## Weeks 11: Final Project Assignment

### Objective

The objective of this project is to build a data pipeline that integrates at least four major components from the following ecosystem: NiFi, HDFS, Hive, HBase, and Spark. You will demonstrate the seamless flow of a single dataset through multiple stages, processing and transforming the data using the selected technologies. This project will showcase your understanding of how to move data between systems in a distributed environment while leveraging the power of big data tools for storage, querying, and processing.

### Requirements

#### Pipeline Composition

Your pipeline must utilize at least four components from the following list:

* **NiFi** for data ingestion
* **HDFS** for data storage
* **Hive** for data warehousing
* **HBase** for NoSQL storage
* **Spark** for data processing, querying, and machine learning

#### Dataset

* Use a dataset of your choice — **do not reuse any course-provided datasets**.
* The dataset must be hosted on **GitHub**, and you must use the **direct file URL** as demonstrated throughout the semester.
* Update the NiFi HTTP processor to point to your dataset’s GitHub URL.
* Rename your dataset either:
  + during ingestion using the **UpdateAttribute** processor in NiFi, or
  + after it is in HDFS using hdfs dfs -mv.

#### Component Integration

The components must be fully connected and work together in sequence. A typical flow might look like this:

1. Use NiFi to ingest data and load it into HDFS.
2. Move the dataset into Hive for querying.
3. Apply **Spark MLlib** to the dataset (refer to SparkMachineLearning.docx).
4. Store performance metrics (accuracy, precision, recall, etc.) into HBase (see WriteHBasePySpark.docx).
5. Retrieve stored metrics from HBase (similar to Week 6).

### Critical System Resource Tip – Stop NiFi After Data Load

Your VM has **8 CPUs and 16 GB RAM**, which is extremely limited to keep cloud costs low. You cannot run all services simultaneously.

* After loading your dataset into HDFS with NiFi, **stop NiFi before continuing** with Hive, Spark, or HBase.
* In real-world production environments, organizations operate at a vastly larger scale with hundreds of servers and terabytes of memory.
* In this academic environment, you must **manage limited resources efficiently** to prevent system crashes.

### Submission Guidelines

Submit a **comprehensive report** that includes the following:

* A detailed write-up of what you attempted, what worked, and what did not.
* A discussion of **challenges and issues encountered**.
* The **link to your dataset** (GitHub direct URL).
* Screenshots showing successful execution at each stage (NiFi, HDFS, Hive, Spark, HBase).
* The full code used (NiFi templates, PySpark scripts, Hive queries, etc.).
* A clear and well-structured format with labeled sections:
  + Introduction
  + Dataset
  + Pipeline Overview
  + Issues Encountered
  + Screenshots
  + Code
  + Conclusion

### Evaluation Criteria

* Completeness and correctness of your pipeline implementation.
* Effective use of the technologies taught in the course.
* Uniqueness of your dataset and schemas.
* Effort and depth of your analysis.
* Clarity and organization of your report.

### Examples

You can refer to examples from the following repository:  
<https://github.com/bellevue-university/dsc650-updated/tree/main/week11-12/assignment/examples>

You are encouraged to use these resources as guides, but you cannot use the same datasets or schemas provided in these examples. The Docker networking has already been configured and tested for seamless connectivity between containers.

* **Final\_Project.json**: NiFi template that downloads customers-100.csv and loads it into HDFS. You can use your own dataset, load it to HDFS, and then put it in a Hive table. You may also explore Hive external tables.
* **SparkAddLibraries.docx**: Shows how to add a Python library and run it with PySpark
* **HdfsToHivePySpark.docx**: Reads a CSV from HDFS, creates a Hive table in PySpark, loads data into it, and queries it.
* **QueryHivePySpark.docx**: Demonstrates creating a Hive table and querying it with PySpark.
* **WriteHBasePySpark.docx**: Walks through installing happybase, creating an HBase table, and writing data with PySpark.
* **SparkMachineLearning.docx**: Shows how to create a linear regression ML model with PySpark, Hive, and Spark MLlib.
* **HiveQueryHbase.docx**: Demonstrates creating an HBase table, loading data into it, and querying via Hive.