# **Kafka Migration Daily Implementation Tracker**

## **Week 1: Foundation & Learning**

### **DAY 1**

**Trello Task Card:** Setup Kafka Development Environment  
 **Task Description:** Install and configure Kafka cluster locally with basic testing  
 **Implementation Details:**

* [ ] Download Kafka 2.8.2 and extract to /opt/kafka
* [ ] Configure server.properties for development (heap size, retention)
* [ ] Start Zookeeper service on port 2181
* [ ] Start Kafka broker on port 9092
* [ ] Create test topic with 3 partitions
* [ ] Verify connectivity using console producer/consumer
* [ ] Document server configuration and startup commands

**Status Update:** ⏳ Not Started | 🔄 In Progress | ✅ Complete

### **DAY 2**

**Trello Task Card:** Master Kafka Core Concepts  
 **Task Description:** Deep dive into Kafka architecture and advanced configurations  
 **Implementation Details:**

* [ ] Study and document consumer groups behavior
* [ ] Test partition assignment with multiple consumers
* [ ] Configure and test different acknowledgment modes (acks=0,1,all)
* [ ] Experiment with offset management (earliest, latest, specific)
* [ ] Test consumer group rebalancing scenarios
* [ ] Create Kafka CLI commands cheat sheet
* [ ] Performance test with kafka-producer-perf-test.sh

**Status Update:** ⏳ Not Started | 🔄 In Progress | ✅ Complete

### **DAY 3**

**Trello Task Card:** Python Kafka Integration Setup  
 **Task Description:** Build basic Python producer and consumer classes  
 **Implementation Details:**

* [ ] Install kafka-python and confluent-kafka libraries
* [ ] Create BasicProducer class with JSON serialization
* [ ] Implement synchronous and asynchronous message sending
* [ ] Build BasicConsumer class with proper error handling
* [ ] Test producer-consumer communication with 1000 messages
* [ ] Implement proper connection management and cleanup
* [ ] Create logging configuration for debugging

**Status Update:** ⏳ Not Started | 🔄 In Progress | ✅ Complete

### **DAY 4**

**Trello Task Card:** Production-Ready Python Patterns  
 **Task Description:** Implement robust error handling and production patterns  
 **Implementation Details:**

* [ ] Build RobustProducer with retry logic and callbacks
* [ ] Implement circuit breaker pattern for connection failures
* [ ] Create consumer with manual offset commits
* [ ] Add dead letter queue handling for failed messages
* [ ] Implement graceful shutdown handlers
* [ ] Load test with 10,000+ messages per second
* [ ] Document production deployment considerations

**Status Update:** ⏳ Not Started | 🔄 In Progress | ✅ Complete

### **DAY 5**

**Trello Task Card:** Design Event Schemas and Topics  
 **Task Description:** Create Avro schemas and topic architecture for our system  
 **Implementation Details:**

* [ ] Design ProductViewEvent schema with all required fields
* [ ] Create RecommendationRequest and Response schemas
* [ ] Define UserProfileUpdate and MetricsEvent schemas
* [ ] Create topics with proper partitioning strategy:
  + product-view-events (6 partitions)
  + recommendation-requests (6 partitions)
  + recommendation-responses (6 partitions)
  + user-profile-updates (3 partitions)
  + metrics-events (2 partitions)
* [ ] Test schema validation with sample data
* [ ] Document topic naming conventions and retention policies

**Status Update:** ⏳ Not Started | 🔄 In Progress | ✅ Complete

### **DAY 6**

**Trello Task Card:** Stream Processing Foundation  
 **Task Description:** Learn stream processing concepts and implement basic transformations  
 **Implementation Details:**

* [ ] Study stateless vs stateful stream operations
* [ ] Learn windowing concepts (tumbling, hopping, session)
* [ ] Install Kafka Streams or Python stream processing library
* [ ] Create basic stream transformation pipeline
* [ ] Implement event enrichment processor
* [ ] Test stream joins between topics
* [ ] Build monitoring for stream processing lag

**Status Update:** ⏳ Not Started | 🔄 In Progress | ✅ Complete

### **DAY 7**

**Trello Task Card:** Advanced Stream Operations  
 **Task Description:** Implement windowing, aggregations, and state management  
 **Implementation Details:**

* [ ] Create windowed aggregation for user behavior
* [ ] Implement session-based user activity tracking
* [ ] Build stateful stream processor with local state store
* [ ] Test stream processor fault tolerance
* [ ] Implement exactly-once processing semantics
* [ ] Create stream processor health checks
* [ ] Document stream processing best practices

**Status Update:** ⏳ Not Started | 🔄 In Progress | ✅ Complete

## **Week 2: Implementation & Migration**

### **DAY 8**

**Trello Task Card:** System Architecture Design  
 **Task Description:** Map current components to Kafka microservices architecture  
 **Implementation Details:**

* [ ] Create detailed microservices architecture diagram
* [ ] Map StreamSimulator → EventProducerService
* [ ] Map AIEngine → RecommendationService
* [ ] Map MetricsTracker → MetricsStreamProcessor
* [ ] Define service boundaries and communication patterns
* [ ] Plan database integration points
* [ ] Create configuration management strategy
* [ ] Document deployment architecture

**Status Update:** ⏳ Not Started | 🔄 In Progress | ✅ Complete

### **DAY 9**

**Trello Task Card:** Migration Strategy Planning  
 **Task Description:** Create detailed step-by-step migration plan with risk mitigation  
 **Implementation Details:**

* [ ] Create migration timeline with rollback points
* [ ] Define feature flags for gradual rollout
* [ ] Plan data migration strategy (if needed)
* [ ] Create testing strategy for each migration phase
* [ ] Document rollback procedures
* [ ] Plan monitoring and alerting setup
* [ ] Create performance benchmark baselines
* [ ] Define success criteria for each phase

**Status Update:** ⏳ Not Started | 🔄 In Progress | ✅ Complete

### **DAY 10**

**Trello Task Card:** Event Producer Service Implementation  
 **Task Description:** Convert StreamSimulator to Kafka-based event producer  
 **Implementation Details:**

* [ ] Create ProductViewEventProducer class
* [ ] Implement user\_id-based partitioning for message ordering
* [ ] Add event validation and schema checking
* [ ] Create batch event sending capability
* [ ] Implement producer metrics collection
* [ ] Build REST API endpoints for event ingestion
* [ ] Test with 10,000 events/second load
* [ ] Add producer configuration management

**Status Update:** ⏳ Not Started | 🔄 In Progress | ✅ Complete

### **DAY 11**

**Trello Task Card:** Stream Processor Implementation  
 **Task Description:** Build event enrichment and routing stream processor  
 **Implementation Details:**

* [ ] Create EventEnrichmentProcessor class
* [ ] Implement user context enrichment from profile data
* [ ] Add event filtering and routing logic
* [ ] Create stream processor health monitoring
* [ ] Implement processor scaling logic
* [ ] Test stream processor with high-volume events
* [ ] Add processor configuration and tuning
* [ ] Document stream processing metrics

**Status Update:** ⏳ Not Started | 🔄 In Progress | ✅ Complete

### **DAY 12**

**Trello Task Card:** Recommendation Engine Service  
 **Task Description:** Extract AIEngine into standalone Kafka-consuming microservice  
 **Implementation Details:**

* [ ] Create RecommendationEngineService class
* [ ] Implement async recommendation request processing
* [ ] Add recommendation caching layer
* [ ] Create recommendation response producer
* [ ] Implement service health checks and metrics
* [ ] Add recommendation quality monitoring
* [ ] Test service with concurrent requests
* [ ] Optimize recommendation latency (<100ms)

**Status Update:** ⏳ Not Started | 🔄 In Progress | ✅ Complete

### **DAY 13**

**Trello Task Card:** Response Handler and Integration  
 **Task Description:** Build recommendation response handling and API integration  
 **Implementation Details:**

* [ ] Create RecommendationResponseHandler class
* [ ] Implement response caching and delivery
* [ ] Build WebSocket/SSE for real-time recommendations
* [ ] Create response timeout and retry handling
* [ ] Add recommendation A/B testing framework
* [ ] Implement response quality validation
* [ ] Test end-to-end recommendation flow
* [ ] Monitor recommendation delivery success rate

**Status Update:** ⏳ Not Started | 🔄 In Progress | ✅ Complete

### **DAY 14**

**Trello Task Card:** Metrics System and Final Testing  
 **Task Description:** Implement Kafka-based metrics and complete system validation  
 **Implementation Details:**

* [ ] Create KafkaMetricsTracker service
* [ ] Implement real-time precision/recall calculation
* [ ] Build metrics aggregation and dashboard
* [ ] Create comprehensive integration tests
* [ ] Perform end-to-end system testing
* [ ] Load test complete pipeline (50,000 events/second)
* [ ] Validate all success criteria:
  + Event processing latency < 100ms
  + 99.9% message delivery success
  + Zero data loss during testing
* [ ] Create deployment documentation

**Status Update:** ⏳ Not Started | 🔄 In Progress | ✅ Complete

## **Weekly Success Metrics**

### **Week 1 Targets:**

* [ ] Kafka cluster running and tested
* [ ] Python integration working with 1000+ msg/sec
* [ ] All schemas defined and topics created
* [ ] Basic stream processing implemented

### **Week 2 Targets:**

* [ ] All services implemented and integrated
* [ ] End-to-end pipeline tested
* [ ] Performance benchmarks met
* [ ] Migration plan validated