RTD GTFS-RT Data Pipeline

Real-time Transit Data Processing with Apache Flink

Press Space for next page →

What is RTD GTFS-RT Pipeline?

A Java application built with Apache Flink that processes real-time transit feeds from RTD Denver

- **Real-time Processing** - Downloads and processes GTFS-RT feeds every hour
- **Apache Flink** - Distributed stream processing framework
- **Three Feed Types** - Vehicle Positions, Trip Updates, and Alerts
- **Scalable Architecture** - Can run locally or on Flink clusters

Key Technologies

- Java 24
- Apache Flink 1.18.0
- Maven 3.6+
- Protocol Buffers (GTFS-RT)

Architecture Overview

High-Level Components

- **RTDGTFSPipeline** - Main orchestration class
- **GTFSRealtimeSource** - Custom source for hourly downloads
- **Data Models** - VehiclePosition, TripUpdate, Alert
- **Table API Sinks** - Configurable output destinations

Data Flow

1. **Source** - HTTP requests to RTD endpoints
2. **Parse** - Protobuf deserialization
3. **Transform** - Convert to structured models
4. **Sink** - Output via Table API

Feed URLs

The pipeline processes three GTFS-RT feed types from RTD Denver:

Vehicle Positions

```
https://www.rtd-denver.com/google_sync/VehiclePosition.pb
```

Real-time vehicle location data

Trip Updates

```
https://www.rtd-denver.com/google_sync/TripUpdate.pb
```

Schedule deviations and predictions

Alerts

```
https://www.rtd-denver.com/google_sync/Alert.pb
```

Service disruptions and notifications

Prerequisites

Before running the application, ensure you have:

- **Java 24** installed
- **Maven 3.6+** for build management
- **Apache Flink 1.18.0** (for cluster deployment)

Verify Installation

```
# Check Java version
java -version

# Check Maven version
mvn -version

# Check Flink (if using cluster)
flink --version
```

Build Commands

Essential Maven commands for building the project:

Clean and Compile

```
mvn clean compile
```

Package Application

```
# Full package with tests
mvn clean package

# Skip tests for faster build
mvn clean package -DskipTests
```

Development Build

```
# Just compile for development
mvn compile
```

Running Tests

Multiple ways to execute the test suite:

Run All Tests

```
mvn test
```

Run Specific Test Class

```
mvn test -Dtest=YourTestClassName
```

Run Tests with Coverage

```
mvn clean test jacoco:report
```

Skip Tests During Build

```
mvn clean package -DskipTests
```

💡 **Tip:** Always run tests after making code changes to ensure functionality remains intact

Running the Application

Two deployment modes available

Local Development Mode

Run with Flink Mini Cluster for development and testing:

Using Maven (Recommended)

```
mvn exec:java -Dexec.mainClass="com.rtd.pipeline.RTDGTFSPipeline"
```

Includes all dependencies automatically

Using Packaged JAR

```
# First package the application
mvn clean package

# Then run the JAR
java -cp target/rtd-gtfs-pipeline-1.0-SNAPSHOT.jar \
  com.rtd.pipeline.RTDGTFSPipeline
```

✅ **Local Mode:** Perfect for development, debugging, and testing

Flink Cluster Deployment

Deploy to production Flink cluster:

Submit Job to Cluster

```
# Package first
mvn clean package

# Submit to running Flink cluster
flink run target/rtd-gtfs-pipeline-1.0-SNAPSHOT.jar
```

Cluster Management

```
# Check job status
flink list

# Cancel a job
flink cancel <job-id>

# Stop a job with savepoint
flink stop <job-id>
```

⚠️ **Production:** Requires a running Flink cluster (standalone or YARN/K8s)

Docker Setup

Containerizing the RTD GTFS-RT Pipeline

Creating Dockerfile

First, let's create a Dockerfile for the application:

```
# Multi-stage build for smaller final image
FROM maven:3.9-openjdk-24-slim AS builder
WORKDIR /app
COPY pom.xml .
COPY src ./src
RUN mvn clean package -DskipTests

FROM openjdk:24-slim
WORKDIR /app
COPY --from=builder /app/target/rtd-gtfs-pipeline-1.0-SNAPSHOT.jar .
EXPOSE 8081
CMD ["java", "-cp", "rtd-gtfs-pipeline-1.0-SNAPSHOT.jar", \
     "com.rtd.pipeline.RTDGTFSPipeline"]
```

Docker Commands

Building and running the containerized application:

Build Docker Image

```
docker build -t rtd-gtfs-pipeline .
```

Run Container (Local Mode)

```
docker run --rm -p 8081:8081 rtd-gtfs-pipeline
```

Run with Environment Variables

```
docker run --rm -p 8081:8081 \
  -e FLINK_JOB_PARALLELISM=2 \
  rtd-gtfs-pipeline
```

Run in Background

```
docker run -d --name rtd-pipeline \
  -p 8081:8081 rtd-gtfs-pipeline
```

Docker Compose Setup

For more complex deployments with external dependencies:

```yaml
version: '3.8'
services:
  rtd-pipeline:
    build: .
    ports:
      - "8081:8081"
    environment:
      - FLINK_JOB_PARALLELISM=2
    depends_on:
      - kafka

  kafka:
    image: confluentinc/cp-kafka:latest
    ports:
      - "9092:9092"
    environment:
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://localhost:9092
```

```bash
# Start all services
docker-compose up -d

# View logs
```

Docker Testing

Running tests within Docker containers:

Test-Only Container

```
# Build test image
docker build --target builder -t rtd-pipeline-test .

# Run tests
docker run --rm rtd-pipeline-test mvn test
```

Integration Testing

```
# Run with test profile
docker run --rm -e SPRING_PROFILES_ACTIVE=test \
  rtd-pipeline-test mvn test
```

Volume Mount for Development

```
# Mount source code for live development
docker run --rm -v $(pwd):/app -w /app \
  maven:3.9-openjdk-24-slim mvn test
```

Monitoring & Debugging

Flink Web UI (Local)

- Access at `http://localhost:8081`
- Monitor job metrics
- View task managers

Docker Logs

```
# View container logs
docker logs rtd-pipeline

# Follow logs in real-time
docker logs -f rtd-pipeline
```

Health Checks

```
# Check container status
docker ps

# Execute commands in container
docker exec -it rtd-pipeline bash

# Check Java processes
docker exec rtd-pipeline jps -l
```

Troubleshooting

- Check memory allocation
- Verify network connectivity
- Monitor resource usage

Development Workflow

Local Development

```
mvn clean compile → mvn test → mvn exec:java
```

Docker Development

```
docker build → docker run → docker logs
```

Production Deployment

```
mvn clean package → flink run target/rtd-gtfs-pipeline-1.0-SNAPSHOT.jar
```

Questions & Next Steps

Key Features Covered

- ✅ Project architecture and components
- ✅ Local development with Maven
- ✅ Testing strategies and commands
- ✅ Docker containerization
- ✅ Flink cluster deployment

What's Next?

- Configure output sinks (Kafka, databases)
- Set up monitoring and alerting
- Implement CI/CD pipelines
- Scale for production workloads

Thank you! 🚀