Data Integration Automation from Heterogeneous Data Sources for Smart Farming Data Lake

Samuel Indra Gunawan Situmeang1, a), Nixon Daniel Hutahaean2, a),Irfan Jumadin Siregar3, a), Agnes Sagita Lumbantobing3, a), Parmonangan Rotua Togatorop4, b), Humasak Tommy Argo Simanjuntak5, b)

Author Affiliations

1Faculty of Informatics & Electrical Engineering, Institut Teknologi Del, Laguboti, Toba 22381, Indonesia   
(use complete addresses, including country name or code).

2. Bandung Fe Institute, Sukajadi, Bandung 40164, Indonesia

Author Emails  
a) samuel.situmeang@del.ac.id  
b) mona.togatorop@del.ac.id

c) *humasak@del.ac.id*

d) *iss19035@students.del.ac.id*

e) *iss19021@students.del.ac.id*

f) *iss19013@students.del.ac.id*)

**Abstract.** Smart Farming combines Information and communication technologies (ICT) with traditional agricultural practices to improve the quality and quantity of agricultural products. These ICTs could be Unmanned Aerial Vehicles (UAVs) or drones, artificial intelligence, robots, and sensors. In Smart Farming systems, various data types are needed, such as food prices, sensors, weather, images, and video. The data can be structured, semi-structured, or unstructured. Therefore, a system that can integrate Smart Farming data with versatility characteristics is needed to empower various types of analysis. In this study, the authors suggested using a Smart Farming Data Lake System based on Apache Airflow as data integration automation technology, Hadoop Distribution File System (HDFS) as data storage technology, and Metabase as dashboard technology. The evaluation result shows that all the data lake system functionality can run smoothly. In addition, the experimental results show that the implemented system can operate stably and without crashes within a time range of 1 hour with process intervals every 5 minutes.

# INTRODUCTION

Smart Farming combines traditional agricultural techniques with innovative solutions based on Information and communication technologies (ICT) using technologies such as Unmanned Aerial Vehicles (UAVs) or drones, machine learning, big data, and sensors, thereby reducing labour consumption and improving agricultural production efficiency [1]. As for analytics that can be done in Smart Farming, such as growth analysis [2], yield prediction [2] [3], quality maintenance [2][3], animal husbandry and aquaculture [2], and climate condition monitoring [3]. Such analyses have data needs that are varied or heterogeneous in shape, such as structured, semi-structured, and unstructured data with different data types and from different sources. The sources of data used in this study are semi-structured weather forecast data obtained from the BMKG website, structured food price data from the PIHPS website, sensor data from a structured database from InfluxDB, and image and video data from an unstructured database from cameras. To combine this heterogeneous data, an integration mechanism and an adequate storage system are required.

There are several systems proposed to facilitate data storage mechanisms in Smart Farming systems, such as transactional databases [4], data warehouses [5], and data lakes [6]. Transactional databases and data warehouses require progressive development and analysis processes. The phase of transactional database development includes the requirement gathering activity, i.e., the collection of requirements related to data and the conditions in which the data needs to be accessed, the analysis of data-related needs that have been collected, and the results can be used in the form of entity-relationship diagrams (ERD), the design of the database based on the results of the analysis, implementation, testing, and maintenance [7]. Like transactional databases, the data warehouse development phase has a similar flow. However, the analysis and design phase is carried out based on dimensional modeling when its development uses a denormalized approach. Data warehouse development phases include business determination processes based on needs, declaring grains, identifying dimensions, and identifying facts. Fact and dimension components are combined in a scheme. The dimensional scheme can be a star, snowflake, or constellation [8]. Unlike transactional databases and data warehouses, data lakes store data in native or raw formats [9][10]. So the data lake allows the transmission of unlimited amounts of data in real-time and data collection from various sources more efficiently than transactional databases or data warehouses. Based on this, using lake data storage systems for Smart Farming systems is better than transactional databases and data warehouses.

# Literature review

## Smart Farming

Smart farming is an approach that utilizes technologies like the internet, cloud computing, and IoT to enhance agricultural productivity and efficiency [1][11]. It combines traditional farming techniques with innovative solutions like drones, machine learning, big data, and sensors to optimize agricultural yields. Sensors measure and monitor parameters like soil moisture, air temperature, and CO2 levels, while robots perform tasks like pesticide spraying, tillage, and harvesting. Drones observe plant growth and collect data, while cloud computing enables farmers to access real-time data. Data analysis is crucial for developing better farming strategies and forecasting future conditions. Analyzing weather data, soil type, water use, and production analysis helps optimize crop yields and identify areas for productivity improvement.

## Data Lake

Data Lake is a centralized repository designed to store, process, and analyze very large amounts of data, including structured, semi-structured, and unstructured data [12]. The main goal of a data lake is to facilitate the ingestion and storage of raw data on a large scale so that the data can be further integrated [7]. Any data stored in a data lake must be provided with metadata so that it can be properly managed and prevent "data swamps" from occurring. Within the data lake ecosystem, metadata plays an important role in facilitating information retrieval and data queries. Some of the important characteristics of a data lake include:

* Large scale: Data lakes enable organizations to store vast amounts of data.
* Diverse data sources: Users can store data from various sources, both internal, including public data.
* Heterogeneous data: Data lake does not require special formats or data structures, so users can store data in various formats such as text, sound, images, and video, as well as in the form of structured, semi-structured, or unstructured data.
* Flexible schema: Data lake allows data storage without having to define a schema first, so data can be stored directly.

Data lakes play an important role in big data management and analysis. In the context of smart farming, data lakes can be used as an infrastructure to store and manage heterogeneous agricultural data, including data from sensors, drones, robots, and other data sources. By leveraging data lakes, farmers can efficiently integrate and analyze data to support better decision-making in farming

## Apache Airflow

Apache Airflow is an open-source platform developed by Airbnb for managing automated and scheduled workflow execution [13][14]. It offers flexibility in handling various tasks, with diverse operators like BashOperator, PythonOperator, and SSHOperator [13]. The core concept is the Directed Acyclic Graph (DAG), which represents structured task workflows with sequencing and dependencies [13]. Apache Airflow can process data from various sources, perform transformations, and send processed data to storage sources. It also offers scheduling features to automate workflow execution at specified times, enhancing data processing efficiency and accelerating business processes [14].

## Metabase

Metabase is a web-based open-source platform that is used as a visualization tool for a query to make it easier for a company to get information from data [15][16]. Metabase allows users to filter and group data according to their needs with or without using structured query language (SQL) [15][16]. Metabase has a feature to monitor questions submitted by users to obtain information or insights from available data [15]. These questions will later be displayed visually as graphs and diagrams, and then they can be stored and managed in a dashboard that can be accessed on the Metabase web page, which has an organized interface [15]. Metabase is here as an alternative business intelligence tool with an open-source license that can help companies get information from data quickly and easily.

## Analysis Extract and Load

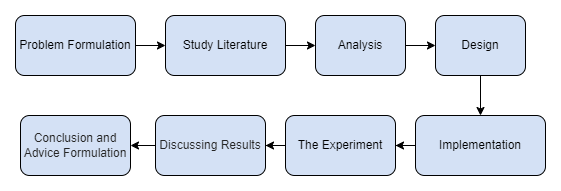
In this subchapter will explain the Extract and load analysis based on the data source. Extract and Load are processes used in the data lake to collect, load data. Data is first extracted from the source, then loaded into the data lake. Some of the data sources used are sensor data obtained from the InfluxDB Database, the PIHPS website which stores information on Indonesian food prices and the BMKG website which stores information on Indonesian weather forecasts, and cameras used to take pictures and record videos of plants.

In the data lake architecture, the data to be absorbed in the data lake is processed without any transformation, this will minimize the time between data extraction from the source and absorption in the data lake. So there is an extraction process to get data from various data sources, such as on the PIHPS website and the BMKG website, web scraping is carried out, next from the agricultural environmental sensor data source using the API from the InfluxDB Database, and for data sources from the camera will use the SCP command to transfer image and video data that has been taken by the Raspberry Pi camera.

There is no transformation process because basically the data lake system can receive raw data. Then the load process can be carried out to continue into the data lake system to be stored. The author chooses the Lambda Extract and load mechanism, because it matches the way the data lake stores data from various data sources with real-time and batch processes.

# METHODS

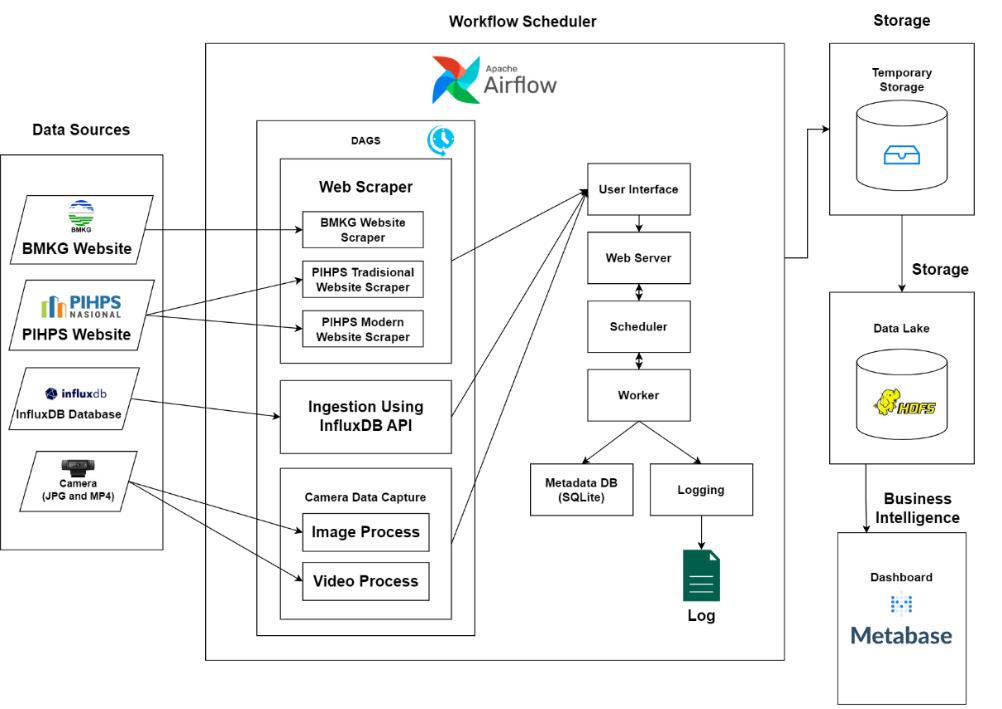
This phase of research is used as a benchmark in the implementation and as a measure of the completion of each stage of research. Analysis is performed to determine the structure of the data and methods used at the implementation stage. The stage of the research to be carried out is shown in Figure. 1 below.



## Figure 1. Research Stages

## General Design System

In this sub-chapter, we will describe the process of integrating data from heterogeneous data sources into smart farming data lakes. The architectural design of the data lake system in this study is shown in Fig. 1. The data sources used are the PIHPS website (CSV - structured data), the BMKG website (JSON - semi-structured data), InfluxDB (structured data), and the camera (unstructured data). Apache Airflow will utilize a Directed Acyclic Graph (DAG) to extract data from each data source, with its supporting components such as user interface, web server, scheduler, worker, database metadata, and logging. The extracted data will be automatically transferred to HDFS. After the data is successfully stored in HDFS, the authors will use Metabase to create a dashboard that displays process-related information or DAGs that have been executed using Apache Airflow.



## Figure 2. General Design System

## Result of Data Analysis

The data analysis aims to help the writer determine whether the data sources are categorized as structured, semi-structured, or unstructured. These sources include the PIHPS website with CSV format (structured), BMKG website with JSON format (semi-structured), InfluxDB (structured), and cameras (unstructured). Additionally, the analysis ensures that data stored in the data lake maintains its original structure from these sources.

* BMKG Website

The weather forecast data includes attributes such as city, morning, daytime, nighttime, and early morning weather predictions, along with temperature and humidity. BMKG website data is collected daily at 03:55 WIB in batches to cover these forecasts and avoid operational hours and website traffic spikes.

* PIHP Website

The food price data includes attributes such as commodity, market type, price information type, and date. It's collected from the PIHPS website daily at 03:55 WIB in batches for stability, avoiding network issues, and website traffic spikes during operational hours.

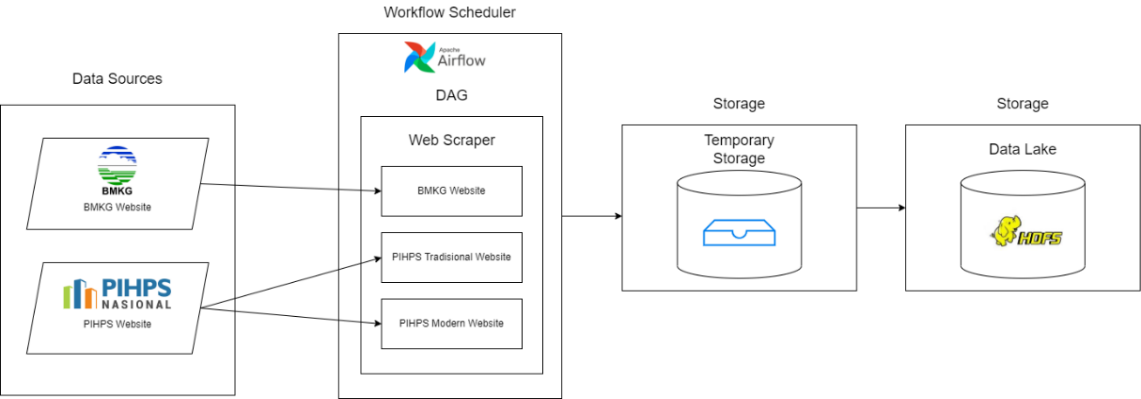
* InfluxDB Database

The data is stored in InfluxDB and updated every hour. Sensor data includes attributes such as start, stop, time, value, field, measurement, and host. Data retrieval occurs at 00:35 WIB to prevent disruptions or interruptions during operational hours for the automated process.

* Camera

The Raspberry Pi captures image and video information to extract file details. Attributes include Location, File type, total size of files, Size on disk, Last modification, Last access, and Last permission change. Image and video data is collected in real-time, every hour starting at 00:00 WIB. Videos are recorded for 30 minutes to avoid conflicts with the scheduled hourly capture process.

### **BMKG and PIHPS Website Scraper**

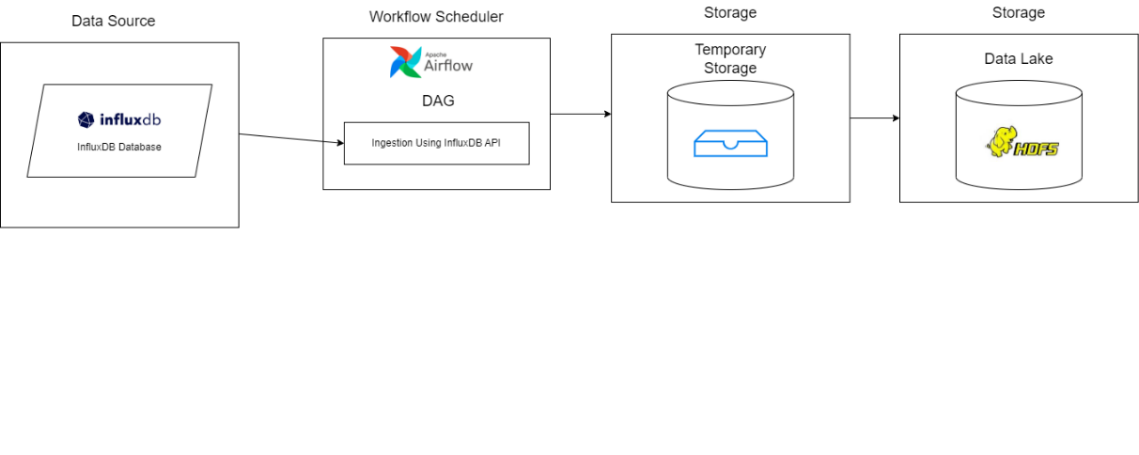


## Figure 3. BMKG and Website Scraper

The Extract and Load design for the BMKG and PIHPS web data involves two processes: data extraction and data loading. In the data extraction process, a web scraping program is executed to retrieve data from the websites using Python libraries such as Selenium and Webdriver Manager. The output is CSV files containing weather forecast data from the BMKG website and food price data from the PIHPS website. In the data loading process, a Hadoop command is run to move the folders containing the weather forecast and food price files into HDFS for storage.

### **InfluxDB Database**

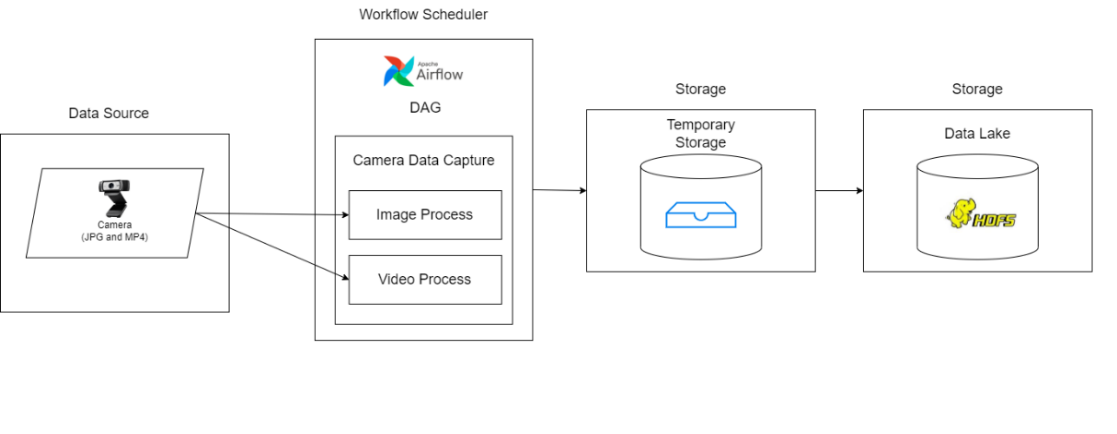
The Extract and Load design for the InfluxDB sensor data involves two processes: data extraction and data loading. In the data extraction process, a program accesses the InfluxDB API using specific endpoints to retrieve sensor data. This data is then stored in a dataframe and saved as a CSV file. In the data loading process, a Hadoop command is executed to move the folder containing the sensor data files into HDFS for storage.



## Figure 4. InfluxDB Database

### **Camera**

The camera data source design involves data extraction and data loading. The first Raspberry Pi captures jpg-format images using the fswebcam command, which are then transferred to local storage using the scp command and loaded into HDFS. The second Raspberry Pi records mp4-format videos using the ffmpeg command, which are then transferred to local storage using the scp command and loaded into HDFS. Finally, the program clears the image folder's contents on both Raspberry Pis using SSH.



## Figure 5. Camera

1. Temporary Storage

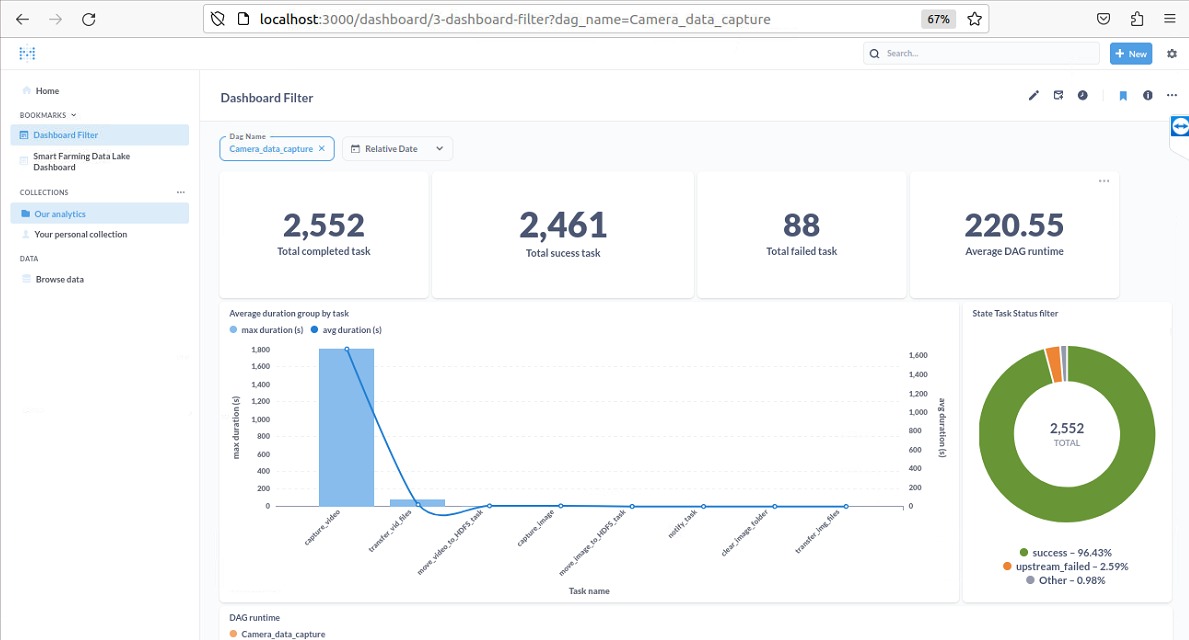
Temporary storage is a temporary storage area in the Linux system used to hold data during the processes of extraction, recording, and data retrieval before being transferred to permanent storage such as HDFS. Temporary storage is in the form of folders in the Linux system.

1. Data Lake

A data lake is employed to store various types of data structures, including structured data like PIHPS data, semi-structured data from BMKG, structured data from InfluxDB, and unstructured data from cameras. The author utilizes HDFS as the data lake to accommodate this heterogeneous data collectively.

1. Dashboard

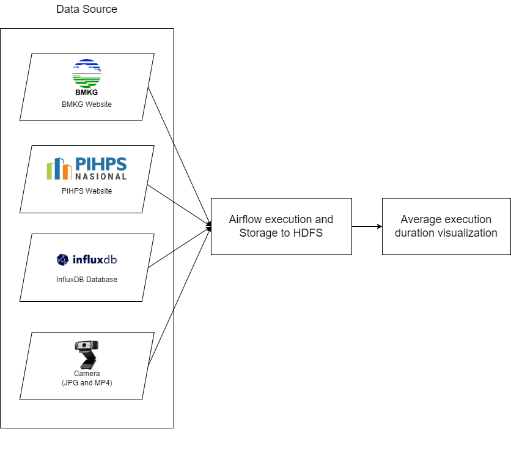
The author employs Metabase to create a dashboard that displays information related to the processes or DAGs (Directed Acycic Graphs) from Apache Airflow, which are stored in a log or database.



## Figure 6. Dashboard Metabase

## Experiment

In this sub-chapter, we will explain the design of the experiment based on the stability aspects of the data lake system. The scenario of the stability experiment can be seen in Fig. 7, which involves four variations of data sources that are processed automatically using Apache Airflow. Each data source is processed every 5 minutes within a 1-hour time range for BMKG and PIHPS website data sources, APIs from InfluxDB databases, and cameras. For the video camera data source, the process is set to record a 60-second video every 5 minutes, with a time interval of 1 hour. The transfer of data that has been extracted into HDFS is also part of the process. Based on the executed processes, 12 average duration values are generated for each process. The results of each process are then visualized to assess the stability of the data lake system.



## Figure 7. Experiment

# RESULT and Discussion

Based on the scenario test results that have been performed using the Web Scraping process on the BMKG website and PIHPS, Ingestion using InfluxDB, and Camera Data Capture from the camera, if the actual result corresponds to the expected result, then the automation process runs stable. Based on the results of the test case scenarios of heterogeneous data sources consisting of structured data, semi-structured data, and unstructured Data, the obtained data integration process has been achieved automatically, where data from various sources can be extracted automatically using Apache Airflow and stored into HDFS. Whereas if the actual result does not match the expected result, it is necessary to identify the cause of the non-conformity. This may be due to changes in the structure or writing of the code that need to be repaired or an unstable internet connection that allows the stability of the system to crash.

## Table 1. Test Case Summary Report Proses Web Scraping

|  |  |  |
| --- | --- | --- |
| Test Summary Report | | |
| Nomor *Test Scenario* | | PDLK – 01 |
| *Test Scenario Name* | | *Test Scenario* Proses Web Scraping dari Sumber Data *Website* BMKG dan PIHPS |
| *Test Type* | | *Stability Test* |
| No. of Tests | Pass | 12 |
| Fail | 0 |
| Approval Time/Date | | 10 July 2023 |
| Approved by | | Agnes Sagita Lumbantobing |

## Table 2. Test Case Summary Report Proses Ingestion using InfluxDB

|  |  |  |
| --- | --- | --- |
| Test Summary Report | | |
| Nomor *Test Scenario* | | PDLK – 02 |
| *Test Scenario Name* | | *Test Scenario Ingestion using* InfuxDB dari Sumber Data API Basis Data InfluxDB |
| *Test Type* | | *Stability Test* |
| No. of Tests | Pass | 5 |
| Fail | 0 |
| Approval Time/Date | | 10 July 2023 |
| Approved by | | Irfan Jumadin Siregar |

## Table 3. Test Case Summary Proses Capture Data Camera

|  |  |  |
| --- | --- | --- |
| Test Summary Report | | |
| Nomor *Test Scenario* | | PDLK – 03 |
| *Test Scenario Name* | | *Test Scenario* Proses *Capture Data Camera* dari Sumber Data Kamera |
| *Test Type* | | *Stability Test* |
| No. of Tests | Pass | 7 |
| Fail | 0 |
| Approval Time/Date | | 10 July 2023 |
| Approved by | | Nixon Daniel Hutahaean |

## 4.1. Result of Experiment

### **4.1.1. Duration and Frequency of Crashes in the Scraping Process of PIHPS and BMKG Websites**

In Table 4 conducting stability experiments for the BMKG and PIHPS Website data sources, the web scraper process is run every 5 minutes in a span of 1 hour automatically**.**

## Table 4. Stability Experiment of Web Scraping Process from PIHPS and BMKG Web Data Sources.\

|  |  |  |
| --- | --- | --- |
| **Timestamp** | **Average duration (seconds)** | **Crash Frequency** |
| 2023-06-05 02:40 | 139,44 | 0 |
| 2023-06-05 02:45 | 142,13 | 0 |
| 2023-06-05 02:50 | 139,84 | 0 |
| 2023-06-05 02:55 | 137,52 | 0 |
| 2023-06-05 03:00 | 141,76 | 0 |
| 2023-06-05 03:05 | 138,1 | 0 |
| 2023-06-05 03:10 | 142,17 | 0 |
| 2023-06-05 03:15 | 138,36 | 0 |
| 2023-06-05 03:20 | 143,26 | 0 |
| 2023-06-05 03:25 | 140,54 | 0 |
| 2023-06-05 03:30 | 142,09 | 0 |
| **2023-06-05 03:35** | **143,4** | **0** |

### **4.1.2. Duration and Crash Frequency of Ingestion Process using InfluxDB API**

In Table 5 conducting stability experiments for the InfluxDB Database API data source, the Ingestion Using InfluxDB API process is run every 5 minutes in a span of 1 hour automatically.

**Table 5.** Ingestion Stability Experiment using Ingestion InfluxDB from Database API Data Source

|  |  |  |
| --- | --- | --- |
| **Timestamp** | **Average duration (seconds)** | **Crash Frequency** |
| 2023-06-06 02:35 | 66,87 | 0 |
| 2023-06-06 02:40 | 64,27 | 0 |
| 2023-06-06 02:45 | 67,48 | 0 |
| 2023-06-06 02:50 | 68,13 | 0 |
| 2023-06-06 02:55 | 69,81 | 0 |
| 2023-06-06 03:00 | 72,83 | 0 |
| 2023-06-06 03:05 | 79,6 | 0 |
| 2023-06-06 03:10 | 78,19 | 0 |
| 2023-06-06 03:15 | 78,45 | 0 |
| 2023-06-06 03:20 | 79,99 | 0 |
| 2023-06-06 03:25 | 95,51 | 0 |
| 2023-06-06 03:30 | 88,14 | 0 |

### **4.1.3. Duration and Crash Frequency of the Camera Data Capture Process**

In conducting stability experiments for camera data sources, the Camera Data Capture process is run every 5 minutes within a span of 1 hour automatically.

**Table 6.** Stability Experiment of Camera Data Capture Process from Camera Data Source

|  |  |  |
| --- | --- | --- |
| **Timestamp** | **Average duration (seconds)** | **Crash Frequency** |
| 2023-06-12 08:00 | 26.64 | 0 |
| 2023-06-12 08:05 | 27.25 | 0 |
| 2023-06-12 08:10 | 26.89 | 0 |
| 2023-06-12 08:15 | 27.02 | 0 |
| 2023-06-12 08:20 | 26.67 | 0 |
| 2023-06-12 08:25 | 26.86 | 0 |
| 2023-06-12 08:30 | 26.51 | 0 |
| 2023-06-12 08:35 | 26.13 | 0 |
| 2023-06-12 08:40 | 25.64 | 0 |
| 2023-06-12 08:45 | 26.43 | 0 |
| 2023-06-12 08:50 | 26.25 | 0 |
| 2023-06-12 08:55 | 26.58 | 0 |

# CONCLUSION

Based on the results of the final task research, several conclusions can be drawn. Firstly, the process of automatically integrating data from heterogeneous data sources, consisting of structured data, semi-structured data, and unstructured information, has been successfully implemented using Apache Airflow, HDFS, and Metabase technologies. Data is extracted from each data source and stored in HDFS without altering the data structure or file format from the original file. In the result chapter and discussion of the implementation of the camera data source, it has been discussed that the video duration is set to 30 minutes to avoid collisions with scheduling configured on the Airflow process. Future research may consider adjusting recording duration and scheduling intervals to avoid conflicts with subsequent processes due to timing.

Secondly, complete and consistent use of metadata is required for every data stored in the data lake system. Metadata can include essential information such as pickup data, data types, and other relevant attributes. Identifying unique data, such as giving a unique ID to each entity, will help in showing correlations and relationships between entities or data

# References

[1] V. Moysiadis, P. Sarigiannidis, V. Vitsas, dan A. Khelifi, “Smart Farming in Europe,” Computer Science Review, vol. 39, hal. 100345, Feb 2021, doi: 10.1016/j.cosrev.2020.100345.

[2] S. De Alwis, Z. Hou, Y. Zhang, M. H. Na, B. Ofoghi, dan A. Sajjanhar, “A survey on smart farming data, applications and techniques,” Computers in Industry, vol. 138, hal. 103624, Jun 2022, doi: 10.1016/j.compind.2022.103624.

[3] A. Rehman, T. Saba, M. Kashif, S. M. Fati, S. A. Bahaj, dan H. Chaudhry, “A Revisit of Internet of Things Technologies for Monitoring and Control Strategies in Smart Agriculture,” Agronomy, vol. 12, no. 1, hal. 127, Jan 2022, doi: 10.3390/agronomy12010127.

[4] S. Wolfert, L. Ge, C. Verdouw, dan M. J. Bogaardt, “Big Data in Smart Farming – A review,” Agricultural Systems, vol. 153, hal. 69–80, Mei 2017, doi: 10.1016/j.agsy.2017.01.023.

[5] M. T. Kechadi, N. A. Le Khac, dan V. M. Ngo, “Data warehouse and decision support on integrated crop big data,” International Journal of Business Process Integration and Management, vol. 10, no. 1, hal. 17, 2020, doi: 10.1504/ijbpim.2020.10035688.

[6] E. M. Ouafiq, R. Saadane, A. Chehri, dan M. Wahbi, “Data Lake Conception for Smart Farming: A Data Migration Strategy for Big Data Analytics,” in Smart Innovation, Systems and Technologies, vol. 310, 2022, hal. 191–201. doi: 10.1007/978-981-19-3455-1\_15.

[7] A. Kaulins dan A. Borisov, “Building Ontology from Relational Database/ Ontoloģiju izveide no relāciju datubāzes/ Построение онтологии по реляционной базе данных,” Information Technology and Management Science, vol. 17, no. 1, Jan 2015, doi: 10.1515/itms-2014-0006.

[8] V. M. Ngo, N.-A. Le-Khac, dan M.-T. Kechadi, “Designing and Implementing Data Warehouse for Agricultural Big Data,” 2019, hal. 1–17. doi: 10.1007/978-3-030-23551-2\_1.

[9] H. Mehmood et al., “Implementing big data lake for heterogeneous data sources,” in Proceedings - 2019 IEEE 35th International Conference on Data Engineering Workshops, ICDEW 2019, Apr 2019, hal. 37–44. doi: 10.1109/ICDEW.2019.00-37.

[10] P. Ren et al., “MHDP: An Efficient Data Lake Platform for Medical Multi-source Heterogeneous Data,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 12999 LNCS, 2021, hal. 727–738. doi: 10.1007/978-3-030-87571-8\_63.

[11] S. Rooney, D. Bauer, L. Garces-Erice, P. Urbanetz, F. Froese, dan S. Tomic, “Experiences with managing data ingestion into a corporate datalake,” Proceedings - 2019 IEEE 5th International Conference on Collaboration and Internet Computing, CIC 2019, hal. 101–109, 2019, doi: 10.1109/CIC48465.2019.00021.

[12] A. Gorelik, What Is a Data Lake? O’Reilly Media, Inc., 2020.

[13] P. Singh, “Airflow,” in Learn PySpark, Berkeley, CA: Apress, 2019, hal. 67–84. doi: 10.1007/978-1-4842-4961-1\_4.

[14] S. Haines, “Workflow Orchestration with Apache Airflow,” in Modern Data Engineering with Apache Spark, Berkeley, CA: Apress, 2022, hal. 255–295. doi: 10.1007/978-1-4842-7452-1\_8.

[15] A. Oliveira dan J. Bernardino, “Evaluating self-service Bi and analytics tools for SMEs,” in ICETE 2020 - Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, 2020, vol. 2, hal. 89–97. doi: 10.5220/0009820400890097.

[16] B. Santos et al., “Open source business intelligence tools: Metabase and redash,” in IC3K 2019 - Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, 2019, vol. 1, hal. 467–474. doi: 10.5220/0008351704670474.

[17] A. Laurent, D. Laurent, dan C. Madera, Data lakes, volume 2. Wiley, 2020. doi: 10.1002/9781119720430.

[18] A. Singh, “Architecture of Data Lake,” International Journal of Scientific Research in Computer Science, Engineering and Information Technology, hal. 411–414, Mar 2019, doi: 10.32628/cseit1952121.

[19] S. G. Manikandan dan S. Ravi, “Big data analysis using apache hadoop,” in 2014 International Conference on IT Convergence and Security, ICITCS 2014, Okt 2014, hal. 1–4. doi: 10.1109/ICITCS.2014.7021746.

[20] Y. Zhai et al., “Hadoop Perfect File: A fast and memory-efficient metadata access archive file to face small files problem in HDFS,” Journal of Parallel and Distributed Computing, vol. 156, hal. 119–130, Okt 2021, doi: 10.1016/j.jpdc.2021.05.011.

[21] R. Li, H. Hu, H. Li, Y. Wu, dan J. Yang, “MapReduce Parallel Programming Model: A State-of-the-Art Survey,” International Journal of Parallel Programming, vol. 44, no. 4, hal. 832–866, Agu 2016, doi: 10.1007/s10766-015-0395-0.

[22] K. Shvachko, H. Kuang, S. Radia, dan R. Chansler, “The Hadoop Distributed File System,” in 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Mei 2010, hal. 1–10. doi: 10.1109/MSST.2010.5496972.

[23] S. Sharma, U. S. Tim, J. Wong, S. Gadia, dan S. Sharma, “A brief review on leading big data models,” Data Science Journal, vol. 13, hal. 138–157, 2014, doi: 10.2481/dsj.14-041.

[24] F. Azzedin, “Towards a scalable HDFS architecture,” in 2013 International Conference on Collaboration Technologies and Systems (CTS), Mei 2013, hal. 155–161. doi: 10.1109/CTS.2013.6567222.

[25] M. A. Martínez-Prieto, A. Bregon, I. García-Miranda, P. C. Álvarez-Esteban, F. Díaz, dan D. Scarlatti, “Integrating flight-related information into a (Big) data lake,” in AIAA/IEEE Digital Avionics Systems Conference - Proceedings, Sep 2017, vol. 2017-Septe, hal. 1–10. doi: 10.1109/DASC.2017.8102023.