting Services

ISYGO Consulting Services
Inspire Success, Your Goals & Opportunities
Delivering Advanced Architectural Training for Enterprise Solutions

Document ID: ede2b784-641c-41b2-8f8d-d9307e59fe85
Version: 1.0

1-Month Expert Training in Microservices and Polyglot Persistence

Sami Mbarki
Solution Architect, Java Expert
ISYGO Consulting Services

17 September 2025

Executive Summary

This intensive 1-month program equips experienced software professionals to design scalable, multitenant, event-driven microservice systems with polyglot persistence for document processing. Participants will build a capstone project—an Intelligent Document Processing Platform—that enables single or bulk document uploads, archiving in Apache Cassandra, AI-driven analysis, JSON transformation, and storage in PostgreSQL with tenant isolation. Using technologies like Apache Kafka, Spark, LLaMA 3, Cassandra, and PostgreSQL, the curriculum emphasizes production-ready architectures with robust testing, monitoring, and optimization. With 35 hours of weekly training (7 hours/day), structured resources, and mentorship, participants emerge as expert architects ready for enterprise challenges.

Contents

1 Program Philosophy & Objectives **3**

2 Target Audience & Prerequisites **3**
2.1 Target Audience . . . 3
2.2 Technical Prerequisites . . . 3
2.3 Recommended Resources . . . 4

3 Core Technologies & Tools **5**
3.1 Event-Driven Design Stack . . . 5
3.2 Big Data Processing Framework . . . 5
3.3 AI/ML Engineering Ecosystem . . . 5
3.4 Orchestration & Infrastructure . . . 5
3.5 Capstone Practice Components . . . 5

4 Detailed Training Plan **6**
4.1 Week 1: Foundational Mastery & Ingestion Layer . . . 6
4.1.1 Day 1: Kafka & Event-Driven Foundations . . . 6
4.1.2 Day 2: Microservices & Bulk Uploads . . . 7
4.1.3 Day 3: Spark Streaming & Delta Lake . . . 8
4.1.4 Day 4: MLOps & Model Lifecycle . . . 9
4.1.5 Day 5: Capstone Ingestion Pipeline . . . 9
4.2 Week 2: Architectural Integration . . . 10

4.2.1 Day 1: Polyglot Persistence & Feature Stores . . . 10
4.2.2 Day 2: Inference Architecture . . . 11
4.2.3 Day 3: Orchestration Patterns . . . 12
4.2.4 Day 4: Rules Engine & DDD . . . 13
4.2.5 Day 5: Capstone Processing Core . . . 13
4.3 Week 3: Optimization, Advanced AI, and Multitenancy . . . 14
4.3.1 Day 1: Performance Optimization . . . 14
4.3.2 Day 2: Advanced AI & LLM . . . 15
4.3.3 Day 3: Model Optimization & Multi-Modal . . . 15
4.3.4 Day 4: Security, Governance, and Multitenancy . . . 16
4.3.5 Day 5: Observability & Capstone Enhancement . . . 17
4.4 Week 4: Production Readiness & Finalization . . . 17
4.4.1 Day 1: Load Testing & Optimization . . . 17
4.4.2 Day 2: Security Hardening . . . 18
4.4.3 Day 3: Documentation & Runbooks . . . 19
4.4.4 Day 4: Performance Baselines . . . 19
4.4.5 Day 5: Final Presentation . . . 20

5 Success Metrics & Evaluation 21
5.1 Assessment Framework . . . 21
5.2 Evaluation Criteria (Aligned with Program Pillars) . . . 21

6 Instructional Methodology 21
6.1 Learning Approach . . . 21
6.2 Delivery Model . . . 22
6.3 Support for Diverse Learners . . . 22

7 Terminology 22

1 Program Philosophy & Objectives

This 1-month program transforms software professionals into expert architects capable of designing scalable, multitenant, event-driven microservice systems integrating Event-Driven Design (EDD), Big Data, AI, and polyglot persistence. The curriculum rests on three pillars:

1. **Technological Mastery:** Deep understanding of tools like Apache Kafka, Spark, LLaMA 3, Cassandra, PostgreSQL, and multitenant architectures.
2. **Architectural Integration:** Combining EDD, Big Data, AI, microservices, and polyglot persistence into cohesive, resilient systems.
3. **Production Readiness:** Prioritizing performance, security, tenant isolation, testing, and monitoring.

Capstone Project: Participants will design an **Intelligent Document Processing Platform**, a microservice-based system enabling tenants to upload documents (single or bulk), archive raw metadata in Cassandra, analyze content using LLaMA 3 for semantic extraction (e.g., invoice numbers), transform extracted data into JSON objects, and store them in PostgreSQL with multitenant isolation. The system uses Kafka for event-driven orchestration, with Cassandra for high-volume archival and PostgreSQL for structured storage, supported by comprehensive testing, monitoring, and optimization.

This philosophy guides the selection of target audience and prerequisites, ensuring participants are equipped to build the capstone system.

2 Target Audience & Prerequisites

2.1 Target Audience

- Senior Software Engineers transitioning to architecture roles
- Data Engineers taking on architectural responsibilities
- DevOps Engineers building scalable data/AI infrastructure
- Solutions Architects deepening expertise in multitenant systems

2.2 Technical Prerequisites

- Proficiency in Java, Scala, or Python
- Strong understanding of distributed systems (e.g., CAP theorem)
- Advanced SQL and NoSQL concepts (e.g., indexing, keyspaces)
- Familiarity with Linux/Unix, Git, and CI/CD pipelines
- Working knowledge of cloud platforms (AWS, Azure, or GCP)
- 3-5 years of software development experience

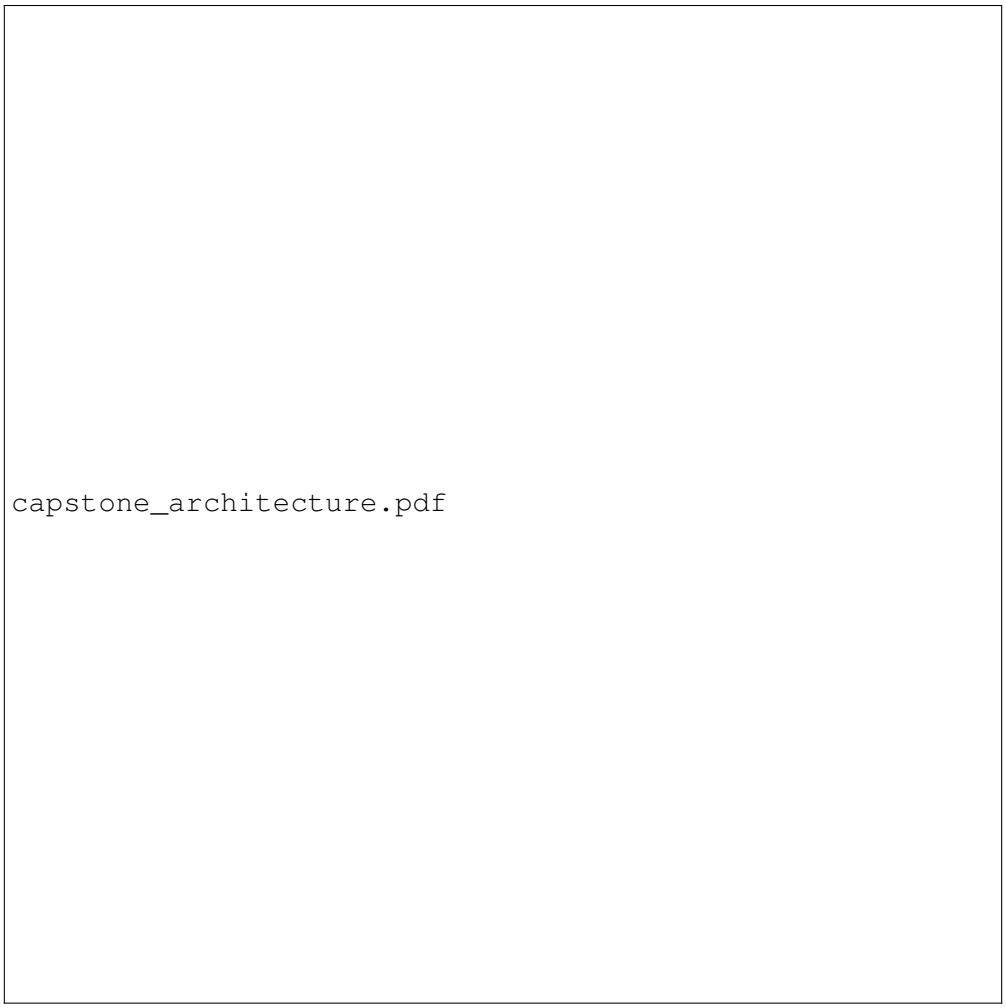

Figure 1: Event-Driven Microservice Architecture with Polyglot Persistence

2.3 Recommended Resources

- *Book: Designing Data-Intensive Applications* by Martin Kleppmann
- *Book: Building Microservices* by Sam Newman
- *Online:* Confluent Kafka Fundamentals (https://confluent.io/learn)
- *Online:* DataStax Academy Cassandra Fundamentals (https://www.datastax.com/learn)
- *Online:* Microsoft Azure Microservices Guide (https://learn.microsoft.com/en-us/azure/architecture/microservices)
- *Online:* Data Modeling in Apache Cassandra (DataStax, https://www.datastax.com/learn/cassandra-data-modeling)
- *Online:* PostgreSQL JSONB Guide (https://www.postgresql.org/docs/current/datatype-json.html)

These prerequisites prepare participants for the core technologies and hands-on labs detailed next, building toward the polyglot, microservice-based capstone project.

3 Core Technologies & Tools

3.1 Event-Driven Design Stack

- **Apache Kafka:** Core platform, Connect, Streams, Schema Registry
- **Debezium:** Real-time change data capture
- **Avro/Protobuf:** Schema-driven serialization

3.2 Big Data Processing Framework

- **Apache Spark:** Large-scale data processing with Structured Streaming
- **Delta Lake/Iceberg:** ACID-compliant table formats
- **Medallion Architecture:** Bronze, Silver, Gold layers

3.3 AI/ML Engineering Ecosystem

- **MLflow:** ML lifecycle management
- **Scikit-learn/TensorFlow/PyTorch:** Model development
- **Feast:** Feature store for multitenant features
- **Triton Inference Server:** Model serving

3.4 Orchestration & Infrastructure

- **Kubernetes:** Container orchestration
- **Docker:** Containerization
- **FastAPI:** RESTful APIs for document upload
- **Cloud Platforms:** AWS (S3, EMR, EKS), Azure (ADLS, Databricks, AKS), GCP (GCS, Dataproc, GKE)

3.5 Capstone Practice Components

- **Apache Camel:** Document routing
- **Drools:** Business rules for JSON validation
- **Ollama/LLaMA 3:** Semantic analysis
- **PostgreSQL:** Structured storage with JSONB
- **Apache Cassandra:** High-volume archival storage

- **Spring State Machine:** Document lifecycle management
- **Tesseract OCR:** Text extraction
- **Redis:** Distributed caching
- **Keycloak:** Multitenant authentication
- **Prometheus/Grafana:** System monitoring
- **JUnit/JMeter/K6:** Testing frameworks
- **OpenLineage:** Data lineage tracking

These technologies form the foundation for the training plan, enabling participants to build a production-ready, microservice-based, polyglot document processing system.

4 Detailed Training Plan

Structure: Each week includes 5 days of 7 hours (3h theory, 3.5h labs, 0.5h wrap-up), totaling 35 hours weekly. Labs build a multitenant, event-driven microservice system with polyglot persistence, supported by testing, monitoring, and optimization. Microservice architecture and polyglot persistence concepts are integrated into theory and labs to support the capstone project.

4.1 Week 1: Foundational Mastery & Ingestion Layer

Goal: Establish event-driven ingestion for document uploads (single/bulk) with PostgreSQL for initial storage, testing, and monitoring.

4.1.1 Day 1: Kafka & Event-Driven Foundations

Objective: Master Kafka fundamentals, security, and event-driven patterns for multitenant document uploads with initial testing and monitoring.

Theory (3h):

- Kafka architecture: Brokers, Topics, Partitions, ISR (1h)
- Kafka security and Schema Registry: SSL/TLS, ACLs, Avro with tenant metadata (1h)
- Testing and monitoring: Unit tests (JUnit), Prometheus metrics for Kafka (1h)

Resources:

- *Book: Kafka: The Definitive Guide* (Ch. 1-3, 7)
- *Online:* Confluent Kafka Security Tutorial (https://docs.confluent.io/platform/current/security)
- *Video:* Introduction to Microservice Testing (YouTube, 30-min pre-watch)

Lab (3.5h): Deploy a secure Kafka cluster for document uploads

- *Objective:* Set up a production-grade Kafka cluster with security, unit tests, and monitoring for multitenant uploads.

- *Steps:*
  1. Deploy Kafka using Docker on Minikube.
  2. Configure SSL/TLS and ACLs for tenant-specific topics (e.g., tenantA-documents).
  3. Set up Schema Registry with Avro schemas embedding tenant IDs.
  4. Develop a producer to publish single document upload events.
  5. Write JUnit tests for producer/consumer logic.
  6. Configure Prometheus for Kafka metrics (e.g., topic lag).
  7. Complete Knowledge Check: Quiz on ACLs, schema validation, and unit testing.
- *Technologies:* Kafka, Docker, Minikube, Avro, JUnit, Prometheus
- *Support:* Sample PDFs, JUnit templates, Prometheus config.

Wrap-Up (0.5h):

- Review test results and Kafka metrics.
- Q&A on multitenant Kafka design.
- *Capstone Progress:* Initialized document upload pipeline.

4.1.2 Day 2: Microservices & Bulk Uploads

Objective: Understand microservice architecture and implement bulk document uploads with integration tests and monitoring.

Theory (3h):

- Microservice architecture: Definition, pros (scalability, fault isolation), cons (complexity, latency), utility in document processing (0.5h)
- Kafka Connect for bulk uploads (1h)
- Spark architecture: Driver/Executors, Catalyst Optimizer (1h)
- Integration testing for microservice pipelines (0.5h)

Resources:

- *Book: Building Microservices* by Sam Newman (Ch. 1-2)
- *Online:* Microsoft Azure Microservices Guide (https://learn.microsoft.com/en-us/azure/architecture/microservices)

Lab (3.5h): Build microservice-based bulk upload pipeline

- *Objective:* Implement bulk document uploads using a FastAPI microservice with integration tests and monitoring.
- *Steps:*
  1. Deploy Kafka Connect with a file source connector for bulk PDFs.
  2. Configure tenant-specific topics for bulk uploads.

3. Develop a FastAPI microservice to handle bulk uploads to Kafka.
4. Set up a Spark cluster (Docker or EMR/Dataproc).
5. Write a Spark job to read bulk upload events.
6. Write integration tests for Kafka-to-Spark microservice flow.
7. Monitor Spark job and microservice metrics with Prometheus.
8. Complete Knowledge Check: Quiz on microservice benefits and trade-offs.

- *Technologies:* Kafka Connect, FastAPI, Spark, JUnit, Prometheus
- *Support:* Sample bulk PDF dataset, FastAPI template, test templates.

Wrap-Up (0.5h):

- Review integration tests and metrics.
- Q&A on microservice-based uploads.
- *Capstone Progress:* Enabled bulk document uploads via microservice.

4.1.3 Day 3: Spark Streaming & Delta Lake

Theory (3h):

- Structured Streaming, watermarking (1h)
- Medallion Architecture with Delta Lake (1h)
- Spark optimization: Partitioning, caching (1h)

Lab (3.5h): Archive documents in Delta Lake

- *Objective:* Stream document uploads to Delta Lake with optimization and monitoring.
- *Steps:*

1. Develop a Spark Streaming job for document events.
2. Implement watermarking for late uploads.
3. Archive to Delta Lake Bronze layer with tenant partitions.
4. Optimize Spark job with caching.
5. Test data integrity with JUnit tests.
6. Monitor job performance with Prometheus.
7. Complete Knowledge Check on partitioning and monitoring.

- *Technologies:* Spark Streaming, Delta Lake, JUnit, Prometheus
- *Support:* Sample dataset, optimization guide.

Wrap-Up (0.5h):

- Discuss optimization and monitoring results.
- *Capstone Progress:* Archived documents in Delta Lake.

4.1.4 Day 4: MLOps & Model Lifecycle

Theory (3h):

- ML lifecycle: CI/CD, MLflow (1h)
- Model deployment: Triton Server (1h)
- Testing ML pipelines (1h)

Lab (3.5h): Mock AI analysis for documents

- *Objective:* Deploy a model for document classification with testing.
- *Steps:*
  1. Install MLflow and configure tracking.
  2. Train a Scikit-learn model for document classification.
  3. Log multitenant artifacts in MLflow.
  4. Deploy model via Triton Server.
  5. Write unit tests for model inference.
  6. Monitor inference latency with Prometheus.
  7. Complete Knowledge Check on ML testing.
- *Technologies:* MLflow, Scikit-learn, Triton, JUnit, Prometheus
- *Support:* Pre-built model template.

Wrap-Up (0.5h):

- Review ML tests and metrics.
- *Capstone Progress:* Simulated AI analysis.

4.1.5 Day 5: Capstone Ingestion Pipeline

Theory (3h):

- Multitenant ingestion patterns with microservices (1h)
- Monitoring strategies: Grafana dashboards (1h)
- PostgreSQL for JSON storage (1h)

Lab (3.5h): Deploy microservice-based ingestion and PostgreSQL storage

- *Objective:* Integrate upload and storage with microservices, testing, and monitoring.
- *Steps:*
  1. Develop a FastAPI microservice for document uploads to Kafka.
  2. Integrate Kafka and Spark for processing upload events.
  3. Store upload metadata in PostgreSQL (JSONB) with tenant schemas.

4. Write integration tests for the microservice pipeline.
5. Set up Grafana dashboards for pipeline metrics.
6. Optimize Kafka partitions for throughput.
7. Complete Knowledge Check on microservice ingestion.

- *Technologies:* FastAPI, Kafka, Spark, PostgreSQL, Grafana
- *Support:* Sample PDFs, PostgreSQL schema template, FastAPI template.

Wrap-Up (0.5h):

- Review test results and dashboards.
- Q&A on microservice-based ingestion.
- *Capstone Progress:* Deployed microservice for document uploads and PostgreSQL storage.

Week in Review:

- *Deliverables:* Multitenant upload (single/bulk) pipeline with microservices, PostgreSQL storage, unit/integration tests, Prometheus/Grafana monitoring, optimized Kafka partitions.
- *Milestone:* Submit capstone progress reflection.

4.2 Week 2: Architectural Integration

Goal: Implement AI analysis, JSON transformation, and Cassandra archival with microservices, polyglot persistence, testing, and monitoring.

4.2.1 Day 1: Polyglot Persistence & Feature Stores

Objective: Understand polyglot persistence and implement a feature store for document metadata with testing and monitoring.

Theory (3h):

- Polyglot persistence: Definition, pros (performance, scalability), cons (consistency, complexity), utility in document processing (0.5h)
- Feature store architecture: Multitenant serving (1h)
- Real-time feature computation with microservices (1h)
- Testing feature pipelines (0.5h)

Resources:

- *Online:* Data Modeling in Apache Cassandra (https://www.datastax.com/learn/cassandra-data-mo
- *Online:* PostgreSQL JSONB Guide (https://www.postgresql.org/docs/current/datatype-json.html)

Lab (3.5h): Store document metadata in Feast with polyglot persistence

- *Objective:* Implement a feature store with PostgreSQL and introduce Cassandra concepts, including testing and monitoring.

- *Steps:*
  1. Deploy Feast with PostgreSQL backend for tenant-specific features.
  2. Define multitenant feature views for document metadata.
  3. Build Spark streaming job for feature computation.
  4. Set up Cassandra with tenant-specific keyspaces for future archival.
  5. Write integration tests for feature pipeline and polyglot setup.
  6. Monitor feature store and Cassandra with Prometheus.
  7. Complete Knowledge Check: Quiz on polyglot persistence (Cassandra vs. PostgreSQL use cases).
- *Technologies:* Feast, Spark, PostgreSQL, Cassandra, JUnit, Prometheus
- *Support:* Feature view templates, Cassandra keyspace template.

Wrap-Up (0.5h):

- Review tests and metrics.
- Q&A on polyglot persistence.
- *Capstone Progress:* Stored document metadata in Feast and initialized Cassandra.

4.2.2 Day 2: Inference Architecture

Objective: Implement an AI analysis microservice with best practices, testing, and monitoring.
Theory (3h):

- Microservice best practices: DDD, API gateways, event-driven communication (1h)
- Low-latency serving with caching (1h)
- Monitoring microservices with Grafana (1h)

Lab (3.5h): Build AI analysis microservice

- *Objective:* Develop a FastAPI microservice for LLaMA 3 analysis using microservice best practices, with testing and monitoring.
- *Steps:*
  1. Create FastAPI microservice for tenant-specific analysis with DDD.
  2. Deploy on Kubernetes with Istio for API Gateway.
  3. Integrate LLaMA 3 for semantic extraction (e.g., invoice numbers).
  4. Write unit tests (JUnit) for FastAPI endpoints.
  5. Monitor API latency with Grafana.
  6. Complete Knowledge Check: Quiz on microservice design patterns (e.g., API Gateway benefits).
- *Technologies:* FastAPI, Kubernetes, Istio, LLaMA 3, JUnit, Grafana

- *Support:* Pre-built FastAPI template, Istio configuration.

Wrap-Up (0.5h):

- Review test results and dashboards.
- Q&A on microservice best practices.
- *Capstone Progress:* Enabled AI-driven document analysis microservice.

4.2.3 Day 3: Orchestration Patterns

Theory (3h):

- Workflow orchestration with Spring State Machine (1h)
- Apache Camel for microservice integration (1h)
- Cassandra for high-volume archival with polyglot best practices (1h)

Lab (3.5h): Orchestrate analysis and Cassandra archival

- *Objective:* Orchestrate document analysis and archive metadata in Cassandra using microservices and polyglot persistence best practices.
- *Steps:*
  1. Define states (UPLOADED, ANALYZED, ARCHIVED) in Spring State Machine.
  2. Implement multitenant state transitions.
  3. Optimize Cassandra keyspaces for write-heavy workloads (e.g., tenant-specific partition keys).
  4. Create Camel routes in a microservice to archive metadata in Cassandra.
  5. Write integration tests for archival flow.
  6. Monitor Cassandra with Prometheus.
  7. Complete Knowledge Check on Cassandra archival and polyglot best practices.
- *Technologies:* Spring State Machine, Apache Camel, Cassandra, JUnit, Prometheus
- *Support:* Pre-configured Cassandra cluster, sample keyspaces, Camel routes.

Wrap-Up (0.5h):

- Discuss archival and test results.
- Q&A on polyglot persistence with Cassandra.
- *Capstone Progress:* Archived metadata in Cassandra via microservice.

4.2.4 Day 4: Rules Engine & DDD

Theory (3h):

- Drools for JSON validation (1h)
- Domain-Driven Design: Bounded contexts for microservices (1h)
- Optimizing Drools performance (1h)

Lab (3.5h): Validate JSON output

- *Objective:* Apply multitenant JSON validation with optimization in a microservice.
- *Steps:*
  1. Define Drools rules for JSON validation in a microservice.
  2. Integrate with Camel for routing.
  3. Optimize Drools rule execution.
  4. Write unit tests for validation logic.
  5. Monitor rule performance with Prometheus.
  6. Complete Knowledge Check on DDD and optimization.
- *Technologies:* Drools, Apache Camel, JUnit, Prometheus
- *Support:* Sample Drools rules, DDD templates.

Wrap-Up (0.5h):

- Review validation and optimization.
- *Capstone Progress:* Validated JSON objects in microservice.

4.2.5 Day 5: Capstone Processing Core

Theory (3h):

- Multitenant AI and JSON integration in microservices (1h)
- Polyglot persistence: Cassandra and PostgreSQL integration, synchronization best practices (1h)
- Monitoring with Grafana dashboards (1h)

Lab (3.5h): Integrate AI, JSON, and polyglot storage

- *Objective:* Store JSON in PostgreSQL and archive in Cassandra with microservices, testing, and monitoring, applying polyglot best practices.
- *Steps:*
  1. Integrate FastAPI microservice with LLaMA 3 for analysis.
  2. Transform data to JSON via Camel microservice.
  3. Store JSON in PostgreSQL with tenant schemas, using JSONB indexes.

4. Archive metadata in Cassandra tenant-specific keyspaces.
5. Use Kafka Connect for data synchronization between Cassandra and PostgreSQL.
6. Write integration tests for polyglot storage.
7. Set up Grafana dashboards for pipeline and database metrics.
8. Complete Knowledge Check on polyglot persistence integration (e.g., synchronization strategies).

- *Technologies:* FastAPI, LLaMA 3, Camel, PostgreSQL, Cassandra, Kafka Connect, Grafana
- *Support:* PostgreSQL/Cassandra templates, dashboard configs.

Wrap-Up (0.5h):

- Review test results and dashboards.
- Q&A on polyglot persistence integration.
- *Capstone Progress:* Integrated polyglot storage with microservices.

Week in Review:

- *Deliverables:* Multitenant AI analysis, JSON transformation, Cassandra archival, PostgreSQL storage, integration tests, Grafana dashboards, all using microservices and polyglot persistence.
- *Milestone:* Submit capstone progress reflection.

4.3 Week 3: Optimization, Advanced AI, and Multitenancy

Goal: Optimize the polyglot document processing system with advanced testing and monitoring.

4.3.1 Day 1: Performance Optimization

Theory (3h):

- Spark optimization: AQE, skew handling (1h)
- Kafka and Cassandra tuning for microservices (1h)
- End-to-end pipeline testing (1h)

Lab (3.5h): Optimize bulk processing

- *Objective:* Enhance throughput for bulk uploads with testing.
- *Steps:*

1. Tune Spark jobs with AQE for bulk processing.
2. Optimize Kafka topics and Cassandra write throughput.
3. Write end-to-end tests for upload pipeline.
4. Monitor performance with Prometheus/Grafana.
5. Complete Knowledge Check on optimization.

- *Technologies:* Spark, Kafka, Cassandra, JUnit, Grafana
- *Support:* Sample bulk dataset, optimization guide.

Wrap-Up (0.5h):

- Discuss optimization and test results.
- *Capstone Progress:* Optimized bulk upload performance.

4.3.2 Day 2: Advanced AI & LLM

Theory (3h):

- RAG architecture with vector embeddings (1h)
- Vector databases: PGVector (1h)
- Monitoring AI microservices (1h)

Lab (3.5h): Implement semantic search for documents

- *Objective:* Enable tenant-specific semantic search with monitoring.
- *Steps:*
  1. Set up PGVector for tenant-specific vectors in PostgreSQL.
  2. Generate embeddings with LLaMA 3 in a microservice.
  3. Implement semantic search with PGVector.
  4. Write tests for search accuracy.
  5. Monitor AI latency with Grafana.
  6. Complete Knowledge Check on AI monitoring.
- *Technologies:* PGVector, LLaMA 3, JUnit, Grafana
- *Support:* Sample embeddings.

Wrap-Up (0.5h):

- Review search and monitoring results.
- *Capstone Progress:* Added semantic search capabilities.

4.3.3 Day 3: Model Optimization & Multi-Modal

Theory (3h):

- Model optimization: Quantization, pruning (1h)
- Multi-modal AI for text/image in microservices (1h)
- Testing multi-modal pipelines (1h)

Lab (3.5h): Enhance multi-modal document analysis

- *Objective:* Optimize AI for text/image analysis with testing.
- *Steps:*
  1. Quantize LLaMA 3 model for efficiency.
  2. Integrate Tesseract OCR with LLaMA 3 in a microservice.
  3. Update JSON transformation for multi-modal data.
  4. Write tests for multi-modal output.
  5. Monitor AI performance with Prometheus.
  6. Complete Knowledge Check on model optimization.
- *Technologies:* LLaMA 3, Tesseract, JUnit, Prometheus
- *Support:* Sample image-based PDFs.

Wrap-Up (0.5h):

- Discuss optimization and test results.
- *Capstone Progress:* Enhanced AI for multi-modal documents.

4.3.4 Day 4: Security, Governance, and Multitenancy

Theory (3h):

- Data governance: GDPR, audit trails (1h)
- Security: mTLS, OAuth2, RBAC for microservices and polyglot databases (1h)
- Polyglot persistence: Securing Cassandra/PostgreSQL (1h)

Lab (3.5h): Secure polyglot storage

- *Objective:* Secure Cassandra and PostgreSQL with tenant isolation in microservices, applying polyglot best practices.
- *Steps:*
  1. Implement audit trails with OpenLineage for polyglot data flows.
  2. Deploy Keycloak for OAuth2 authentication in microservices.
  3. Configure RBAC for microservices and databases.
  4. Secure Cassandra keyspaces and PostgreSQL schemas with tenant isolation.
  5. Write security tests for tenant isolation.
  6. Monitor lineage with OpenLineage.
  7. Complete Knowledge Check on polyglot security and microservices.
- *Technologies:* OpenLineage, Keycloak, PostgreSQL, Cassandra
- *Support:* Pre-configured Keycloak, Cassandra keyspaces, PostgreSQL schemas.

Wrap-Up (0.5h):

- Review security and lineage.
- Q&A on securing polyglot persistence.
- *Capstone Progress:* Secured polyglot storage with microservices.

4.3.5 Day 5: Observability & Capstone Enhancement

Theory (3h):

- Advanced monitoring with Prometheus/Grafana for microservices and polyglot databases (1h)
- Multitenant AI enhancements (1h)
- Optimizing Cassandra/PostgreSQL (1h)

Lab (3.5h): Enhance observability

- *Objective:* Monitor polyglot document processing with optimization in microservices.
- *Steps:*
  1. Deploy Prometheus/Grafana for tenant-aware microservice and database monitoring.
  2. Implement OpenLineage for JSON data lineage across Cassandra/PostgreSQL.
  3. Enhance LLaMA 3 with tenant-specific prompts in a microservice.
  4. Optimize Cassandra queries and PostgreSQL JSONB indexes.
  5. Test storage performance with JUnit.
  6. Complete Knowledge Check on observability and polyglot persistence.
- *Technologies:* Prometheus, Grafana, OpenLineage, LLaMA 3, Cassandra, PostgreSQL
- *Support:* Sample dashboard templates, Cassandra/PostgreSQL optimization guides.

Wrap-Up (0.5h):

- Validate dashboards and optimization.
- *Capstone Progress:* Added observability to polyglot pipeline with microservices.

Week in Review:

- *Deliverables:* Optimized polyglot processing, secure storage, end-to-end tests, OpenLineage integration, tenant-specific dashboards, all using microservices and polyglot persistence.
- *Milestone:* Submit capstone progress reflection.

4.4 Week 4: Production Readiness & Finalization

Goal: Finalize the production-ready polyglot document processing system with microservices.

4.4.1 Day 1: Load Testing & Optimization

Theory (3h):

- Load/stress testing with JMeter/K6 for microservices (1h)
- Cloud cost management for polyglot systems (1h)
- Kubernetes auto-scaling for microservices (1h)

Lab (3.5h): Test bulk document processing

- *Objective:* Ensure scalability for bulk uploads with load testing, applying microservice best practices.
- *Steps:*
  1. Run load tests with JMeter/K6 for tenant-specific microservices.
  2. Analyze bottlenecks in upload/AI/storage microservices.
  3. Implement Kubernetes auto-scaling for microservices.
  4. Monitor load test metrics with Grafana.
  5. Complete Knowledge Check on load testing microservices and scalability use cases (e.g., tenant-specific processing).
- *Technologies:* JMeter, K6, Kubernetes, Grafana
- *Support:* Sample load test scripts.

Wrap-Up (0.5h):

- Discuss scalability and test results.
- *Capstone Progress:* Validated bulk processing scalability with microservices.

4.4.2 Day 2: Security Hardening

Theory (3h):

- Security hardening for polyglot microservices (1h)
- GDPR compliance checks for Cassandra/PostgreSQL (1h)
- Monitoring security metrics for microservices and databases (1h)

Lab (3.5h): Harden polyglot storage

- *Objective:* Secure Cassandra and PostgreSQL with monitoring in microservices, applying polyglot best practices.
- *Steps:*
  1. Encrypt JSON data in PostgreSQL and Cassandra.
  2. Harden Kubernetes with pod security policies for microservices.
  3. Conduct GDPR compliance checks for polyglot databases.
  4. Write security tests for tenant isolation in microservices and databases.
  5. Monitor security metrics with Prometheus.
  6. Complete Knowledge Check on security testing and polyglot persistence.
- *Technologies:* Kubernetes, Keycloak, PostgreSQL, Cassandra, Prometheus
- *Support:* Pre-configured security policies, encryption templates.

Wrap-Up (0.5h):

- Review security measures and metrics.
- *Capstone Progress:* Hardened polyglot system security with microservices.

4.4.3 Day 3: Documentation & Runbooks

Theory (3h):

- Architecture documentation for microservices and polyglot persistence (1h)
- Multitenant runbook creation (1h)
- Documenting polyglot test/monitoring setups (1h)

Lab (3.5h): Document the polyglot system

- *Objective:* Produce documentation for the microservice-based system with polyglot persistence.
- *Steps:*
  1. Create architecture diagrams with draw.io for microservices and Cassandra/PostgreSQL.
  2. Write runbooks for tenant-specific microservice and database operations.
  3. Document test cases and monitoring for Cassandra/PostgreSQL.
  4. Review with peers; validate with Knowledge Check.
- *Technologies:* draw.io, LaTeX, Markdown
- *Support:* Diagram templates, runbook samples.

Wrap-Up (0.5h):

- Discuss documentation quality.
- *Capstone Progress:* Documented polyglot microservice system.

4.4.4 Day 4: Performance Baselines

Theory (3h):

- Defining multitenant SLAs for microservices and polyglot databases (1h)
- Benchmarking polyglot microservices (1h)
- Optimizing Cassandra/PostgreSQL performance (1h)

Lab (3.5h): Establish polyglot system baselines

- *Objective:* Set performance baselines for tenant workloads in microservices and polyglot databases, applying optimization best practices.
- *Steps:*
  1. Define SLAs for upload/AI/storage microservices and Cassandra/PostgreSQL.
  2. Run benchmarks with JMeter/K6 for Cassandra queries and PostgreSQL JSONB.
  3. Optimize Cassandra compaction strategies and PostgreSQL GIN indexes.
  4. Monitor benchmarks with Grafana dashboards.
  5. Complete Knowledge Check on SLAs and polyglot optimization use cases (e.g., e-commerce, IoT).

- *Technologies:* JMeter, K6, Cassandra, PostgreSQL, Grafana
- *Support:* Sample benchmark scripts, optimization guides.

Wrap-Up (0.5h):

- Review baseline results.
- *Capstone Progress:* Established polyglot performance baselines with microservices.

4.4.5 Day 5: Final Presentation

Theory (3h):

- Presentation skills for architects (1h)
- Defending microservice and polyglot design decisions (1h)
- Presenting test and monitoring results (1h)

Lab (3.5h): Present the polyglot system

- *Objective:* Deliver a demo of the microservice-based document processing system with polyglot persistence.
- *Steps:*
  1. Prepare a live demo of upload, AI, Cassandra archival, and PostgreSQL storage microservices.
  2. Create slides highlighting microservice architecture, polyglot persistence, tests, and monitoring.
  3. Rehearse with Q&A.
  4. Present to peers/instructors.
  5. Complete Knowledge Check on presentation clarity.
- *Technologies:* PowerPoint, Kubernetes, FastAPI, Cassandra, PostgreSQL
- *Support:* Presentation template.

Wrap-Up (0.5h):

- Finalize deliverables.
- *Capstone Progress:* Presented complete polyglot microservice system.

Week in Review:

- *Deliverables:* Production-ready polyglot document processing system with microservices, load tests, tenant-specific dashboards, documentation, presentation.
- *Milestone:* Submit final capstone deliverables.

5 Success Metrics & Evaluation

5.1 Assessment Framework

- **Weekly Lab Completion (30%):** Quality of microservice and polyglot persistence implementations, tests, and monitoring, evaluated via mentor reviews and Knowledge Checks.
- **Capstone Project (50%):**
  - *Functionality (15%):* Reliable polyglot document processing with microservices.
  - *Architecture Quality (15%):* Scalability, maintainability, tenant isolation in microservices and polyglot databases.
  - *Code Quality (10%):* Clean code, test coverage.
  - *Performance (10%):* Meeting multitenant SLAs with optimization.
- **Final Presentation (20%):** Clarity in articulating microservice architecture, polyglot persistence, testing, and monitoring.

5.2 Evaluation Criteria (Aligned with Program Pillars)

- **Technological Mastery:** Depth in EDD, Big Data, AI, microservices, polyglot persistence, testing, and monitoring.
- **Architectural Integration:** Design of cohesive multitenant microservice systems with Cassandra and PostgreSQL.
- **Production Readiness:** Focus on performance, security, scalability, testing, and monitoring.
- **Problem-Solving:** Innovation in polyglot and microservice challenges.
- **Communication:** Effective collaboration and presentation of results.

6 Instructional Methodology

6.1 Learning Approach

- **Theory Sessions (43%):** Workshops on concepts, microservices, polyglot persistence, testing, and monitoring.
- **Hands-On Labs (50%):** Practical multitenant scenarios with microservices and polyglot storage.
- **Wrap-Ups (7%):** Daily reflection and Q&A.
- **Mentorship:** Weekly 1:1 sessions with architects.
- **Collaborative Learning:** Pair programming, group reviews.

6.2 Delivery Model

- Expert-led workshops by Sami Mbarki and team
- Guided labs with microservice architecture, polyglot persistence, testing, and monitoring reviews
- Peer learning via collaborative problem-solving
- Continuous feedback and iterative improvement
- Production-like environment with enterprise tools

6.3 Support for Diverse Learners

- **Pre-Course Refreshers:** Optional modules on Kafka, Spark, Cassandra, PostgreSQL, and microservices.
- **Advanced Tracks:** Deep dives into microservice optimization, polyglot persistence, and monitoring.
- **Knowledge Checks:** Daily quizzes on microservices, polyglot persistence, and testing.
- **Milestone Reviews:** Weekly capstone progress reflections.

7 Terminology

- **Multitenancy:** Designing systems to securely isolate and process data for multiple clients (tenants) within a shared infrastructure.
- **Event-Driven Design (EDD):** Architecture using events to trigger and communicate between decoupled services.
- **Microservices:** Small, independently deployable services that handle specific business functions, communicating via APIs or events.
- **Polyglot Persistence:** Using multiple databases (e.g., Cassandra, PostgreSQL) to optimize for different data needs.

gantt_chart.pdf

Figure 2: 4-Week Training Schedule