



Big Data to Enable Global Disruption of the Grapevine-powered Industries

D3.2 - Data Ingestion & Integration Components

| | |
|---------------------------|---|
| DELIVERABLE NUMBER | D3.2 |
| DELIVERABLE TITLE | Data Ingestion & Integration Components |
| RESPONSIBLE AUTHOR | Mihalis Papakonstadinou (Agroknow) |



Co-funded by the Horizon 2020
Framework Programme of the European Union

| | |
|--------------------------------|--|
| GRANT AGREEMENT N. | 780751 |
| PROJECT ACRONYM | BigDataGrapes |
| PROJECT FULL NAME | Big Data to Enable Global Disruption of the Grapevine-powered industries |
| STARTING DATE (DUR.) | 01/01/2018 (36 months) |
| ENDING DATE | 31/12/2020 |
| PROJECT WEBSITE | http://www.bigdatagrapes.eu/ |
| COORDINATOR | Nikos Manouselis |
| ADDRESS | 110 Pentelis Str., Marousi, GR15126, Greece |
| REPLY TO | nikosm@agroknow.com |
| PHONE | +30 210 6897 905 |
| EU PROJECT OFFICER | Ms. Annamária Nagy |
| WORKPACKAGE N. TITLE | WP3 Data & Semantics Layer |
| WORKPACKAGE LEADER | Agroknow |
| DELIVERABLE N. TITLE | D3.2 Data Ingestion & Integration Components |
| RESPONSIBLE AUTHOR | Ioanna Polychronou (Agroknow) |
| REPLY TO | ioanna.polyxronou@agroknow.com |
| DOCUMENT URL | http://www.bigdatagrapes.eu/ |
| DATE OF DELIVERY (CONTRACTUAL) | 30 September 2018 (M9), 30 September 2019 (M21, Updated Version), 31 July (M30, Final Version) |
| DATE OF DELIVERY (SUBMITTED) | 25 September 2018 (M9), 3 January 2020 (M25, Updated Version), 31 July (M31, Final Version) |
| VERSION STATUS | 3.0 Final |
| NATURE | Report (R) |
| DISSEMINATION LEVEL | Public (PU) |
| AUTHORS (PARTNER) | Mihalis Papakonstadinou (Agroknow) |
| CONTRIBUTORS | Giannis Stoitshs (Agroknow), Timotheos Lanitis (Agroknow) |
| REVIEWER | Florian Schlenz (GEOCLEDIAN) |

| VERSION | MODIFICATION(S) | DATE | AUTHOR(S) |
|---------|-----------------------------------|------------|---|
| 0.1 | Initial Table of Contents | 20/08/2018 | Pythagoras Karampiperis (Agroknow), Panagiotis Zervas (Agroknow), Sotiris Konstantinidis (Agroknow), Antonis Koukourikos (Agroknow) |
| 0.4 | Sections 2 and 3 | 24/08/2018 | Pythagoras Karampiperis (Agroknow), Panagiotis Zervas (Agroknow), Sotiris Konstantinidis (Agroknow), Antonis Koukourikos (Agroknow) |
| 0.6 | Section 4 | 02/09/2018 | Pythagoras Karampiperis (Agroknow), Panagiotis Zervas (Agroknow), Sotiris Konstantinidis (Agroknow), Antonis Koukourikos (Agroknow) |
| 0.8 | Section 1 and 5 | 18/9/2018 | Pythagoras Karampiperis (Agroknow), Panagiotis Zervas (Agroknow), Sotiris Konstantinidis (Agroknow), Antonis Koukourikos (Agroknow) |
| 0.9 | Internal Review | 21/09/2018 | Florian Schlenz (GEOCLEDIAN) |
| 1.0 | Final edits after internal review | 25/09/2018 | Pythagoras Karampiperis (Agroknow), Panagiotis Zervas (Agroknow), Sotiris Konstantinidis (Agroknow), Antonis Koukourikos (Agroknow) |
| 2.0 | Updated version | 3/1/2020 | Panagis Katsivelis (Agroknow) |
| 2.5 | Add Dataflow section | 15/05/2020 | Mihalis Papakonstantinou, Timotheos Lanitis, Giannis Stoitshs (Agroknow) |
| 2.7 | Internal review | 15/06/2020 | Mihalis Papakonstantinou, Timotheos Lanitis, Giannis Stoitshs (Agroknow) |
| 2.9 | Final additions after review | 30/06/2020 | Mihalis Papakonstantinou, Timotheos Lanitis, Giannis Stoitshs (Agroknow) |
| 3.0 | Final version | 31/07/2020 | Mihalis Papakonstantinou, Timotheos Lanitis, Giannis Stoitshs (Agroknow) |

| PARTICIPANTS | | CONTACT |
|---|---|--|
| <p>Agroknow IKE (Agroknow, Greece)</p> |  | <p>Nikos Manouselis Email: nikosm@agroknow.com</p> |
| <p>Ontotext AD (ONTOTEXT, Bulgaria)</p> |  | <p>Todor Primov Email: todor.primov@ontotext.com</p> |
| <p>Consiglio Nazionale DelleRicerche (CNR, Italy)</p> |  | <p>Raffaele Perego Email: raffaele.perego@isti.cnr.it</p> |
| <p>Katholieke Universiteit Leuven (KULeuven, Belgium)</p> |  | <p>Katrien Verbert Email: katrien.verbert@cs.kuleuven.be</p> |
| <p>Geocledian GmbH (GEOCLEDIAN, Germany)</p> |  | <p>Stefan Scherer Email: stefan.scherer@geocledian.com</p> |
| <p>Institut National de la Recherche Agronomique (INRA, France)</p> |  | <p>Pascal Neveu Email: pascal.neveu@inra.fr</p> |
| <p>Agricultural University of Athens (AUA, Greece)</p> |  | <p>Katerina Biniari Email: kbiniari@aua.gr</p> |
| <p>Abaco SpA (ABACO, Italy)</p> |  | <p>Simone Parisi Email: s.parisi@abacogroup.eu</p> |
| <p>SYMBEEOSIS EY ZHN S.A. (Symbeeosis, Greece)</p> |  | <p>Konstantinos Rodopoulos Email: rodopoulos-k@symbeeosis.com</p> |

ACRONYMS LIST

| | |
|------|---------------------------------------|
| BDG | BigDataGrapes |
| BDE | BigDataEurope |
| RDBM | Relational Database Management System |
| HDFS | Hadoop Distributed File System |
| OLTP | Online Transaction Processing |

EXECUTIVE SUMMARY

This accompanying document for deliverable D3.2 Data Ingestion & Integration Components describes the mechanisms and tools that will be used in the BigDataGrapes platform to ingest data of different nature from multiple sources. Also, the document describes the tools that will be used for data integration across the different BigDataGrapes platform layers, as well as for long-term storage and preservation of data.

The document first introduces the big picture of the architecture of the BDG platform and where the ingestion components are positioned. Afterwards, the document describes the different nature of data, and which technologies can be used to facilitate the ingestion process.

Then, the data fusion aspect is described, focusing on how the data will be made available across the BigDataGrapes platform, and how the different BigDataGrapes components will communicate with each other, in an effective and fault-tolerant way.

Moreover, the document provides documentation links for all the described technologies along with links to tools that facilitate their setup & maintenance. Finally, the document includes links that point to the dockerized versions of the respective tools, as provided by the BigDataEurope (BDE, <https://www.big-data-europe.eu/>) Project, which is starting point regarding the technical solutions that BigDataGrapes will built upon (as it has ben described in details in the BigDataGrapes DoA).

TABLE OF CONTENTS

| | |
|---|----|
| EXECUTIVE SUMMARY..... | 4 |
| 1. INTRODUCTION..... | 8 |
| 2. DATA INGESTION | 9 |
| 2.1 BATCH DATA | 10 |
| 2.2 STREAM DATA..... | 10 |
| 2.3 DATA SCRAPING AND WEB DATA..... | 12 |
| 3. DATA FUSION..... | 14 |
| 3.1 MESSAGING SYSTEM | 14 |
| 3.2 LONG TERM STORAGE | 16 |
| 4. DATA HARVESTING WORKFLOW | 17 |
| 4.1 DATA WORKFLOW PROCESS..... | 17 |
| 4.1.1Data Collection and optimization of scanning frequency in Big Data Platform..... | 17 |
| 4.1.2 Data Transformation | 17 |
| 4.1.3 Data Enrichment..... | 17 |
| 4.1.4Data Translation Service..... | 18 |
| 4.1.5 Data Curation | 18 |
| 4.2 MACHINE DATA ANNOTATION COMPONENT | 18 |
| 4.3 HUMAN DATA ANNOTATION COMPONENT..... | 19 |
| 4.4 DATA INTEGRATION TOOL..... | 19 |
| 5. DOCUMENTATION & TOOLS..... | 22 |
| 6. CONCLUSIONS..... | 24 |

LIST OF TABLES

| | |
|---|----|
| Table 1: Non-Functional Requirements satisfied by Flume..... | 12 |
| Table 2: Non-Functional Requirements satisfied by Kafka..... | 15 |
| Table 3: Non-Functional Requirements satisfied by Hbase & MongoDB | 16 |
| Table 4 Agroknow Semantic API..... | 18 |
| Table 5: Links documenting Flume tool | 22 |
| Table 6: Links documenting Kafka tool | 22 |
| Table 7: Links documenting HBase tool | 22 |
| Table 8: Links documenting MongoDB tool..... | 22 |

LIST OF FIGURES

| | |
|---|----|
| Figure 1. BigDataGrapes Architecture..... | 8 |
| Figure 2. BigDataGrapes Architecture Layers | 9 |
| Figure 3. Exemplary Flume data flow | 11 |
| Figure 4. Sequence Diagram between two Agents | 12 |
| Figure 5. Scrapy web crawler and transformation | 13 |
| Figure 6. Kafka Architecture | 14 |
| Figure 7 Data Curation Tool Interface | 19 |
| Figure 8 Data Integration Tool first page..... | 20 |
| Figure 9 Select Type of Data | 20 |
| Figure 10 Upload your file | 20 |
| Figure 11 Add some metadata | 21 |

1. INTRODUCTION

The BigDataGrapes platform aspires to provide components that go beyond the state-of-the-art on various stages of the management, processing, and usage of grapevine-related big data assets thus making it easier for grapevine-powered industries to take important business decisions. The platform employs the necessary components for carrying out rigorous analytics processes on complex and heterogeneous data helping companies and organizations in the sector to evolve methods, standards and processes based on insights extracted from their data.

For this purpose, the BigDataGrapes software stack has been designed and built using core technologies and frameworks for efficient processing of large datasets, such as Apache Spark and Apache Hadoop, making sure that the execution environment and methodology retain scalability and efficient use of the computational resources available. The distributed execution paradigm serves as the basis for efficiently solving the equally urgent challenge of Heterogeneity and Scalability in the context of Big Data processing.

In this first version of the deliverable we focus on the ingestion layer and the integration components. As is depicted in the next figure, the ingestion layer allows the BigDataGrapes software stack to communicate with the outside world, and more specifically it imports heterogenous data to be processed and analyzed by the next layers of the software stack. The integration components are crucial for ensuring the smooth interaction among the different layers of the software stack and ensure the sustainability of the software stack over time.

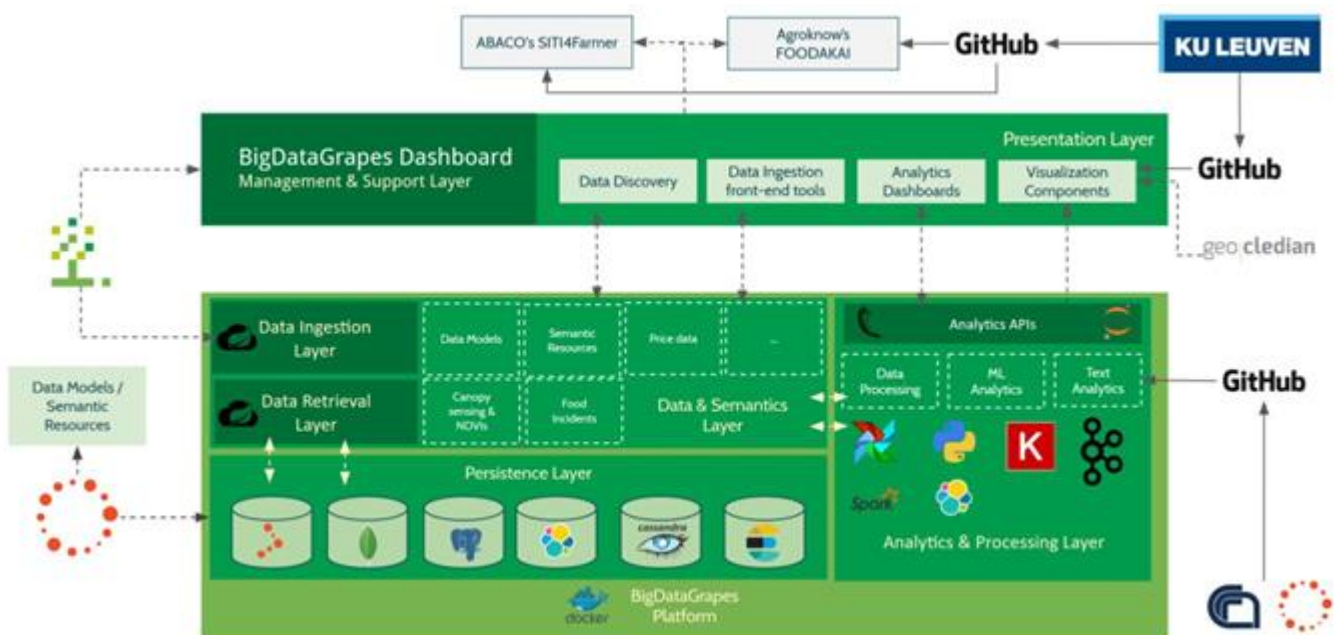


Figure 1. BigDataGrapes Architecture

The rest of this document is structured as follows: Section 2 highlights the proposed ingestion technologies used in the BDG software stack. Section 3 describes the integration components that will accommodate the interaction among the different components of the BDG. It also describes the long-term storage available options. Section 4 describes the data harvesting workflow, the human and the machine annotation component. Moreover, in this Section data integration and curation tools are described as well. Section 5 provides references to documentation and tools for each described technology, along with their dockerized version.

2. DATA INGESTION

Data ingestion is the first step for building data pipelines and also one of the toughest tasks in Big Data processing. Big Data Ingestion involves connecting to several data sources, extracting the data, and detecting the changed data. It concerns the data flow from their source to a system where they can be stored, processed and analysed. Furthermore, these several sources exist in different formats such as: Images, OLTP data from RDBMS, CSV and JSON files, etc. Therefore, a common challenge faced at this first phase is to ingest data at a reasonable speed and further process it efficiently so that data can be properly analysed to improve business decisions.

In the following figure, which depicts the logical structure of the BigDataGrapes system, the pivotal role of the Data Ingestion components and the need to select mature and/ or develop custom components so as to smoothly import data from sources of different nature is apparent.

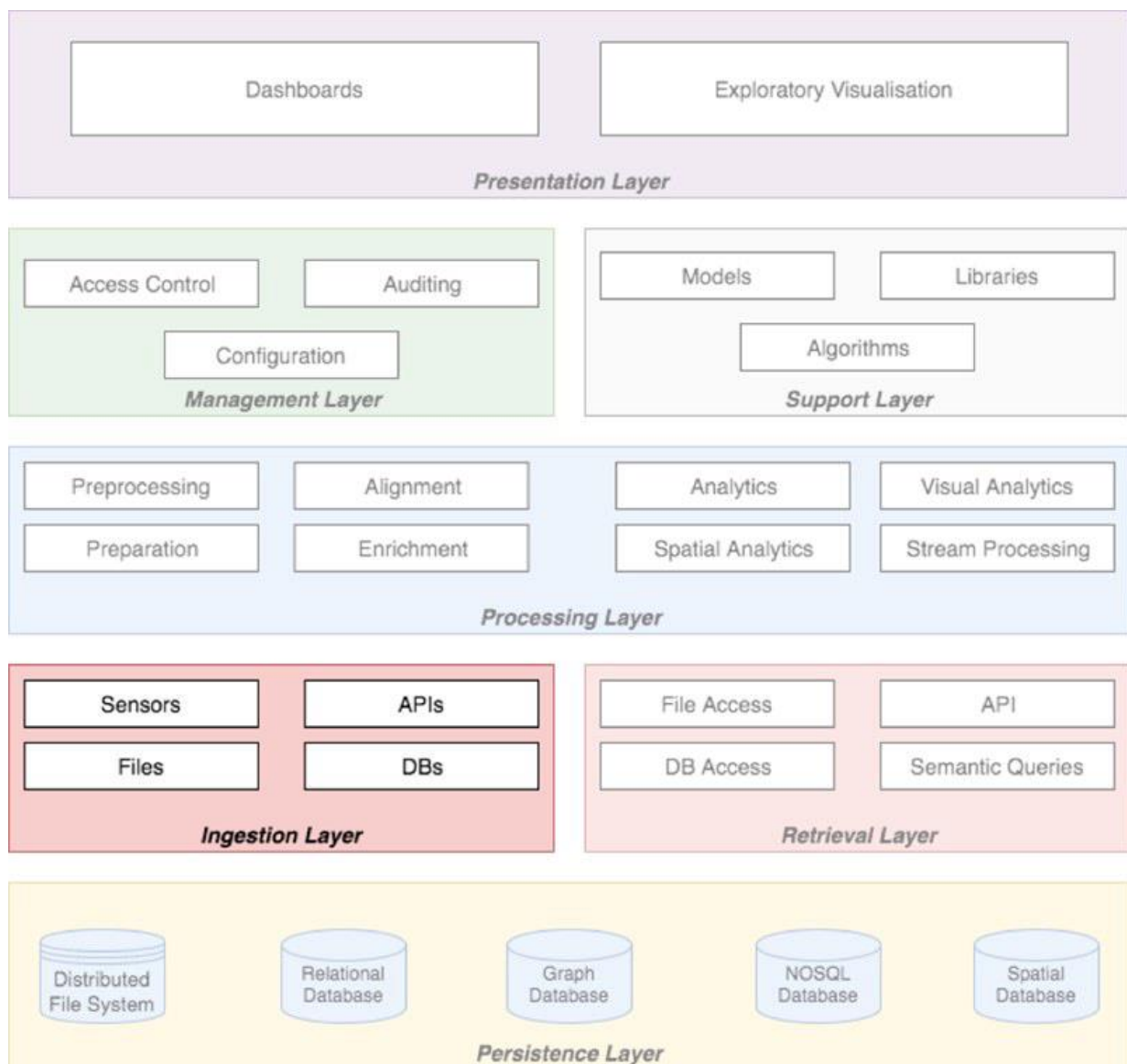


Figure 2. BigDataGrapes Architecture Layers

Data can be streamed in real time or ingested in batches. When data is ingested in real time it is ingested as soon as it arrives. When data is ingested in batches, data items are ingested in some chunks at a periodic interval

of time. Effective Data Ingestion processes begins by prioritizing data sources, validating individual files and routing data items to the correct destination.

2.1 BATCH DATA

Batch data ingestion means the pulling of data in discrete chunks at periodic intervals of time. Usually, batch data sources are RDBM's or filesystems that store the data in various format like CSV, JSON or even images. The sources can be diverse in terms of the nature of the data or their formats.

Therefore, the ingestion component should be able to connect to multiple sources (in meaningful intervals) and pull the data regardless of their size. Moreover, the ingestion component should perform basic sanitization tasks, like deduplication and model validation, to ensure the integrity of the data that are ingested to the overall system.

The diverse nature of batch data sources makes the development of custom ingestion components necessary, that will be customizable to satisfy the different requirements that different sources pose.

Within the context of this project such a case is the processing and integration of the laboratory monitoring results performed by countries worldwide. Such results involve the testing for pesticide residues as well as veterinary drugs residues in agricultural commodities and animal products. The monitoring programs involve millions of data records making their integration in the data platform a challenge with respect to speed and performance. To tackle this problem parallel processing has been employed. More specifically a python script has been created that initially identifies the number of cores and threads available to the server in which it is executed. After that, multiple processes are initiated, each tasked with a specific country's results. Within this script and prior to the initial ingestion of the data, deduplication and integrity checks are performed so as to ensure the data quality necessary for future work.

2.2 STREAM DATA

The main difference between batch and stream data is the fact that stream data are continuously generated by multiple sources and are emitted as records in small sizes. Stream data are processed sequentially and incrementally.

Therefore, the ingestion component should guarantee data handling, in the sense that an efficient strategy for data loss or late arrived data should be applied. Also, contrary to the batch data case, in stream data the sources push data to the system, therefore, the ingestion component should be able to handle efficiently simultaneous requests from different sources.

Apache Flume¹ is a tool which has been designed specifically for ingesting stream data. Flume is distributed in nature, and its flexible architecture makes it a robust solution. Also, Flume provides a tunable fault-tolerant mechanism that can be customized to satisfy the different requirements of different sources. Its distributed nature encapsulates a variety of failover and recovery mechanisms.

The following concepts summarize the core characteristics of Flume:

- Event: The unit of data that is transported through Flume starting from the data source to its final destination.

¹ <https://flume.apache.org/>

- **Agent:** A process that implements the different flume components. An agent can receive, store and send events to the next destination.
- **Client:** An interface located between the data source and Flume. The data source pushes the data through that interface, in order to be delivered to a Flume agent.
- **Channel:** In the context of Flume, a channel is a transient storage component for events. An event stays in the channel until a sink removes it. Such a channel could be a messaging system, a schema-less database or even an RDBMS.
- **Sink:** Sinks remove events from channels and drains them to other channels, agents or to its final destination.

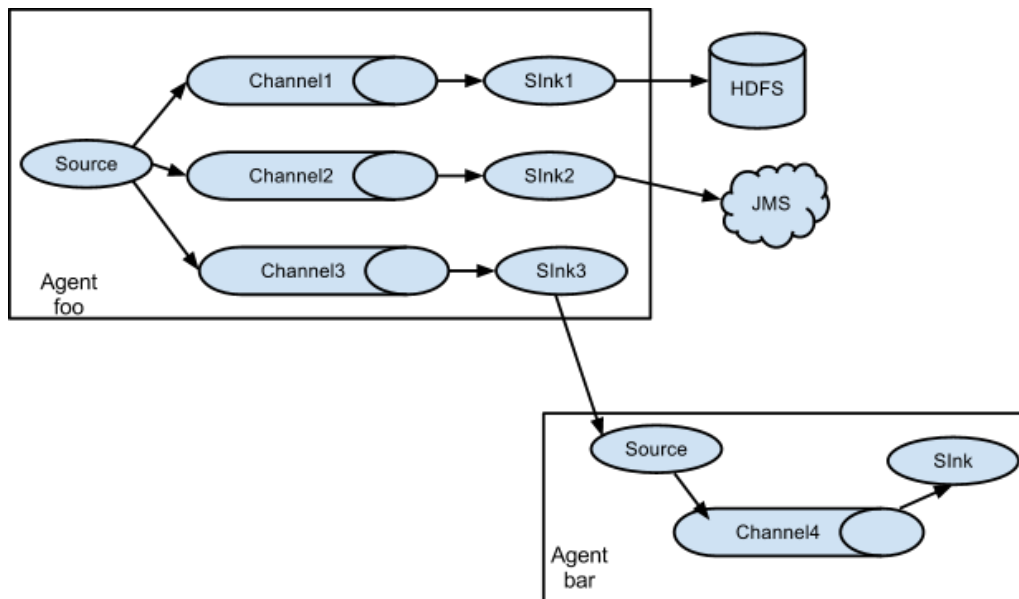


Figure 3. Exemplary Flume data flow
(source: <https://flume.apache.org/FlumeUserGuide.html>, Sep 2018)

Flume implements a channel-based transaction protocol to ensure event delivery. Whenever an event is about to move from one agent to another, both agents initiate a transaction. The sending agent initiates first its transaction and sends the event to the receiving agent.

Afterwards, the receiving agent initiates its transaction, receives the event, applies its logic to the event, puts it to the next channel and commits the transaction.

Upon committing its transaction, the receiving agent sends a success indication to the sending agent, which in turn commits its transaction. The next figure depicts the communication flow between the two agents.

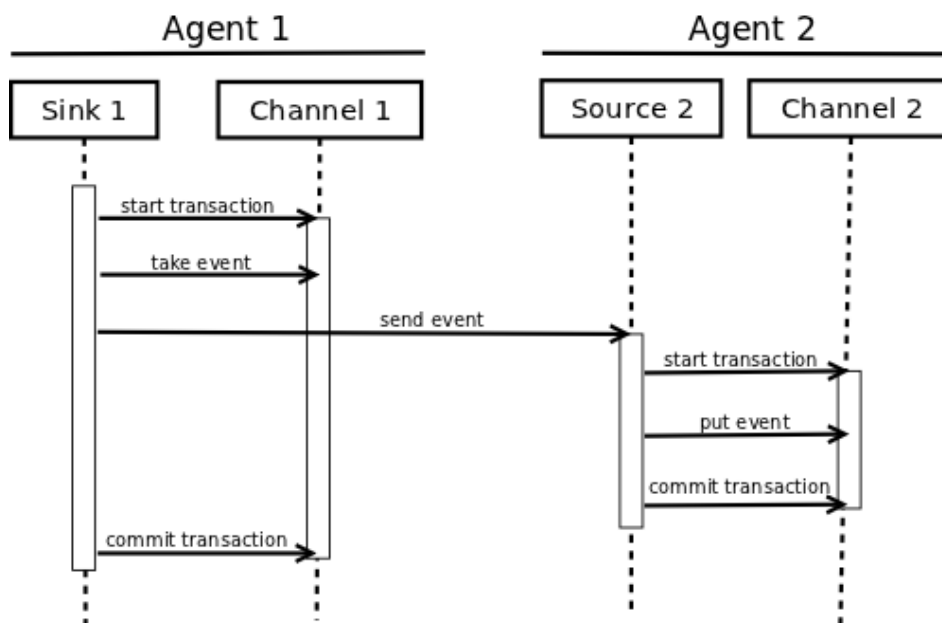


Figure 4. Sequence Diagram between two Agents
(source: https://blogs.apache.org/flume/entry/flume_ng_architecture, Sep. 2018)

When an event hops through many agents, and a communication error is encountered, the next events start getting buffered to the last unaffected agent. This gives enough time for the communication failure to be resolved. In the case that the error is not resolved, the first agent will report to the client, which will allow it to take appropriate action. If the error is resolved in time, the buffered event will start to flow again through the agents.

The following table summarizes the non-functional requirements defined in the deliverable D2.3 BigDataGrapes Software Stack Design and whether Flume satisfies them:

Table 1: Non-Functional Requirements satisfied by Flume

| Requirement | Satisfied by Flume | How |
|--|--------------------|------------------------------------|
| STREAMLINED, EFFICIENT DEPLOYMENT STRATEGY | Yes | Dockerized |
| DEVELOPMENT | Yes | Highly modular |
| FAULT TOLERANCE | Yes | Channel-based transaction protocol |
| SCALABILITY | Yes | Distributed by nature |
| OPEN SOURCE | Yes | - |

2.3 DATA SCRAPING AND WEB DATA

Data scraping², also known as web scraping, is the process of importing information from a website into a spreadsheet or local file saved on your computer. It's one of the most efficient ways to get data from the web, and in some cases to channel that data to another website. In the food protection pilot, many different data sources and data types have been used like textual information and numerical data. Textual information

² <https://www.targetinternet.com/what-is-data-scraping-and-how-can-you-use-it/>

includes mainly announcements about food recalls and border and the main source of our datasets is the open data published by the governments. For instance, scrapy used for web crawling. Scrapy is a fast high-level web crawling and web scraping framework used to crawl websites and extract structured data from their pages. It can be used for a wide range of purposes, from data mining to monitoring and automated testing. First, a scrapy web crawler (figure 5) is developed that crawls HTML pages and transforms them into a JSON file. A directory with all jsons sends to a Mongo database through a JsonToMongo project. Data from Mongo database transform to an XML format using MongoExporter project. This project includes Text classification, the most important information is to extract the Date. The date is the most important field for the Food Protection pilot. Next, the XML file translated and enriched using translation and increment components. Finally, the data is uploaded to the elastic index and is ready to use.

```
import scrapy
import requests
from datetime import datetime
import json

class BrickSetSpider(scrapy.Spider):
    name = "brickset_spider"
    start_urls = ['https://www.efet.gr/index.php/el/enlnerost/deltia-typou/anakletsets-cat']
    def tojson(self, response):
        print(response.meta.get('url'))
        json_antigonl={
            "url": response.meta.get('url'),
            "belongs": "efet",
            "accessed": int(datetime.timestamp(datetime.now())),
            "html_size": 173468,
            "text_size": 173468,
            "metatags": "",
            "html": response.text,
            "text": response.text
        }
        print(json)
        with open("jsons/"+str(hash(response.meta.get('url')))+'.json', 'w') as j:
            json.dump(json_antigonl, j)
    def parse(self, response):
        SET_SELECTOR = '.catItemTitle'
        for brickset in response.css(SET_SELECTOR):
            NAME_SELECTOR = 'a::attr(href)'
            url="https://www.efet.gr" + str(brickset.css(NAME_SELECTOR).extract_first())
            yield scrapy.Request(
                response.urljoin(url),
                callback=self.tojson, meta={'url': url}
            )
```

Figure 5.Scrapy web crawler and transformation

3. DATA FUSION

3.1 MESSAGING SYSTEM

The ingested data, either batch or stream, should be distributed all over the system by a simple, effective and reliable messaging system. The most popular systems available are RabbitMQ³ and Apache Kafka⁴.

The main difference between the two systems is that Kafka has a dumb broker and relies to the message consumers to implement the desired logic, while RabbitMQ uses a smart broker that communicates the messages to the consumers. This different design choice makes Kafka an interesting candidate in a system that ingest batch and stream data.

Kafka is a messaging framework, that is distributed in nature and runs as a cluster in multiple servers across multiple datacenters. Moreover, Kafka allows the real-time subscription and data publishing of large numbers of systems or applications.

This allows streamlined development and continuous integration facilitating the development of applications that handle either batch or stream data.

An important factor in data ingestion technology, especially when handling data streams, is the fault tolerance capability of the chosen technology. Kafka ensures the minimization of data loss through the implementation of the Leader/Follower concurrency architectural pattern⁵.

This approach allows a Kafka cluster to provide advanced fault tolerant capability, which is a mandatory requirement for streaming data applications.

The following figure depicts the general architecture of Kafka:

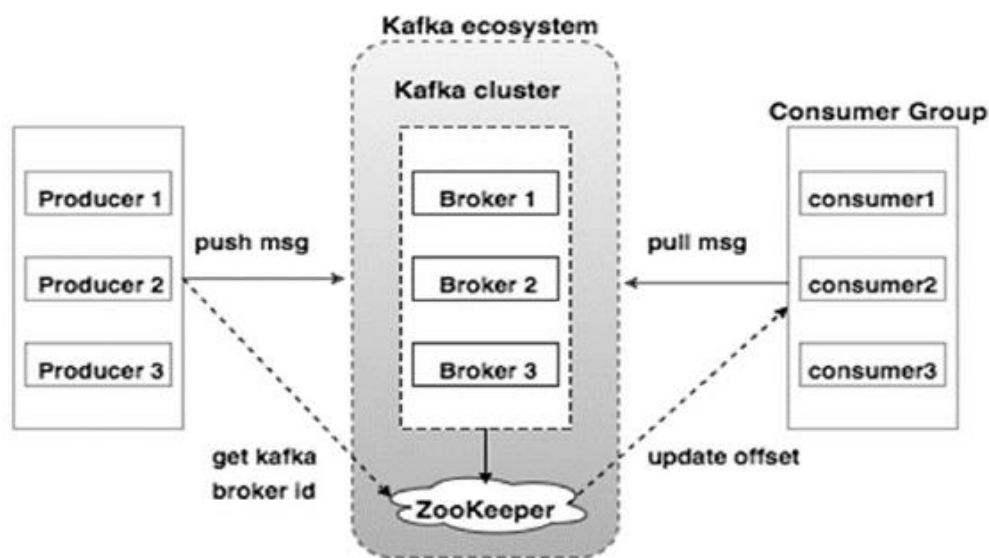


Figure 6. Kafka Architecture

(https://www.tutorialspoint.com/apache_kafka/images/cluster_architecture.jpg, Sep 2018)

The core of Apache Kafka consists of the following four major components:

³ <https://www.rabbitmq.com/>

⁴ <https://kafka.apache.org/>

⁵ C. D. Schmidt, M. Stal, H. Rohnert and F. Buschmann, Pattern-Oriented Software Architecture, Volume 2, Patterns for Concurrent and Networked Objects, Wiley, 2000.u

- **Producer API:** Allows applications to communicate with the Kafka cluster and send data in the form of messages
- **Consumer API:** Allows application to read data in the form of messages from the Kafka cluster
- **Connect API:** This component has an integral role for streaming systems, because it allows the creation of pipelines that continuously send or receive data from the Kafka cluster
- **Streams API:** This component offers high-level operators like filters, maps, grouping, windowing, aggregation, and joins. It also allows the development of low-level custom operators.

These core components not only ensure that Kafka is a high-performance messaging framework, in terms of read/write cost⁶, but also ensures the scalability of the framework, allowing near-to-real-time processing, regardless of the volume of the data^{7 8}.

Kafka is an open source project, that has vast support from the community by providing extensions that facilitate the connection of Kafka with other BigData technologies like the Hadoop Distributed File System (HDFS), the Apache Flink and the Elasticsearch.

Also, rich documentation is available for both batch and stream data facilitating the development of an ecosystem of applications. Also, a Kafka cluster can be dockerized making easier its distribution and deployment.

The following table summarizes the non-functional requirements defined in the deliverable D2.3 BigDataGrapes Software Stack Design and whether Kafka satisfies them:

Table 2: Non-Functional Requirements satisfied by Kafka

| Requirement | Satisfied by Kafka | How |
|--|--------------------|--|
| STREAMLINED, EFFICIENT DEPLOYMENT STRATEGY | Yes | Can be dockerized |
| DEVELOPMENT | Yes | Highly modular |
| FAULT TOLERANCE | Yes | Implements the Leader/Follower pattern |
| SCALABILITY | Yes | Distributed by nature |
| OPEN SOURCE | Yes | - |

⁶ <https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines>

⁷ <https://insidebigdata.com/2018/04/19/developing-deeper-understanding-apache-kafka-architecture-part-2-write-read-scalability/>

⁸ <https://blog.cloudera.com/blog/2018/05/scalability-of-kafka-messaging-using-consumer-groups/>

3.2 LONG TERM STORAGE

It is critical to store the ingested data, either batch or stream, to be available for further processing and analysis. It is also important to select a solution that does not pose processing overheads either when storing the data or retrieving them. Therefore, to minimize the storing and retrieving complexity the use of schema-less database technology helps.

Such a database does not conform to a schema; thus, it does not require any modelling of the data, which in case of multiple sources would be a large burden. Also, a schema-less database can store data with different structure without the need to design a grant common schema or migrating between schemas. Popular schema-less databases are Apache HBase⁹ and MongoDB¹⁰.

HBase is a distributed database which follows the Bigtable technology of Google. HBase implements a fault-tolerant method for storing large quantities of sparse data, in the sense that the useful information is a small amount within a large collection of data. Also, HBase has a natural connection with Hadoop, hence can connect with MapReduce processes.

MongoDB is also a distributed database which treats and stores data as JSON documents. Thus, data can have different fields and the data structure is essentially alive since it can be changed over time. Also, MongoDB provides ad-hoc queries, supporting field query, range query and regular expression searches. Moreover, MongoDB has fault-tolerant and load balancing capabilities by providing replication and sharing of the main database.

Table 3: Non-Functional Requirements satisfied by Hbase & MongoDB

| Requirement | Satisfied by Hbase & MongoDB | How |
|--|------------------------------|-----------------------|
| STREAMLINED, EFFICIENT DEPLOYMENT STRATEGY | Yes | Can be dockerized |
| DEVELOPMENT | Yes | Highly modular |
| FAULT TOLERANCE | Yes | Replication/ Sharding |
| SCALABILITY | Yes | Distributed by nature |
| OPEN SOURCE | Yes | - |

⁹ <https://hbase.apache.org/>

¹⁰ <https://www.mongodb.com/>

4. DATA HARVESTING WORKFLOW

4.1 DATA WORKFLOW PROCESS

Data workflow is a process where a small script, also known as a malicious bot, is used to automatically extract large amount of data from websites and use it for other purposes. It is usually interchangeable with web scraping, web crawling, and data extraction. Data harvesting is the process to extract valuable data out of target websites and put them into your database in a structured format. To conduct data harvesting, you need to have an automated crawler to parse the target websites, capture valuable information, extract the data and finally export into a structured format for further analysis. Data harvesting, therefore, doesn't involve algorithms, machine learning, nor statistics. Instead, it relies on computer programming like Python, R, Java, to function. Besides, data harvesting is more about being accuracy.

The general functions of the data harvesting that can achieve this are:

1. Collect
2. Transform
3. Enrich
4. Curate

4.1.1 Data Collection and optimization of scanning frequency in Big Data Platform

The data platform uses various cron jobs and web crawlers for frequency scanning of data sources. On the one hand, croni is a Linux utility which schedules a command or script on your server to run automatically at a specified time and date. A cron job is the scheduled task itself. Cron jobs can be very useful to automate repetitive tasks. On the other hand, the Web crawlerii, or spider or spiderbot or crawler, is an Internet bot that systematically browses the World Wide Web, typically for the purpose of Web indexing. In the case of the food safety pilot, some cron jobs have been created on the data platform and they are responsible for the frequent execution of web crawlers. These crawlers programs systematically browses the food safety authorities for the purposes of data collection and indexing. Through various experiments, the time between the web crawling of these sources from 3 days was devoted to 2 hours and parallel processing. As a result, the data collected is synchronized.

4.1.2 Data Transformation

Data Transformation is used to bring all collected data under the umbrella of a common schema. The first step towards that end is the cleaning of data records that are irrelevant to the identified thematic scope or generally malformed records that cannot be interpreted or used by the machine. The next step is the harmonization of records, in the sense that their underlying values are normalized to follow the same conventions (date fields transformed to follow a specific date format, numerical fields to use the same decimal separator etc).

4.1.3 Data Enrichment

Data Enrichment is used to generate richer or previously unidentified metadata descriptors for data assets. These descriptors are drawn from the semantic resources that were identified and engineered by the engaged communities. To achieve linking of data to the semantic resources, entity recognition routines are required in the textual content of each data asset and along with its identified context linking to the appropriate semantic resource concept or class.

4.1.4 Data Translation Service

For these step POS Tagger Service is used. In particular, these step words are distinguished based on whether they are verbs, objects, intentions, etc. The text data comes from food safety authorities worldwide, which is why the data are in different languages and must be translated to be harmonized. That's why the Translate API is used. The API ¹¹provides access to the Yandex online machine translation service. It supports more than 90 languages and can translate separate words or complete texts.

4.1.5 Data Curation

Data Curation allows human curators to review, organize and enrich data manually. Although optional and time-consuming, this step ensures the quality of data. The tools used in this step consist of an intermediate storage component, the machine curator and a user interface that enables the click-through the different data records. After this step, data is forwarded to the Indexing component of the platform that is responsible for exposing it to consumer applications and users in a performance-optimized machine-readable format. The curation user interface results help the machine curator to learn how to recognize patterns to enhance automatic enrichment

4.2 MACHINE DATA ANNOTATION COMPONENT

Data Annotation service aims at identifying hierarchical relationships that data objects may have with the domain or scope of activities that they reference. Typically, every demonstrator use-case handles data that is largely untagged with descriptors that puts it in context of a wider domain. For instance, a data asset can simply be a CSV file. With the appropriate linking routine, said CSV file can be labelled as a food safety resource, a phenotyping experiment record, or a simulation algorithm output file, ready to be reused in the other workflows.

The Agroknow Semantic API is responsible for delivering textual analysis and semantic link suggestions in response to user input. If chained correctly, NLP services can be used in conjunction with the Search API to produce links to semantic resources harvested by BigDataGrapes's Data Platform. Otherwise, users may invoke the "annotate" service directly to produce links to any or a specific vocabulary harvested by the platform.

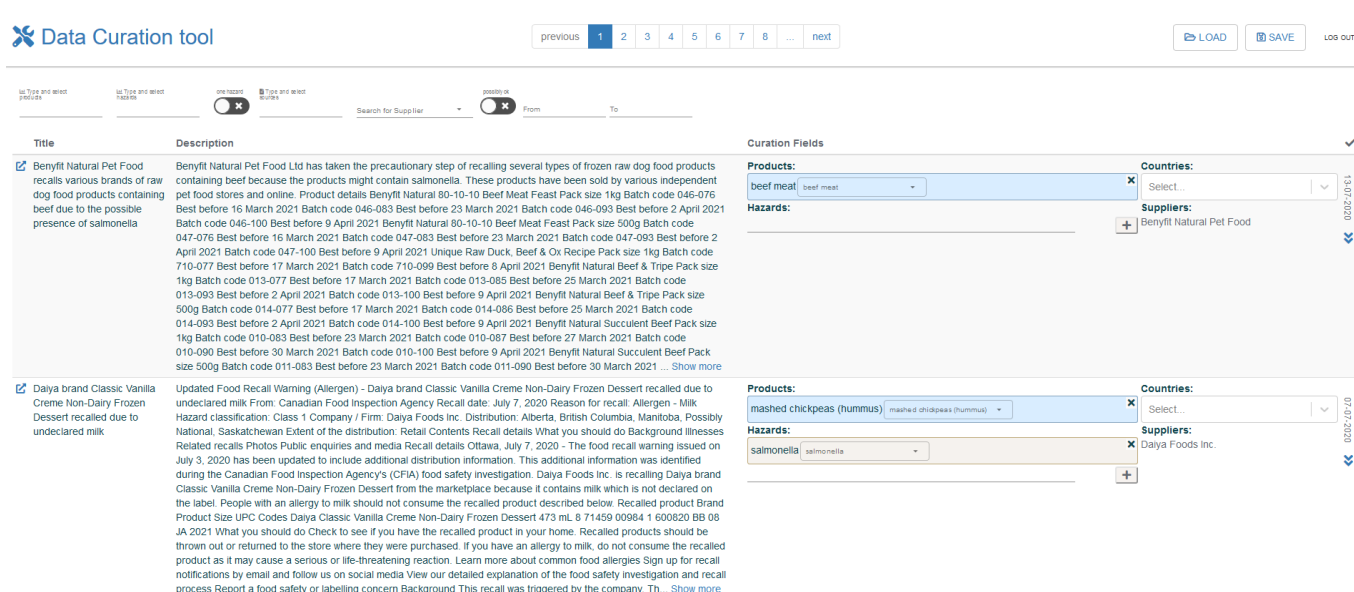
Table 4 Agroknow Semantic API

| Semantic API | | |
|-------------------|--|---|
| Service endpoints | Inputs | Output |
| Ngrams[POST] | [String] text to analyze, [Integer] size of output n-grams | [JSON Array] a list of n-grams, i.e. a contiguous sequence of String items, that can be phonemes, syllables, letters or words |
| Stopwords [POST] | [String] text to analyze | [String] text tokens without common words, such as "a", "to" etc |
| Tag [POST] | Yes - | [JSON Array] part-of-speech tags |
| Annotate [POST] | Yes | [JSON Array] list of vocabulary links |

¹¹ <https://tech.yandex.com/translate/>

4.3 HUMAN DATA ANNOTATION COMPONENT

Data Curation tool is a web interface that on the one hand, helps curators to annotate data and on the other hand train the algorithms to the automatic annotation. In figure 6 the Data Curation Tool is presented. On the top of the page, there are some filters that affect the data coming from the data platform. The curator can filter based on the desired hazard (i.e. reason behind a recall), the product or ingredient involved, the data source from which the record was collected, the date of the announcement and whether the records available are tagged as “possibly ok”. The last option returns data that is annotated by algorithms with low accuracy and needs validation. When a curator finds a product in the title or in the description, selects it and then clicks to add to products. Then the product appears to curation fields and must be mapped with the product hierarchy. The same procedure is followed for hazards. When a curator selects a product or a hazard, the position is kept and helps algorithm training. When the curation is ready, selects ok and a request with the ready data is sent to the data platform.



The screenshot shows the 'Data Curation tool' interface. At the top, there are navigation buttons: 'previous', '1', '2', '3', '4', '5', '6', '7', '8', and 'next'. On the right, there are 'LOAD', 'SAVE', and 'LOG OUT' buttons. Below the navigation bar, there are tabs for 'all Type and select records', 'all Type and select records', 'one record', and 'all Type and select records'. A 'Search for Supplier' field is present. The main content area is divided into two columns. The left column lists food recall records with details like 'Title', 'Description', and 'Show more'. The right column, titled 'Curation Fields', shows the details for two selected records. The first record is 'Benyfit Natural Pet Food' and the second is 'Daiya brand Classic Vanilla Creme Non-Dairy Frozen Dessert'. For each record, there are fields for 'Products', 'Hazards', 'Countries', and 'Suppliers'. The 'Products' field has a dropdown menu with 'beef meat' and 'mashed chickpeas (hummus)'. The 'Hazards' field has a dropdown menu with 'salmonella'. The 'Countries' field has a dropdown menu with 'Select...'. The 'Suppliers' field has a dropdown menu with 'Benyfit Natural Pet Food' and 'Daiya Foods Inc.'.

Figure 7 Data Curation Tool Interface

4.4 DATA INTEGRATION TOOL

Data Integration tool is a web interface that helps data providers to upload their data to data platform. Using the Data Integration Tool, you can transform your data in a way that can be used from our services. You can start by clicking the [Add New] button (figure 8). You can also check or remove previously uploaded data in the table shown below.

DATA INTEGRATION TOOL

Using the Data Integration Tool, you can transform your data in a way that can be used from our services. You can start by clicking the [Add New] button. You can also check or remove previously uploaded data in the table shown below. For any questions regarding the usage of the tool please visit the website www.test.test or contact us.

Add New


Figure 8 Data Integration Tool first page

The next page includes a selection page which the user chooses the type of data you want to upload. There are two options: file from computer or stream from an endpoint (figure 9).

● Select Type of Data — ● Upload Data — ● Add Metadata — ● Finished


WHAT WOULD YOU LIKE TO IMPORT?

Starting, please choose the type of data you want to upload. Is it a data file or a stream like a sensor API?



FILE FROM COMPUTER

Upload any CSV, XLS, or JSON files with entity information.



STREAM FROM AN ENDPOINT

Import a URL pointing to a data stream (IoT sensors, data APIs etc.)

Figure 9 Select Type of Data

The next page includes the upload data (figure 10). In this page, the user chooses the file they want to upload. All .csv, .xlsx, and .xls file types are supported.


● Select Type of Data — ● Upload Data — ● Add Metadata — ● Finished

UPLOAD YOUR FILE

Before you upload your file, please make sure it is ready to be imported.

Drag & Drop Files

Drag n Drop or



Choose a File

to upload to the Data Platform.
All .csv, .xlsx, and .xls file types are supported.

Figure 10 Upload your file

In the next step, the user maps their file columns to the expected schema elements, according to the Data Integration API specification. Moreover, the user proceeds to fill-in missing or erroneous values in an online spreadsheet-like interface. The user inputs some metadata that accompany the file that they want to publish and optionally, select the data asset that they want to associate the file with (figure 11). In the last step, the file is published in a target repository.

✓ Select Type of Data ———— ✓ Upload Data ———— 1 Add Metadata ———— 2 Finished

ADD SOME FINAL USEFUL INFORMATION

Adding some final information about your dataset, will help us manage it better.

title

pilot

data license

type

tags

description

Figure 11 Add some metadata

5. DOCUMENTATION & TOOLS

The following tables contains links that point to the official documentation of the aforementioned tools and also links that point to open source tools that facilitate their configuration and monitoring.

Table 5: Links documenting Flume tool

| Flume | | |
|--------------------------|-------------|---|
| Documentation/ Tool | Description | Link |
| Official Documentation | - | https://flume.apache.org/documentation.html |
| Monitoring Documentation | - | http://www.bigdatareflections.net/blog/?p=83 |
| Docker | - | https://github.com/big-data-europe/docker-flume |

Table 6: Links documenting Kafka tool

| Kafka | | |
|------------------------|--------------------------|---|
| Documentation/ Tool | Description | Link |
| Official Documentation | - | https://kafka.apache.org/documentation/ |
| ZooKeeper | Distributed Coordination | https://zookeeper.apache.org/doc/r3.4.13/ |
| Prometheus | Monitoring/ Alerting | https://prometheus.io/ |
| Landoop | Management | https://www.landoop.com/kafka/kafka-tools/ |
| kafka-benchmark | Benchmarking Tools | https://github.com/fede1024/kafka-benchmark |
| Docker | - | https://github.com/big-data-europe/docker-kafka |

Table 7: Links documenting HBase tool

| HBase | | |
|------------------------|----------------------------|---|
| Documentation/ Tool | Description | Link |
| Official Documentation | - | https://hbase.apache.org/ |
| HBase Explore | Management/ Administration | https://sourceforge.net/projects/hbaseexplorer/ |
| Docker | - | https://github.com/big-data-europe/docker-hbase |

Table 8: Links documenting MongoDB tool

| MongoDB | | |
|------------------------|-------------|---|
| Documentation/ Tool | Description | Link |
| Official Documentation | - | https://docs.mongodb.com/ |

| | | |
|---------------|-------------------------------|---|
| Studio 3T | Management/ Administration | https://robomongo.org/ |
| MongoDB Tools | Suite of MongoDB utilities | https://github.com/mongodb/mongo-tools |
| Docker | - | https://hub.docker.com/_/mongo/ |

6. CONCLUSIONS

This accompanying document for deliverable D3.2 Data Ingestion & Integration Components describes the mechanisms and tools that will be used in the BigDataGrapes platform to ingest data of different nature from multiple sources. The document describes the tools that will be used for data integration across the different BigDataGrapes platform layers, as well as for long-term storage and preservation of data.

The ingestion layer as well as the integration components are key factors of the smooth development and the sustainability of the platform over time. The presented technologies ensure that both batch & stream data will be imported effectively to the platform, and that the components from different layers of the platform will be able to smoothly intercommunicate.

ⁱ <https://www.hostgator.com/help/article/what-are-cron-jobs>

ⁱⁱ https://en.wikipedia.org/wiki/Web_crawler