MongoDB to Power BI Real-time Data Pipeline

Complete Setup & Operation Guide

📋 Prerequisites

Required Software (Windows 11)

1. **Docker Desktop for Windows**
    - Download: https://www.docker.com/products/docker-desktop
    - Enable WSL 2 backend during installation
2. **Python 3.8 or higher**
    - Download: https://www.python.org/downloads/
    - ⚠️ Check "Add Python to PATH" during installation
3. **Microsoft ODBC Driver 18 for SQL Server**
    - Download: https://learn.microsoft.com/en-us/sql/connect/odbc/download-odbc-driver-for-sql-server
    - Choose "Download ODBC Driver 18 for SQL Server (x64)"
4. **Power BI Desktop**
    - Download: https://powerbi.microsoft.com/desktop/

Required Files

All files should be in the same folder:

- `requirements.txt`
- `docker-compose.yml`
- `Dockerfile.dockerfile`
- `complete-historical-migrator.py`
- `comprehensive-stream-processor.py`
- `mongodb-source-config-complete.json`
- `create_tables.py`

🚀 Initial Setup (One-Time)

Step 1: Install Python Dependencies

```powershell
# Create and activate virtual environment (recommended)
python -m venv .venv
.venv\Scripts\activate

# Install all dependencies
pip install -r requirements.txt
```

Step 2: Start Docker Infrastructure

```bash
# Start all services (Kafka, Zookeeper, SQL Server)
docker-compose up -d

# Verify all containers are running
docker ps
```

Expected output: You should see 5 containers running:

- zookeeper
- kafka
- kafka-connect
- kafka-ui
- sqlserver

Step 3: Wait for Kafka Connect to Initialize

Kafka Connect needs 2-3 minutes to start and install the MongoDB connector.

```powershell
# Check if Kafka Connect is ready (repeat until you get a response)
Invoke-RestMethod -Uri "http://localhost:8083/connectors"

# Expected: [] (empty array means it's ready)
```

Step 4: Create SQL Server Tables

```bash
# Create the database tables
python create_tables.py
```

Expected output: "Tables created successfully!"

Step 5: Load Historical Data

```bash
# Run one-time historical data migration
python complete-historical-migrator.py
```

This will:

- Load all customers and products
- Process all historical sales and payments
- Insert aggregated data into SQL Server

Expected duration: 5-30 minutes depending on data volume

Step 6: Register MongoDB Source Connector

```powershell
# For Windows PowerShell
Invoke-RestMethod -Method Post -Uri "http://localhost:8083/connectors" `
  -ContentType "application/json" `
  -InFile "mongodb-source-config-complete.json"
```

Verify it's registered:

```powershell
Invoke-RestMethod -Uri "http://localhost:8083/connectors"
# Expected: ["mongodb-source-complete"]
```

Step 7: Start Real-time Stream Processor

```bash
```

```
# Start the streaming processor (keep this running)
python comprehensive-stream-processor.py
```

Expected output:

```
============================================================
Starting Kafka Stream Processor
============================================================
Loading reference data from SQL Server...
Loaded 150 customers into cache
Loaded 500 products into cache
Started thread for topic: mongodb.ClearVueDB.customer
Started thread for topic: mongodb.ClearVueDB.product
Started thread for topic: mongodb.ClearVueDB.sales
Started thread for topic: mongodb.ClearVueDB.payment
Started thread for topic: mongodb.ClearVueDB.purchase
Started thread for topic: mongodb.ClearVueDB.age_analysis
Started thread for topic: mongodb.ClearVueDB.representative
```

🔄 Daily Operations

Starting the Environment

Terminal 1: Start Docker Services

```bash
docker-compose up -d
```

Terminal 2: Start Stream Processor

```bash
# Wait 2-3 minutes after starting Docker
python comprehensive-stream-processor.py
```

That's it! The pipeline is now running and processing real-time changes.

Stopping the Environment

Terminal 2: Stop Stream Processor

```
Press Ctrl+C
```

Terminal 1: Stop Docker Services

```bash
docker-compose down
```

🔍 Monitoring & Verification

Check Kafka Topics

Open browser: http://localhost:8080

You should see topics like:

- `mongodb.ClearVueDB.sales`
- `mongodb.ClearVueDB.customer`
- `mongodb.ClearVueDB.payment`
- etc.

Check SQL Server Data

```bash
# Connect to SQL Server
sqlcmd -S localhost,1433 -U sa -P akXHrP5xP02YfluQ -d PowerBIDashboards

# Check record counts
SELECT 'Sales', COUNT(*) FROM HistoricalSalesSummary
UNION ALL
SELECT 'Customers', COUNT(*) FROM HistoricalCustomerPerformance
UNION ALL
SELECT 'Products', COUNT(*) FROM HistoricalProductPerformance
UNION ALL
SELECT 'Payments', COUNT(*) FROM HistoricalPaymentSummary
UNION ALL
SELECT 'Purchases', COUNT(*) FROM HistoricalPurchaseSummary
UNION ALL
SELECT 'AgeAnalysis', COUNT(*) FROM HistoricalAgeAnalysis
GO
```

View Stream Processor Logs

The stream processor outputs logs showing:

- Records processed
- Buffer flushes to SQL Server
- Any errors encountered

💾 Power BI Connection

Connect to SQL Server

1. Open Power BI Desktop
2. Get Data → SQL Server
3. Enter connection details:
   - **Server:** `localhost,1433`
   - **Database:** `PowerBIDashboards`
   - **Authentication:** Database
   - **Username:** `sa`
   - **Password:** `akXHrP5xP02YfIuQ`

Recommended Tables & Views

Base Tables:

- `HistoricalSalesSummary`
- `HistoricalCustomerPerformance`
- `HistoricalProductPerformance`
- `HistoricalPaymentSummary`
- `HistoricalPurchaseSummary`
- `HistoricalAgeAnalysis`

Pre-built Views (Recommended):

- `vw_CustomerRiskAnalysis` - Customer health & payment risk
- `vw_ProductProfitability` - Product margins & performance
- `vw_SupplierPerformance` - Supplier spending analysis

- `vw_RegionalPerformance` - Regional sales breakdown
- `vw_CashFlowAnalysis` - Daily cash flow tracking

🧪 Testing the Pipeline

Test Real-time Updates

Option 1: Insert test data in MongoDB

```javascript
// Connect to MongoDB and insert a test sale
use ClearVueDB
db.sales.insertOne({
  "Customer_Number": "TEST001",
  "Trans_Date": "2025-09-29",
  "Doc_Num": "TEST-001",
  "Fin_Period": "202509",
  "Rep_Code": "REP01",
  "Trans_Type_Code": "SALE",
  "Sales_Line": [{
    "Inventory_Code": "PROD001",
    "Quantity": 5,
    "Unit_Sell_Price": {"$numberDecimal": "100.00"},
    "Total_Line_Price": {"$numberDecimal": "500.00"},
    "Last_Cost": {"$numberDecimal": "60.00"}
  }]
})
```

Option 2: Watch the stream processor logs You should see:

```
Processed sale: Customer TEST001, Amount: 500.0
Flushing buffers to SQL Server...
```

Option 3: Refresh Power BI Click "Refresh" in Power BI and verify the new data appears.

📦 Sharing with Your Team

Package for Distribution

Create a project folder with:

```
mongodb-powerbi-pipeline/
├── docker-compose.yml
├── Dockerfile.dockerfile
├── complete-historical-migrator.py
├── comprehensive-stream-processor.py
├── mongodb-source-config-complete.json
├── create_tables.py
├── README.md (this guide)
└── requirements.txt
```

Create requirements.txt:

```
pymongo==4.6.0
pyodbc==5.0.1
kafka-python==2.0.2
```

Team Setup Instructions

Share this guide with your team. Each team member needs to:

1. Install Docker Desktop
2. Install Python 3.8+
3. Install ODBC Driver 18
4. Clone/copy the project folder
5. Run the setup steps above

Important: Only ONE person should run the historical migrator. Everyone else can just start the stream processor to monitor real-time changes.

🐛 Troubleshooting

"Cannot connect to SQL Server"

```bash
```

```
# Check if SQL Server container is running
docker ps | grep sqlserver

# Check SQL Server logs
docker logs sqlserver
```

"Kafka Connect not responding"

```bash
# Restart Kafka Connect
docker-compose restart kafka-connect

# Wait 2-3 minutes and check again
curl http://localhost:8083/connectors
```

"No data in Power BI"

1. Verify historical migration completed successfully
2. Check SQL Server has data (see monitoring section)
3. Verify stream processor is running
4. Refresh Power BI connection

Stream processor shows errors

Check the logs for specific error messages. Common issues:

- MongoDB connection failed (check credentials)
- SQL Server connection failed (check Docker is running)
- Kafka topics not found (check Kafka Connect is registered)

🔒 Security Notes

⚠️ **IMPORTANT:** The current setup uses hardcoded credentials which is fine for development/testing but NOT for production.

For production deployment:

- Use environment variables for credentials
- Change default passwords

- Implement proper network security
- Use SSL/TLS for connections
- Restrict SQL Server access

📊 Architecture Overview

```
MongoDB Atlas (Cloud)
      ↓
Kafka Connect (Change Data Capture)
      ↓
Kafka Topics (Streaming)
      ↓
Stream Processor (Python)
      ↓
SQL Server (Local)
      ↓
Power BI Dashboards
```

Data Flow:

1. Changes happen in MongoDB
2. Kafka Connect captures changes
3. Stream processor aggregates and transforms
4. SQL Server stores analytics-ready data
5. Power BI visualizes in real-time

📞 Support

If team members encounter issues:

1. Check this guide first
2. Verify all prerequisites are installed
3. Check Docker containers are running
4. Review stream processor logs
5. Test SQL Server connectivity

Logs locations:

- Stream processor: Terminal output
- Docker services: `docker logs <container-name>`
- SQL Server: `docker logs sqlserver`