FADI - A Deployment Framework for Big Data Management and Analytics

Rami Sellami, Faiez Zalila, Alexandre Nuttinck, Sbastien Dupont, Jean-Christophe Deprez, Stphane Mouton
CETIC - Centre d'Excellence en Technologies de l'Information et de la Communication
Charleroi, Belgium
first.lastname@cetic.be

***Abstract*—The production of huge amount of data and the emergence of new technologies in the industry sector have introduced new requirements for big data management. Many applications need to interact with several heterogeneous data sources to ingest, harmonise (normalise), persist, analyse and synthesize results to enable informed decisions and draw benefits from data. These operations are ensured by different tools and these tools are heterogeneous and not connected with each other. Besides, the whole tool-chain lacks automation in terms of its deployment, its operational workflow and its orchestration for satisfying the elastic and resilient properties needed by Industry. In this paper, we present FADI, a framework for deploying and orchestrating a Big Data management and analysis platform fully composed of open source tools. FADI has been developed through several research projects, namely, BigData@MA, Grinding 4.0, Quality 4.0 and ARTEMTEC where Industry use cases are used for validation purposes.**

***Index Terms*—Big Data, Deployment, DevOps, Kubernetes (K8s), Helm, Open Source**

I. INTRODUCTION

From the exponential growth in the volume of data collected by IT systems arises new challenges in terms of data analytic and exploitation. This situation forces researchers to find new ways to manage and process data, notably, for discovering new methods for capturing, searching, sharing, storing, analyzing and presenting data hence the emergence of "Big Data".

According to the NIST Big Data Public Working Group [1], Big Data is data which exceeds the capacity or capability of current or conventional methods and systems. In [2], IBM defines the term Big Data as the information that cannot be processed or analyzed using traditional processes or tools. The Big Data problematic is associated to the 3-Vs: volume, velocity and variety. Volume denotes the processing of large amounts of data. Velocity signifies the increasing rate at which data flows and must be processed [2]. Variety refers to the diversity of data sources (e.g. IoT devices, web applications, etc.) and data structures (Web sockets, artifacts and media). With the emergence of new technologies (e.g. Cloud Computing, sensors, smart devices, social networks, etc.), Data-sets have grown in complexity since they are a combination of highly heterogeneous data (e.g. type, structure, format, etc.).

For IT industry, one of the most important challenges is how to process and exploit the collected data [3]. Efficient data processing has translated into competitive business advantages. Stakeholders such as data scientists/engineers, and business analysts should easily collect data, analyze it and publish results. To efficiently solve the 3Vs from Big Data problematic, a well-crafted platform must be put in place. Existing solutions exhibit limitations in terms of cost (expensive) or flexibility and extensibility (vendor locking).

In this paper, we present FADI[1], a generic framework for deploying and orchestrating a Big Data platform to collect, analyze and visualize data. FADI is a containerized cloud-native platform for Big Data based on mature open source tools. It is defined on top of Kubernetes container orchestrator. Therefore, it is a Cloud agnostic and portable solution.

The paper is structured as follows. First, Section II recalls the main requirements for building a Big Data platform. Section III presents the Big Data platform and its different components deployed using FADI . Sections IV and V illustrate different Industry use-cases and validates the different requirements satisfied by the FADI deployment framework. Section VI presents the related work. Section VII concludes and presents our future works.

II. MOTIVATION & REQUIREMENTS

According to the ISO/IEC 25010 System and software quality models [4], we have identified three levels of requirements that should be considered when designing a Big Data platform, how to deploy it and how orchestrate its operations.

First of all, the setup level concerns the requirements related to the ease of the platform deployment (**R1-Ease of Deployment**). IT administrators expect a flexible and simple means for deploying a Big Data platform anywhere (i.e. on-premises, hybrid, public cloud infrastructures). It should also be extensible, interoperable and portable. Swapping or adding new software tools or services to fulfill new requirements shall be possible, simple and quickly done (**R1.1 Flexible Deployment**). It shall be possible to migrate Cloud providers and to avoid Cloud vendor lock-in (**R1.2 portable**). The deployment procedure must also integrate the resulting Big Data platform in an overall existing architecture and infrastructure with its specific environment such as tools, internally developed frameworks, IoT devices and local databases (**R1.3 Interoperability**). Second, at the operations level a first major requirement relates to the handling of workload variation notably through an appropriate elastic behavior, which is the

[1]https://www.cetic.be/FADI?lang=en

ability to grow or shrink infrastructure resources dynamically to adapt automatically to the workload changes (**R2-Automatic Elastic Orchestration**: **R2.1 Scaling up**, **R2.2 Scaling Down**). Finally, on the usability level, it is important to fulfill the user experience requirements also for other users than IT infrastructure administrators. In fact, Big Data platform end users (e.g. data engineers, data scientists, business analysts) must be able to adapt and tune a Big Data platform on their own without constantly requiring support from IT infrastructure administrators (**R3-Usability**). Data engineers must be able to create the necessary ingestion and bridges between various data sources and types (**R3.1 Usability for Data Engineers**). Data scientists must be provided with means to develop and package their data analytic and machine learning algorithms (**R3.1 Usability for Data Scientists**). Business Analysts shall have the means to define dashboards to easily and quickly interpret the information resulting from data analysis (**R3.2 Usability for Business Analysts**).

III. FADI: A CUSTOMIZABLE END-TO-END BIG DATA PLATFORM

FADI is a customizable end-to-end big data platform enabling the deployment and the integration of open source tools in a portable and scalable way. It is a multi-tenant and multi-actors (i.e. business analyst, data scientist/engineer, IT admin, etc.) platform. The main features of FADI are five-fold (see Figure 1): (1) collecting batch and stream data coming from various data sources, (2) storing data in different types of data stores, (3) processing data using ML and Artificial Intelligence (AI) techniques, (4) visualizing and analyzing data in a user Web interface, and (5) generate and publishing reports. In the rest of this section, we present an overview of the software architecture of FADI (see Section III-A), a sample implementation of it (see Section III-B) and the DevOps features and its Continuous Integration/Continuous Delivery (CI/CD) pipeline (see Section III-C).

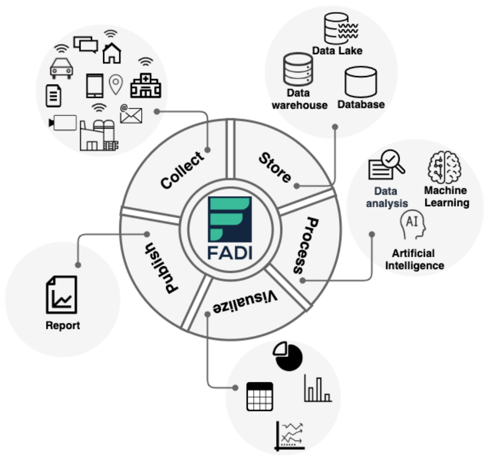

Fig. 1. An overview of FADI's features

A. Overview of the software architecture of FADI

In Figure 2, we showcase an overview about FADI. The figure is divided in five main layers that are:

- **The internal administration services** layer enables to manage and maintain FADI. It includes the following components: The **logs management** component allows logs collecting, storing and making them available to FADI administrators. The **backup management** component enables to manage the database backups by integrating a data recovery mechanism in case of hardware or software failures. The **monitoring** component is used to control the status of the other components and to provide information on the state of the system (e.g. memory, CPU, error logs, etc.). The **user management** component allows to control users identity and access to the different components of FADI and to the data.
- **The data integration** layer is responsible for collecting, ingesting, transforming and routing data. Data may be collected via stream or batch. This layer includes three main components. First, the **data transport** component is a message queue system enabling to control and manage the data collection. Second, the **data source integration** component enables to collect data coming from heterogeneous data stores and to store it in a single one. Third, the **data flow management** component collects, validates and formats the raw payload objects.
- **The data analysis** layer is responsible for analyzing stream and batch data. It includes three main components. The **broker** component enables to collect and distribute data in a scalable and fault tolerant way. It supports raw data coming from the data ingestion tool and the validated (pre-processed) data dedicated to the stream processes. Then, the **pre-processing** component manages the operations related to the Extract Transform and Load (ETL) processes in order to prepare data to be analyzed and processed by the **Stream Processing** component. This latter processes data using ML and AI libraries, or custom processing algorithms to give meaning to it.
- **The data storage** layer supports the storage of various and heterogeneous kinds of data. It includes three types of data stores. The **Data Lake** is used to store raw ingested data in order to gather all the data into one system to easily establish links between the different kinds of data. The **Data Warehouse** is dedicated to store processed data and is linked to the data analysis components. The **Model Repository** contains models like ML models, etc.
- **The serving layer** has two main tasks. First, it allows to present data to the users via a reporting user interface (e.g. notebooks, dashboards, etc.) in order to (1) define and build data analysis models, (2) exploit data, and (3) visualize data via dashboards and reports. Second, it enables the FADI administrators to maintain and manage the FADI components and infrastructure.

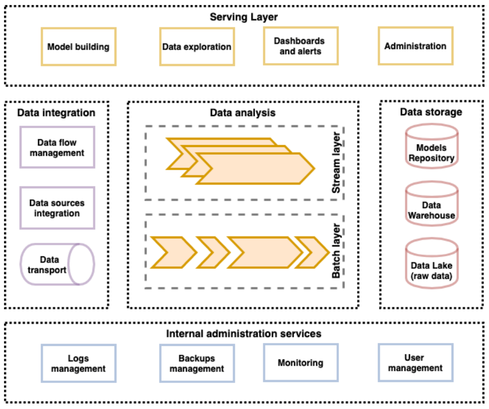

Fig. 2. An overview of the software architecture of FADI

B. A sample implementation of FADI

One of the innovative aspect of FADI is to provide a software architecture enabling to integrate together various open source tools dedicated to the management of Big Data. Indeed, we present a sample implementation of FADI that allows to set-up an end-to-end example enabling to collect data, ingest it, analyze it, and visualize it. For more details about this sample implementation, interested readers may refer to the FADI GitHub project[2]. It is noteworthy that FADI is customizable and the user may replace any tool by another one. In the following, we introduce the adopted tools:

- For the **internal administration services**, we propose to use the following tools: The ELK[3] tools suite is used to manage the logs. ELK stands for the three open source tools: Elasticsarch to search information in the collected logs, Logstash to transport and process logs by collaborating with the Elasticsearch tool, and Kibana to visualize the logs and the results of its analysis. Then, Zabbix[4] is used to set-up the monitoring in the FADI infrastructure. Finally, OpenLDAP[5] is an open source implementation of the Lightweight Directory Access Protocol (LDAP) used to provide Identity and Access Management.
- We propose to use the Apache Nifi[6] tool to set up the **data integration** layer. It automates the data flow between systems: both with external systems (e.g. sensors data sources) and between internal systems (e.g. the various analysis and storage components). It allows to manage the data transport, ingestion, pre-processing, and storage.
- For the **data analysis** layer, we propose to use Apache Spark[7] in order to conduct analysis for large-scale data processing. Added to that, it supports various programming languages. It also includes libraries for diverse tasks ranging from SQL querying to streaming and machine learning, and it runs anywhere (e.g. laptop, clusters, etc.).
- For the **data storage** layer, the following database systems are selected. The data warehouse is implemented by the PostgreSQL[8] which is an object-relational database management system. It has been chosen because it contains advanced database features (e.g. aggregation, parallel queries, replication system, etc. Then, the data lake is represented by Cassandra[9] which is a (NoSQL) column database. It is known for its high-speed and online transactional data processing in data lakes.
- For the **serving layer**, we propose to use three tools. The Jupyter[10] notebook is a web application that allows to define ML models and to be coupled to Spark in order to set up a user interface. To explore and visualize data, Superset[11] and Grafana[12] are integrated in FADI. Finally, Adminer [13] is used to manage the PostgreSQL database in a web interface.

C. The CI/CD pipeline of FADI

FADI provides a streamlined way to set-up and deploy the open-source Big Data tools to various infrastructures. It is noteworthy that the only technological choices in our solution are the tools and the techniques that are dedicated to the set-up of the DevOps features (the justification of the technological choices is available in Section VI. Indeed, we propose to use the Container Orchestration Engines (COE) Kubernetes in order to be able to package each component of FADI in a Docker container. Hence, we ensure the portability, modularity and the ease of maintenance of our solution. In addition, we propose to use Helm[14] in order to automate the configuration and the deployment of Kubernetes services by defining a single Helm chart for each one and to integrate them in the main Helm chart of FADI. In Figure 3, we showcase an overview of the CI/CD pipeline of FADI. In the following, we introduce the different stages of the CI/CD pipeline:

- **Source code and Helm charts development**: In this stage, the developer implements the services, defines their configuration and edit the Helm chart of FADI. We propose two open source tools. First, Minikube[15] is used to set-up a local Kubernetes-based environment (i.e. the development and test environments) since FADI will be deployed on a Kubernetes cluster in the production environment. Second, Docker is used to containerize the different FADI services. By choosing these tools, developers may work directly from their host OS in headless virtual machines (VM) or containers using folders shared with the host system. Furthermore, the provisioning scripts (and/or the Dockerfiles) can be versioned and be partially reused when deploying the application in other

[2]https://github.com/cetic/fadi
[3]https://www.elastic.co/what-is/elk-stack
[4]https://www.zabbix.com/
[5]https://www.openldap.org
[6]https://nifi.apache.org/
[7]https://spark.apache.org/

[8]https://www.postgresql.org/
[9]https://cassandra.apache.org/
[10]https://jupyter.org/
[11]https://superset.incubator.apache.org/
[12]https://grafana.com/
[13]https://www.adminer.org/
[14]https://helm.sh/
[15]https://github.com/kubernetes/minikube

environments. Finally, when possible, the integrated services should run in separate VM/containers to maximize flexibility, isolation and portability.

- **Source code and Helm charts committing and pushing**: Once the developers finish editing the source code and the Helm charts, they commit and push these resources to two types of Source Code Management (SCM) tools. The first one is GitLab and it is used to privately customize and manage FADI. The second one is GitHub and it is used to share the different Helm charts defined by the open source community working around FADI.
- **Artifacts and Helm charts building and publishing**: The used SCMs publish the last versions of the source code and the Helm charts to the CI/CD process. We propose to use GitLabCI in order to manage the continuous integration of the source code versions. In order to find the appropriate artifacts and the right Helm dependencies, GitLabCI is connected to an artifact repository (i.e. Nexus) and a Helm chart repository respectively. This latter is connected to CircleCI[16] in order to automate the test and the publication of the Helm charts that we use in FADI. For the interested readers, you can find all the defined Helm charts with relation of FADI here: https://github.com/cetic/helm-charts/tree/gh-pages.
- **Provisioning and deployment**: At this stage, we propose to use Terraform[17] in order to provision the infrastructure in the dedicated environment (i.e. test or production). In addition, we use Helm to deploy the kubernetes services based on the pushed Helm charts in the previous stages.
- **Functional and load testing**: We propose to conduct two types of testing. First, the functional testing enables to verify whether the installed services in FADI are working normally. In this context, we provide on the FADI GitHub project a user-guide to help users to define and run test scripts using Puppeteer and Jest to test the web interfaces of each service. Second, we propose to conduct load tests in order to check whether the environment is enough capable to support a sudden increase in load or not. The load testing is a work in progress and is not public yet.
- **Monitoring**: Based on the results of the testing stage, we collect information and we send it to the development team in order to consider the detected improvements or new features to be integrated. We propose to use Zabbix and ELK to monitor all the stages of the CI/CD pipeline.

IV. Industrial Use Cases

In this section, we only introduce four research projects (i.e. BigData@MA, Grindings 4.0, ARTEMTEC, and Quality 4.0) where FADI has been developed and improved to meet project goals. The details about the use of FADI in these projects will be introduced in the next section.

The BigData@MA[18] aims at developing a specific Big Data application framework to provide an online support for decisions tasks in pharmaceutical and mechanical manufacturing sectors. Specific data ingestion connectors, streaming analytics tools and predictive maintenance models are implemented to detect anomalies and deviations. One of the use cases in BigData@MA comes from the industrial partner I-care[19]. It consists in using FADI to collect, store, analyze and monitor data coming from the API of I-care called I-see[20]. This data is produced by machine analysis systems in order to store it in a single database.

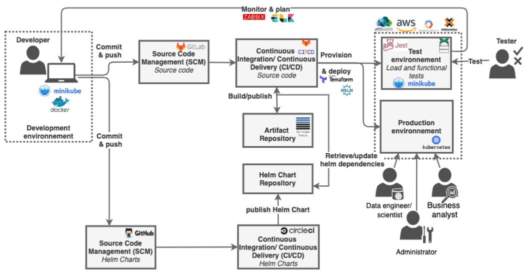

Fig. 3. An overview of CI/CD pipeline of FADI

Grinding 4.0 is a Eurostar research project identified by the convention number 1710101. It aims at creating a smart grinding operation unit for the tooling industry, fully connected to the automotive industry end users. Grinding machines unit and auxiliary equipment are connected between each others.

The ARTEMTEC project is a SKYWIN project identified by the convention number 7904. It aims to create a new platform to integrate algorithms for predictive maintenance 4.0 applicable to similar geographically distributed sites. Added to that, it focus to implement a smart help system facilitating the management of maintenance operations by an on-site technician with support of a remote expert. This help system may also suggest actions based on the results of the predictive maintenance algorithm.

The Quality 4.0 is a RFCS research project identified by the convention number 788552. Its aim is to realize the horizontal integration of quality information over the complete supply chain via the adaptive Quality4.0 platform. This latter allows online analytics of large data streams to realize decisions on product quality and provides tailored information of high reliability that can be individually exchanged with customers.

V. Evaluation

In this section, we aim to validate the various features of FADI according to the different requirements identified and listed in section II.

[16]https://circleci.com/
[17]https://www.terraform.io/
[18]https://www.bigdata-ma.eu/ and ref. MNET17/ICT1140
[19]https://www.icareweb.com/en-gb
[20]https://www.icareweb.com/en-gb/i-see/the-fastest-and-easiest-way-to-maintenance-40/

A. Ease-of-access to FADI

FADI is intended to simplify the users tasks in order to easily integrate open source big data tools and set-up a CI/CD pipeline. In the context of the different research projects, we organize a hands-on workshop for the different partners in which we present the FADI user guide, available publicly in the Github repository, and considered as ”hello world” use case. This first impression is crucial because it allows users to decide to use FADI or not. Indeed, during the ARTEMTEC project, our industrial partner, Safran Aero Boosters, has deployed FADI in its own infrastructure and starts to use it. This fact validates the **R1-Easy of deployment** requirement.

B. The portability

In the context of the research projects where we used FADI, we have validated the portability capabilities of Kubernetes by deploying FADI using different deployment methods. For BigData@MA and Grinding 4.0 projects, the associated test beds are deployed to Google Kubernetes Engine (GKE), which is a managed Kubernetes service in the Google public cloud. For Quality 4.0, the associated testbed are deployed as Kubernetes clusters in our local infrastructure using Minikube and Proxmox, an open-source server virtualization platform. Finally, the testbed associated to the ARTEMTEC project is deployed on the infrastructure of our Safran Aero Boosters partner. These different ways to deploy FADI fulfill the requirements **R1.1 Flexible Deployment** and **R1.2 portable**.

C. The interoperability

Interoperability is the ability of different systems, devices, applications or products to connect and communicate in a coordinated way. By integrating FADI in various research projects, we have validated this property. For example, in Grinding 4.0, we have connected FADI to an Azure Event Hubs[21], the fully managed, real-time data ingestion service proposed by Microsoft, to ingest industrial data. In BigData@MA project, we have connected FADI to the I-see API. Finally, during Grinding 4.0 project, we have integrated Tsimulus[22], a toolkit for generating random time series developed by the CETIC, in order to simulate vibration machines data . All these scenarios approve that FADI satisfies the **R1.3 Interoperable** requirement.

D. The scalability

Thanks to the choice of Kubernetes as a target infrastructure to deploy, FADI is a scalable platform. Indeed, it enables (1) horizontal scalability by automatically creating Kubernetes nodes and pods, and (2) vertical scalability by increasing the configuration of the CPU and RAM of the Kubernetes pods. In the context of the Grinding 4.0 project, we aim to support data bursts sent by industrial machines, via the Azure Event Hub, in order to ingest and process it in a scalable and efficient way. The first results show a reactive FADI face to a huge quantity of data. This behavior validates the **R2-Automatic Elastic Orchestration** requirement. In the future, we aim to add a load testing mechanism associated to a monitoring tool to observe how FADI behaves.

E. The modularity and extensibility

The use of Helm in FADI enables to ensure its modularity and extensibility. Indeed, it allows to deploy multiple components integrated in FADI based on Helm charts as a single application. Adding a new component or release is simple and requires only the integration of the appropriate Helm chart either by defining it or by uploading it from the Helm Hub[23]. This feature is very important especially for industrial partners who have proprietary components that they would like to integrate in FADI. For instance, in the Quality 4.0 project, we are working on integrating internal components of our industrial partner in FADI. To do so, we are defining a Helm chart for each one separately. Then, we will add these dependencies in the FADI Helm Chart. Such a task may require between one and four days in general. To support the user during this task, we provide the user with an automatic test and publication process of the Helm chart here[24]. Thus, we validate the **R3-Usability** requirement.

F. Supporting heterogeneous data

FADI supports the integration and the use of the heterogeneous data. Indeed, it has been used in at least four research projects in which we were faced on different types of data in each one. For instance, in the Grindings 4.0 project, FADI collects and ingests JSON data. In the BigData@MA project, the I-see API sends web sockets data. During ARTEMTEC project, FADI ingests MS Excel spreadsheets. Added to that, it is possible to handle heterogeneous data in the same use case. For instance, in Quality 4.0 project, we process temperature measurements captured by cameras, values sent from sensors, and data coming from databases. All these use-case match with the **R3-Usability** requirement.

VI. Related works

Over the past decade, a new paradigm has emerged which is referred to as the DevOps methodology. One of the most important pillars of the DevOps is the automation of the configuration management of a given software solution by putting in place innovative techniques like continuous integration, test driven development, build/deployment automation, etc. [5]. Indeed, it consists in configuring the infrastructure where the solution will be launched and deploying its different components by controlling their dependencies and interconnections. Against this background, applications tend to be specified and realized in a modular and generic fashion using containers in order to ease their portability and scalability. In order to manage these containers, COEs enable to deploy, monitor, and dynamically control the configuration and the lifecycle of multi container packaged applications in various hosts [6].

[21]https://azure.microsoft.com/en-us/services/event-hubs/

[22]https://tsimulus.readthedocs.io/en/latest/

[23]https://hub.helm.sh/

[24]https://github.com/cetic/helm-charts

Today, it exists various COE-based solutions that we cite the most common and efficient Docker Swarm[25], Kubernetes, etc. Although these solutions are relevant and seem similar, Kubernetes represents some highlights and is the most suitable for the FADI requirements. Indeed, Arundel et al [7] present Kubernetes as the de facto standard COE and as the most widely used solution in production environments nowadays. Since it has a large and cooperative community behind it. Then, Jawarneh et al [8] conduct a functional and performance comparison between the most known and used COEs. They conclude that Kubernetes presents the most capabilities to manage and deploy production-level services even if it requires very high provisioning time compared to other solutions due to its complex and complete architecture. Afterward, coupling the Helm package manager to Kubernetes enables to automate the configuration and the deployment of its services [9].

Added to the DevOps aspect, we are interested to analyze the solutions dedicated to the Big Data management. In the one hand, it exists the category of the SaaS-based solutions offered by the cloud providers (e.g. Google Cloud, Microsoft Azure and Amazon AWS). They offer a set of services grouped under the term "Big Data and Analytics". These services are provided "off-the-shelf" (i.e. technological bricks that are easy to acquire and take in hand). Despite the maturity of this kind of solutions, their main drawback is the problem of the vendor lock-in. In the other hand, we have the proprietary solutions. The Statistical Analysis Systems (SAS)[26] is a suite of products created by SAS Institute which performs advanced analysis, data management, business intelligence and various different tasks. It has been used for a while by several major organizations. Due to its advanced features, SAS is an extremely popular solution. However its license is extremely expensive and it is not open source. The Hortonworks Data Platform (HDP) [10] is an open source Apache Hadoop-based framework enabling the management of large amount and heterogeneous data using ML and deep learning techniques. It enables to set-up and deploy data-intensive applications in a scalable and flexible way based on the package manager YARN to enable the use of Docker containers. Nevertheless, its installation is highly complex and the resource requirements base line is rather high which means that if used on premise, you need to think of about at least 10 machines for a minimal reasonable deployment. The Digazu[27] is an end-to-end solution to manage and process real time data. Indeed, it enables to collect data and to store it in a data lake. Then, it enables to define data management workflows and pipelines. Although this solution is interesting, it is not open source. WiDE[28] is a tool to collect measurements coming from sensors and to analyze it using self-defined libraries in Python. It enables to monitor the status of sensors during a given production process and to detect device failures/anomalies. The Wide tool provide the possibility to define dashboards in order to visualize the analytics results. This solution lacks of portability and modularity. Hopsworks [11] is an open-source platform dedicated to ML pipelines design and operation. It supports open source solutions to conduct data engineering/science. To the best of our knowledge, it does not provide a CI/CD mechanism in order to ease its deployment.

[25]https://docs.docker.com/engine/swarm/

[26]https://www.sas.com/fr_fr/home.html

[27]https://www.digazu.com/

[28]http://www.widetech.com/software/wide/

VII. Conclusion

In this paper, we proposed FADI which is a customizable end-to-end big data platform enabling the deployment and the integration of open source tools in a portable and scalable way. Indeed, we presented an overview of the different components of FADI and its CI/CD pipeline using Kubernetes and Helm. Then, we introduced the four research projects in which FADI has been integrated.

Currently, we are working on applying FADI to other real use cases in order to identify possible discrepancies and make our work more reliable. In the medium term, we focus on finishing the automated load testing process in order to publish it in the GitHub project. Afterward, we aim to integrate in FADI a Single-Sign-On (SSO) feature. In long term, we focus on using FADI in Edge/multi-cloud computing environments. Added to that, we target to add security activities in the DevOps pipeline (i.e. DevSecOps) in order to ensure a secure integration and to automatically prevent, detect, correct and trace vulnerabilities and security issues.

Acknowledgment

This work has been funded by the MANUNET research project BigData@MA identified by the project reference MNET17/ICT1140. We would like to thank the rest of our team composed by M. Badri, F. Steels, T. Selleslagh, and A. Ayadi for their contributions in the realization of FADI.

References

[1] "NIST big data interoperability framework:," Tech. Rep., Oct. 2019. [Online]. Available: https://doi.org/10.6028/nist.sp.1500-1r2
[2] P. Zikopoulos and C. Eaton, *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw Hill Professional.
[3] M. Khan, X. Wu, X. Xu, and W. Dou, "Big data challenges and opportunities in the hype of industry 4.0," in *IEEE International Conference on Communications, ICC 2017, Paris*. IEEE, 2017, pp. 1–6.
[4] ISO/IEC, "Iso/iec 25010 system and software quality models," Tech. Rep., 2010.
[5] M. Hüttermann, *DevOps for Developers*. Berkeley, CA: Apress, 2012.
[6] E. Casalicchio, "Container orchestration: A survey," in *Systems Modeling: Methodologies and Tools*, ser. EAI/Springer Innovations in Communication and Computing, A. Puliafito and K. S. Trivedi, Eds. Cham: Springer International Publishing, 2019, vol. 14, pp. 221–235.
[7] J. Arundel and J. Domingus, *Cloud Native DevOps with Kubernetes: Building, Deploying, and Scaling Modern Applications in the Cloud.*
[8] I. M. A. Jawarneh, P. Bellavista, F. Bosi, L. Foschini, G. Martuscelli, R. Montanari, and A. Palopoli, "Container orchestration engines: A thorough functional and performance comparison," in *ICC - IEEE International Conference on Communications (ICC)*. IEEE, May 2019.
[9] S. Buchanan, J. Rangama, and N. Bellavance, *Introducing Azure Kubernetes Service*. Berkeley, CA: Apress, 2020.
[10] Cloudera, *Hortonworks Data Platform (HDP) 3.0 — Faster, Smarter, Hybrid Data*.
[11] *Hopsworks - Data Intensive AI: Design and Operate ML Applications at Scale (White paper)*, logical clocks.