# Second Assignment - Working on Data Ingestion and Transformation

Last modified: 25.02.2026 By Linh Truong (linh.truong@aalto.fi)

> **You are not allowed to share/publish this assignment description**.

> Make sure you follow the rule to avoid academic violation.

The goal of this assignment is to develop and manage data ingestion and transformation tasks in big data platforms and to demonstrate your understanding on key issues of big data ingestion and transformation.

## 1 Introduction

This assignment assumes that **you** (the student doing this assignment) design and implement key features for data ingestion and transformation tasks. You will play two main roles in this assignment:

- platform designer/provider: provides key services for tenants
- tenant developer/user: designs tenant's components and features based on offerings from the platform provider.

## 2 Constraints and inputs for the assignment

We assume that you develop and operate a big data platform **mysimbdp**, of which **mysimbdp-coredms** is the component for big data databases and storage. Overall, you have a set of APIs for your customers to move data being ingested and to store processed data into **mysimbdp-coredms**.

We assume that you have different tenants who need to ingest and transform their original data into **mysimbdp**. Each tenant has its own data (with different syntaxes and/or semantics) but we assume that all tenants use the database/storage models (e.g., relational, document-based, column family-based or file-based models) offered by **mysimbdp**.

> There is no assumption that the data sources and data from two tenants are similar. A platform can be used by multiple tenants. One tenant can run many data producers/consumers, whereas each producer and consumer might use a set of concurrent tasks for writing and reading/analyzing data.

Tenants can design how they would ingest and transform their data into **mysimbdp** using the APIs and components **mysimbdp** offers. We assume you will also have to play the role of tenants to design and implement tenant's components based on offerings from the platform provider.

> **Important note 1**: You can reuse your previous implementation of **mysimbdp-coredms** and **mysimdbp-dataingest**. However, we consider any previous work, which has already been graded, as *external, reusable components*; they must be arranged and declared clearly if included.

> **Important note 2**: Different tenants must not use the same data. You can emulate a new dataset from an existing dataset or split data from a dataset to create suitable test data for different tenants.

> **Important note 3**: If you need a cloud storage for testing, you can ask the TA to get a google storage bucket for testing.

Near real-time ingestion ***must be done*** via messaging systems, using one of the following frameworks/technologies:

- MQTT (e.g., VerneMQ, EQMX), Apache Kafka, Apache Pulsar, RedPanda

Students will be asked to select **one** of the following technologies for **mysimpbdp-coredms**:

- MongoDB, ElasticSearch, Cassandra, Scylla, CockroachDB, Apache Druid, Clickhouse, Apache Pinot

> *If you want to work with other databases, you can discuss with the responsible teacher.*

you must select real **datasets from** the following datasets as input data for examples/testings

- The list of datasets: https://github.com/rdsea/bigdataplatforms/blob/master/data/README.md

> *You can bring your own data through a discussion with the responsible teacher. It is up to you to decide if you want to have an advanced scenario of multiple types of databases/data storage.*

and you can only use the following programming languages:

- Python, JavaScript/TypeScript, Java, GoLang

# 3 Requirements and delivery

## Part 1 - Streaming data ingestion (weighted factor for grades = 3)

1. Tenants will send their original data via a messaging system for ingestion as **bronze data** in the view of the tenants.The messaging system **mysimbdp-messagingsystem** is provisioned by **mysimbdp**. Tenants will develop message structures for data records being sent. Tenants will develop ingestion workers, **streamingestworker**, which read data from the messaging system, then ingest the data into **mysimbdp-coredms**. For near real-time ingestion, explain your design for the multi-tenancy model in **mysimbdp**: which parts of the **mysimbdp** will be shared for all tenants, which parts will be dedicated for individual tenants so that **mysimbdp** can add and remove tenants based on the principle of pay-per-use. (1 point)

2. Design and implement a component **mysimbdp-streamingestmanager**, which can start and stop **streamingestworker** instances *on-demand*. **mysimbdp** imposes the model that **streamingestworker** has to follow so that **mysimbdp-streamingestmanager** can invoke **streamingestworker** as a blackbox. Explain the model w.r.t. steps and what the tenant has to do in order to develop **streamingestworker** to work with **mysimbdp-streamingestmanager**. (1 point)

3. Develop different **streamingestworker** for at least two tenants for testing. Show the performance of ingestion tests, including failures and exceptions, for at least 2 different tenants in your test environment, under normal assumed loads. Demonstrate and report performance for heavy/intensive workload of incoming ingestion data but with a limited capability, under-provisoning of **streamingestworker** due to the limitation of **mysimbdp** resources. Note to test **streamingestworker** you need some *data producers sent the original data*. (1 point)

4. A **streamingestworker** decides to report its processing performance, including average ingestion time, total ingestion data size, and number of records to **mysimbdp-streamingestmonitor** as an

observability component. Design the report format and explain possible components, flows and the mechanism for reporting. (1 point)

5. Implement a feature in **mysimbdp-streamingestmonitor** to receive the report from **streamingestworker**. Based on the report from **streamingestworker**, when the performance is below a threshold, e.g., average ingestion time is too low, **mysimbdp-streamingestmonitor** decides to inform **mysimbdp-streamingestmanager** about the situation. Implement a feature in **mysimbdp-streamingestmanager** to receive information informed by **mysimbdp-streamingestmonitor**. Demonstrate these features. (1 point)

## Part 2 - Silver data transformation with batch processing (weighted factor for grades = 3)

1. From the **bronze data**, tenants will design transformation pipelines which analyze the bronze data to create **silver data** and the platform will execute these pipelines for the tenants. Design a schema for a set of constraints for *tenant service agreement* that **mysimbdp** will support the pipelines. Explain why you, *as a platform provider*, decide and use such constraints. Implement these constraints into simple configuration files. Provide two different examples (e.g., JSON or YAML) for two different tenants to specify constraints on service agreement for the tenants and explain why such constraints are suitable for the tenants. (1 point)

2. Each tenant has a caching directory **tenant-caching-dir** (local disk within the platform or a cloud storage bucket) provisioned and managed by **mysimbdp**. Each tenant provides its silver transformation pipelines, **silverpipeline**, which will extract the tenant's bronze data, store the extracted data in *files* in **tenant-caching-dir**, and transform the extracted data to produce silver data stored into **mysimbdp-coredms**. *As a tenant*, explain the design of **silverpipeline** and provide one implementation. **silverpipeline** follows the guideline of **mysimbdp** given in the next Point 3. (1 point)

3. The **mysimbdp** provider provisions an execution environment for running **silverpipeline** from tenants. As the **mysimbdp** provider, design and implement a component **mysimbdp-batchmanager** that invokes tenant's **silverpipeline** to perform the ingestion/transformation for available files in **tenant-caching-dir**. **mysimbdp** imposes the model that **silverpipeline** has to follow but **silverpipeline** is, in principle, a blackbox to **mysimbdp-batchmanager**. Explain how **mysimbdp-batchmanager** knows the list of **silverpipeline** and schedules the execution of **silverpipeline** for different tenants. (1 point)

4. Develop test **silverpipeline**, test data, and test service agreements for tenants according to a deployment of **mysimbdp**. Show the performance of ingestion tests, including failures and exceptions, for **2 different tenants** in your test environment and constraints. Demonstrate examples in which **silverpipeline** will not be executed due to a violation of constraints. Present and discuss the performance of **silverpipeline** when using *local disk vs cloud storage* for **tenant-caching-dir** in your tests. (1 point)

5. Implement and provide logging features for capturing successful/failed **silverpipeline** and its tasks, transformation time, data size, etc. Logging information must be stored in separate files, databases or a monitoring system for observability of **silverpipeline**. Explain how **mysimbdp** could use such logging information. Show and explain simple statistical data extracted from the logs for individual tenants and for the whole platform with your tests. (1 point)

## Part 3 - Integration and Extension (weighted factor for grades = 1)

> **Notes**: no implementation is required for this part

1. Produce an integrated architecture, *with a figure*, for the logging and monitoring of both near real-time ingestion and silver transformation features (Part 1, Points 4-5 and Part 2, Point 5). Explain how a platform provider could know the amount of data ingested/processed and existing errors/performance for individual tenants. (1 point)

2. In the stream ingestion worker, assume that a tenant has to ingest the same data but to different data sinks, e.g., **mybdp-coredms** for bronze data storage and a new **mybdp-extradatasink** component. What features/solutions can you provide and recommend to your tenant? (1 point)

3. In the case of near real-time ingestion, assume that we want to (i) detect the quality of data being ingested and store only data with a predefined quality of data and (ii) save the quality of data detected into the platform. *Given your implementation in Part 1*, how would you suggest a design/change for achieving this goal? (1 point)

4. Assume a tenant has multiple **silverpipeline**. Each is suitable for a type of bronze data, has different workloads and service agreements, such as complex transformation or feature engineering (e.g., lead to different CPU needs, memory consumption, and execution times). How would you extend your design and implementation in Part 2 to support the above-mentioned situation? (1 point)

5. Assume **silverpipeline** is too complex with two parts: (i) access the bronze data and prepare the data into files and (ii) analyze the prepared data to produce silver data. What would be your recommendations for redesigning **silverpipeline** to improve the performance, fault management, and maintenance, for the tenant. Explain the reasons for your recommendations. (1 point)

> You will address the above-mentioned points by writing your solutions into the design document (template: see the git assignment template https://version.aalto.fi/gitlab/bigdataplatforms/assignment-nr-studentid) and provide source files. See the guideline of the template for organizing code and document how to run the code. Set **nr as 2** for folder/document names.

# 4 Other notes

Remember that we need to **reproduce** your work. Thus:

- Include the (adapted) deployment scripts/code you used for your installation/deployment
- Explain steps that one can follow in doing the deployment
- Include logs to show successful or failed tests/deployments
- Include git logs to show that you have incrementally solved questions in the assignment