Technical overview

DATAGRAM is a framework for data processing applications development. It supports both batch and streaming data processing mode.

The core of the framework is Metadata Server. Metadata Server provide store and management tools for information about data sources and targets, execution environment, data mappings, transformation sequences, etc.

DATAGRAM supports full cycle of data processing applications development:

- visual design of data flows, data mappings and data transformations;
- visual design of control flows and transformations sequences;
- source code generation for **Scala** with **Apache Spark** library;
- compiling and building application;
- deploying application to runtime environment;
- scheduling application execution;
- monitoring application execution;
- tools for stopping and restarting applications.

Runtime Environment

Runtime environment is based on **Apache Spark/Yarn**.

DATAGRAM can execute applications with **Apache Oozie** or **Apache Livy** servers. Apache Oozie is used in production. Apache Livy is used in development/debugging environment.

Transformation Designer

Visual designer for dataflows and data mappings.

Wide range of sources/targets are supported:

- RDBMS sources/targets via **JDBC** (including stored procedures)
- hierarchical sources/targets: **xml, avro** and **json**
- HDFS specific file formats: **ORC**, **PARQUET**
- **CSV, Apache Hive, Apache Kafka** source/targets

Data transformations types:

- Extended set of relational algebra operations: join, sort, aggregation, union, selection, projections, pivot, explode arrays, sequence generation
- Spark specific: **Spark SQL** – perform arbitrary SQL query over data streams
- Machine learning using **Spark MLLib** (decision trees, SVM, logistic regression etc.)
- JBoss Rules (**Drools**) - Business Rule Management System

Key features:

- Simple fields **data types**: STRING, DECIMAL, INTEGER, DATE, TIME, DATETIME, BINARY, BOOLEAN, LONG, FLOAT, DOUBLE;
- Support for **struct/array** data types for deep nested data structures;
- **View content** of arbitrary source/target ;

- Data fields **lineage** tracking;
- Partial execution of the transformation with viewing of **intermediate results;**
- **Generated code** can be viewed, edited and executed immediately;
- **Validation** of transformations based on database of the most frequent errors;
- Support for Spark Catalyst Optimizer.

Workflow Designer

Visual designer for transformations sequences:

- **Conditional**, **parallel** or **sequential** transformation execution;
- Generic steps: **shell** scripts or **java** programs;
- Sub workflows of arbitrary nesting level;
- Scheduled workflow execution by time or file system events.

Security

- **Active Directory (LDAP)** authentication
- Role-based authorization. Possible roles: developer, operator, viewer
- Encrypted passwords for access to external systems
- Support for **Kerberos** authentication of runtime environment

Versioning and teamwork

- Optimistic locking for concurrent updates
- **Apache Subversion** integration
- Hierarchical projects
- Project based, or object based check-in/check-out
- Protected (user-defined) code persists while updating the metadata version

Environment support

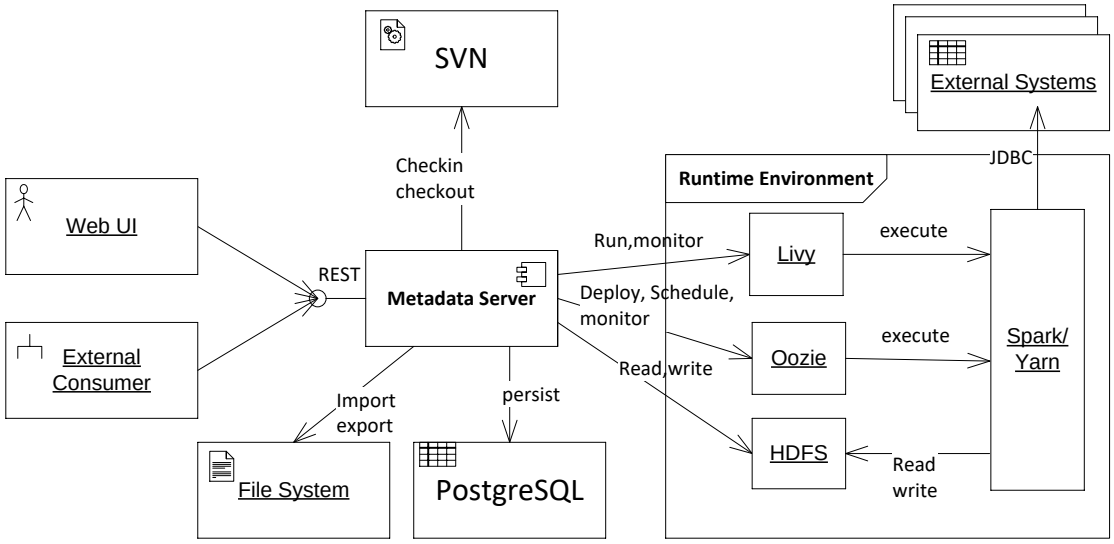
Support for dev->test->prod development cycle.

- Import/Export metadata to/from file system
- Move metadata (full, or project based) to a new environment
- Rewrite urls, passwords etc., while moving to the new environment

Supplement tools

- HDFS Console: browse, download and upload files from/to **HDFS** file system
- Livy Console: browse tasks on **Livy** server, view logs, cancel tasks
- Oozie Console: browse workflow and coordinator jobs on Ozzie, view logs and database rejects, cancel or restart jobs
- Object Explorer: explore and filter metadata object tree

Architecture overview



Framework internals

The framework is developed in model-driven architecture (**MDA**).

For model management, Eclipse Modelling Framework (**EMF**) is used.

To persist models, **PostgreSQL/Hibernate/Teneo** is used.

For model validation, model-to-model (**M2M**) and model-to-text (**M2T**) transformation, **Eclipse Epsilon** is used.

The following types of models are used internally: Authentication, Relational, ETL, Runtime, DWH, UI, Metadata.