YouTube ELT Pipeline - Complete Component Functions Guide

Table of Contents

Project Overview

The YouTube ELT Pipeline is a production-ready data engineering solution that automatically extracts, loads, and transforms YouTube video data using Apache Airflow, PostgreSQL, and Docker. The system provides automated data quality checks and a scalable architecture for analytics workloads.

Technology Stack:

- **Orchestration:** Apache Airflow
- **Database:** PostgreSQL
- **Containerization:** Docker & Docker Compose
- **Data Quality:** Soda Core
- **API Integration:** YouTube Data API v3
- **Language:** Python

Core Infrastructure Components

1. Apache Airflow (Orchestrator)

Function: Workflow orchestration and scheduling
Purpose: Manages, schedules, and monitors all pipeline tasks

Components:

- **Webserver:** User interface accessible at http://localhost:8080
- **Scheduler:** Task execution and dependency management
- **DAG Processor:** Loads and validates workflow definitions
- **Executor:** Manages task execution across workers

Key Features:

- Web-based UI for monitoring and management
- Dependency resolution between tasks

- Retry logic and error handling
- Scheduling with cron expressions
- Task logging and history tracking

2. PostgreSQL Database (Data Warehouse)

Function: Data storage and transformation layer
Purpose: Stores raw and processed YouTube data

Database Schemas:

- **staging:** Raw extracted data from YouTube API
- **core:** Cleaned, transformed data ready for analysis
- **public:** Airflow metadata and system tables

Key Tables:

```
-- Staging layer
staging.videos (raw YouTube data)

-- Core layer
core.videos (processed analytics data)

-- System tables
public.dag_run, public.task_instance (Airflow metadata)
```

3. Docker Containers (Environment)

Function: Containerized deployment and isolation
Purpose: Ensures consistent, reproducible environment

Docker Services:

- **airflow-webserver:** Web UI access (Port 8080)
- **airflow-scheduler:** Task scheduling engine
- **postgres:** Database service (Port 5434)
- **airflow-init:** Database initialization service

Configuration Files:

- `docker-compose-production.yml`: Production orchestration
- `Dockerfile`: Custom Airflow image
- `.env.production`: Environment variables

Data Pipeline Components (DAGs)

4. `produce_json` DAG

Function: YouTube data extraction
Purpose: Fetches video data from YouTube API

Process Flow:

1. Connects to YouTube API v3 using API key
2. Retrieves video IDs from specified playlists/channels
3. Extracts detailed video metadata:
    - Title, description, duration
    - View count, like count, comment count
    - Publication date, tags, category
4. Saves timestamped JSON files to `data/json/`
5. Handles API pagination and quota limits

Key Features:

- Robust error handling and retry logic
- API quota management
- Incremental data extraction
- JSON data validation

5. `update_db` DAG

Function: Data loading and transformation
Purpose: Moves data from JSON files to database

Process Steps:

1. **Table Creation:** Ensures staging.videos and core.videos tables exist
2. **Staging Load:** Reads JSON files and loads raw data to staging.videos
3. **Data Transformation:** Applies business logic and data cleaning
4. **Core Load:** Transfers processed data to core.videos table
5. **Duplicate Handling:** Manages upserts and data consistency

Transformation Logic:

```
-- Example transformations
- Date parsing and standardization
- Numeric data type conversions
- Text cleaning and normalization
- Duplicate detection and removal
```

6. `data_quality` DAG

Function: Data quality validation
Purpose: Ensures data integrity and reliability

Quality Checks:

- **Row Count Validation:** Ensures expected data volumes
- **Missing Value Detection:** Identifies incomplete records
- **Duplicate Record Identification:** Prevents data duplication
- **Data Type Consistency:** Validates field formats
- **Business Rule Validation:** Custom domain-specific checks

Soda Core Integration:

- Configurable quality thresholds
- Automated alerting on failures
- Historical quality trend tracking
- Custom quality metrics

7. `youtube_elt` DAG

Function: Master pipeline orchestration
Purpose: Coordinates the entire ELT process

Workflow Dependencies:

```
produce_json → update_db → data_quality
```

Scheduling Options:

- Manual trigger for ad-hoc runs
- Scheduled execution (daily/weekly)
- Event-driven triggers
- Sensor-based activation

Supporting Components

8. `include/scripts/youtube_elt.py`

Function: Core extraction logic
Purpose: Contains YouTube API interaction code

Key Features:

- **API Authentication:** Secure API key management
- **Video Data Retrieval:** Comprehensive metadata extraction
- **Error Handling:** Robust exception management and retries
- **JSON File Generation:** Structured data output
- **Logging:** Detailed execution tracking

Data Extraction Capabilities:

```
# Video metadata extracted:
- video_id, title, description
- duration, view_count, like_count
- comment_count, published_at
- channel_id, channel_title
- tags, category_id
```

9. `include/soda/` (Data Quality Framework)

Function: Data quality configuration
Purpose: Defines quality rules and checks

Configuration Files:

- **`configuration.yml`**: Database connection settings
- **`checks/videos.yml`**: Quality validation rules

Sample Quality Checks:

```
checks for core.videos:
  - row_count > 0
  - missing_count(title) = 0
  - duplicate_count(video_id) = 0
  - avg(view_count) > 1000
```

10. Environment Configuration

Function: System configuration management
Purpose: Stores sensitive data and settings

Configuration Files:

- **`.env.production`**: Production environment variables
- **`docker-compose-production.yml`**: Container orchestration
- **`init-db.sql`**: Database initialization scripts
- **`requirements.txt`**: Python dependencies

Environment Variables:

```
YOUTUBE_API_KEY=your_api_key_here
POSTGRES_USER=airflow
POSTGRES_PASSWORD=airflow
POSTGRES_DB=airflow
AIRFLOW__CORE__EXECUTOR=LocalExecutor
```

Data Flow Architecture

High-Level Data Flow

```
┌─────────────────┐     ┌──────────────────┐     ┌─────────────────┐
│   YouTube API   │───▶│   produce_json   │───▶│   data/json/    │
│  (Data Source)  │     │  (Extraction)    │     │  (Raw Files)    │
└─────────────────┘     └──────────────────┘     └─────────────────┘
                                                          │
                                                          ▼
┌─────────────────┐     ┌──────────────────┐     ┌─────────────────┐
│  core.videos    │◀───│    update_db     │◀───│ staging.videos  │
│  (Analytics)    │     │  (Transform)     │     │  (Raw Data)     │
└─────────────────┘     └──────────────────┘     └─────────────────┘
         │                        │
         ▼                        ▼
┌─────────────────┐     ┌──────────────────┐
│  data_quality   │     │     Airflow      │
│  (Validation)   │     │   (Metadata)     │
└─────────────────┘     └──────────────────┘
```

Detailed Process Flow

1. **Data Extraction:** YouTube API → JSON files
2. **Staging Load:** JSON files → staging.videos table
3. **Data Transformation:** staging.videos → core.videos (with business logic)
4. **Quality Validation:** Soda Core checks on core.videos
5. **Monitoring:** Airflow UI provides execution visibility

Deployment Information

Production Deployment

Server URL: http://localhost:8080
Login Credentials:

- Username: `admin`
- Password: `admin`

Container Status

```
# Check running containers
docker ps

# Expected services:
- youtube-elt-pipeline-airflow-webserver-1
- youtube-elt-pipeline-airflow-scheduler-1
- youtube-elt-pipeline-postgres-1
- youtube-elt-pipeline-airflow-init-1
```

Database Access

```
# PostgreSQL connection details
Host: localhost
Port: 5434
Database: airflow
Username: airflow
Password: airflow
```

DAG Status

All 4 DAGs should be loaded and available:

- ✅ `youtube_elt` - Master orchestration pipeline
- ✅ `produce_json` - YouTube data extraction
- ✅ `update_db` - Database loading and transformation
- ✅ `data_quality` - Data quality validation

Quick Reference

Starting the System

```
# Start all services
docker-compose -f docker-compose-production.yml up -d

# Check service health
docker-compose -f docker-compose-production.yml ps
```

Stopping the System

```
# Stop all services
docker-compose -f docker-compose-production.yml down

# Stop and remove volumes (complete cleanup)
docker-compose -f docker-compose-production.yml down -v
```

Monitoring Commands

```
# View logs
docker-compose -f docker-compose-production.yml logs airflow-webserver
docker-compose -f docker-compose-production.yml logs airflow-scheduler

# Access database
docker exec -it youtube-elt-pipeline-postgres-1 psql -U airflow -d airflow
```

Common Operations

```bash
# Restart specific service
docker-compose -f docker-compose-production.yml restart airflow-scheduler

# View container resource usage
docker stats

# Access Airflow container
docker exec -it youtube-elt-pipeline-airflow-webserver-1 bash
```

Architecture Benefits

Scalability

- Modular DAG design allows independent scaling
- Docker containers enable horizontal scaling
- PostgreSQL supports large datasets

Reliability

- Comprehensive error handling and retries
- Data quality validation prevents bad data
- Container isolation provides stability

Maintainability

- Clean separation of concerns
- Well-documented codebase
- Version-controlled configuration

Monitoring

- Airflow UI provides real-time visibility
- Detailed logging at all levels
- Data quality metrics tracking

This guide provides a comprehensive overview of all components in the YouTube ELT Pipeline. Each component is designed to work together in a cohesive, production-ready data engineering solution.

Last Updated: December 2024
Version: Production v1.0