Managing Lifecycle of Big Data Applications

Ivan Ermilov[1], Axel-Cyrille Ngonga Ngomo[2], Aad Versteden[3], Hajira Jabeen[4], Gezim Sejdiu[4], Giorgos Argyriou[5], Luigi Selmi[6], Jürgen Jakobitsch[7], and Jens Lehmann[4]

[1] Instituts für Angewandte Informatik, Hainstraße 11, 04107 Leipzig, Germany,
iermilov@informatik.uni-leipzig.de
[2] Paderborn University, Warburger Str. 100, 33098 Paderborn, Germany
axel.ngonga@upb.de
[3] Tenforce, Havenkant 38, 3390 Leuven, Belgium
aad.versteden@tenforce.com
[4] University of Bonn, Römerstraße 164, 53117 Bonn, Germany
[jabeen,sejdiu,jens.lehmann]@cs.uni-bonn.de
[5] University of Athens, Ilisia, Panepistimioupolis, 15703 Athens, Greece
gioargyr@di.uoa.gr
[6] Fraunhofer IAIS, Schloss Birlinghoven, 53757 Sankt Augustin, Germany
luigi.selmi@iais.fraunhofer.de
[7] Semantic Web Company GmbH, Mariahilfer Strasse 70, 1070 Vienna, Austria
juergen.jakobitsch@semantic-web.com

Abstract. The growing digitization and networking process within our society has a large influence on all aspects of everyday life. Large amounts of data are being produced continuously, and when these are analyzed and interlinked they have the potential to create new knowledge and intelligent solutions for economy and society. To process this data, we developed the Big Data Integrator (BDI) Platform with various Big Data components available out-of-the-box. The integration of the components inside the BDI Platform requires components homogenization, which leads to the standardization of the development process. To support these activities we created the BDI Stack Lifecycle (SL), which consists of development, packaging, composition, enhancement, deployment and monitoring steps. In this paper, we show how we support the BDI SL with the enhancement applications developed in the BDE project. As an evaluation, we demonstrate the applicability of the BDI SL on three pilots in the domains of transport, social sciences and security.

Keywords: Big Data, Software Methodologies, Microservice Architecture

1 Introduction

Digitization is taking an increasingly important role in our everyday life. The analysis and interlinking of the large amounts of data generated permanently by our society has the potential to build the foundation for novel insights and

intelligent solutions for economy and society. However, processing this information at scale remains the privilege of a chosen few, who can integrate and deploy the plethora of Big Data processing tools currently available. What is needed are innovative technologies, strategies and competencies for the beneficial use of Big Data to address societal needs. In the Big Data Europe (BDE) project, we address this need by developing the Big Data Integrator (BDI) Platform. This platform is designed to accommodate various Big Data components such as Hadoop[8], Spark[9], Flink[10], Hive[11] and others[12] available out-of-the-box. The BDI Platform is based on the Docker technology stack and seamlessly integrates components into complex architectures necessary to solve societal data challenges. The platform is being used in 7 pilots in the areas of Health, Food, Energy, Transport, Climate, Social Sciences, and Security. Each of the pilots builds upon available Big Data components[13] and develops new ones specific to the challenge.

The integration of the components inside the BDI Platform requires an approach to the homogenization of components. To support the development of the BDI components and their integration in the BDI platform, we propose a novel methodology of developing Big Data applications dubbed BDI Stack Lifecycle (SL). Our methodology is based on the experience of developing Docker images inside the BDE project and their integration in the pilots. It consists of the following steps: (1) development, (2) packaging, (3) composition, (4) enhancement, (5) deployment and (6) monitoring. The prime goal of BDI SL is the creation of complex Big Data applications dubbed BDI Stacks. Each instantiation of BDI Stack methodology is a complex use-case-driven application designed to address a particular challenge (e.g., processing sensor data from taxi drives in Thessaloniki).

The BDI Platform itself was described in the previous work [1]. In this paper, we focus on the methodology for developing BDI Stack applications. In particular, the contributions of this paper are as follows:

- We propose a methodology for developing and maintaining complex Big Data applications using Docker containers dubbed BDI Stack Lifecycle.
- We describe the supporting tools for the six steps of the BDI SL.
- We apply our methodology on the real-world pilots and show its applicability.

2 Related Work

The BDI Platform is a distribution of Big Data components available out-of-the-box for easy installation and setup. The idea of creating library of components

[8] `http://hadoop.apache.org/`
[9] `https://spark.apache.org/`
[10] `https://flink.apache.org/`
[11] `https://hive.apache.org/`
[12] `https://www.big-data-europe.eu/bdi-components/`
[13] At the time of writing, more than 30 components are available in BDE Components Library.

is not novel and a lot of other distributions are available at the time of writing [8]: Hortonworks[14], Cloudera[15], MapR[16], Bigtop[17]. Most of distributions are provided as open source software under Apache 2.0 license. The key difference between the BDI Platform and other distributions are (1) the flexibility of the Platform, which enables creating custom stacks for the pilots, and (2) the availability of documentation and use cases. To the best of our knowledge, we are the first project, which summarizes the development process and proposes a methodology for developing big-data stack applications.

Data management lifecycles relevant to this work have been developed in the domain of data warehousing. CRISP DM [17] is a generic data mining methodology, which can be applied in any kind of data mining tasks. CRISP DM breaks down the lifecycle of a data mining project into six phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. It is common to extend or modify CRISP DM for particular needs of a project. For example, in [5] the authors extend CRISP DM to process big scientific data. CRISP DM is data centric and does not tackle service composition or deployment problems in detail. In the scope of our BDI SL, CRISP DM only covers the first development step, where a single Spark or Flink application is being designed to address a particular data mining challenge.

The other relevant research field, which studies the software lifecycles is Continuous Integration/Continuous Delivery (CI/CD). CI/CD assumes that the new software needs to be developed and has the goals of fast and frequent release, flexible product design and architecture, continuous and rapid experimentation etc. (see the recent state-of-the-art survey [14]). The main difference between the methodologies represented in CI/CD studies and our approach is that we reuse existing components for assembling complex architectures.

Mesosphere DCOS[18] and Canonical Juju [18] simplifies the deployment of complex architectures. Juju operates with charms, which are packaged components with preprogrammed dependencies and connection logic. The charms can be written in any programming language and are available in Juju Store [19] for reuse. Juju creates a thin layer over resource provisioning. To implement an architecture using Juju, it is necessary to program deployment logic, when the required components are not available in Juju Store or the available implementation does not fit the user requirements. For example, in [15] the authors implement flexible architecture for deploying applications on Apache Storm. In our approach the deployment logic is encapsulated into Docker images, which allows user to employ default Docker facilities such as *docker-compose*.

The architectures for big data applications include Model Deployment and Execution Framework (MDEF) [7], Extended IoT Framework [16], Big Data Processing Framework for Healthcare Applications [13], and On Demand Data

[14] https://hortonworks.com/
[15] https://www.cloudera.com/
[16] https://mapr.com
[17] http://bigtop.apache.org/
[18] https://dcos.io/
[19] https://jujucharms.com/store

Analytics in High Performance Cluster (HPC) Environments [6] among others. These architectures challenge a particular problem, which is often specific to the infrastructure. For example, in [6] the authors claim that facility resources (i.e. High Performance Cluster) cannot be simply partitioned or repurposed for specific architectures. Therefore, they propose an on demand solution for creating Spark instances on fly for HPC.

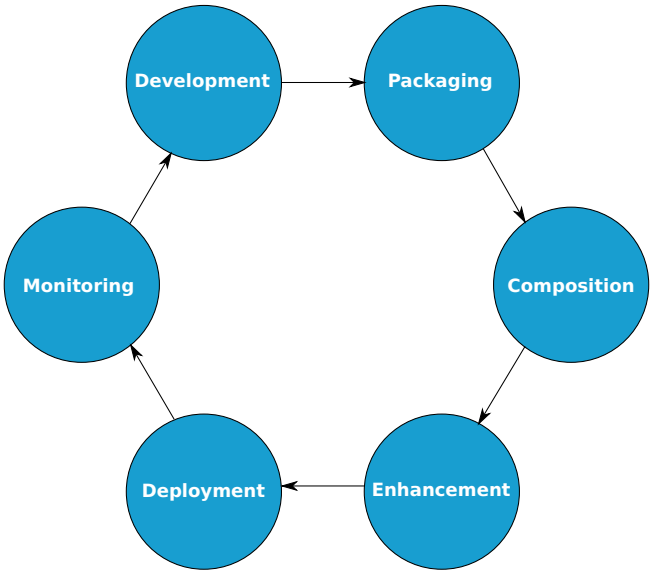

Fig. 1. BDI Stack Lifecycle.

3 BDI Stack Lifecycle

In this section, we describe BDI SL methodology (see Figure 1), which supports the creation, deployment and maintenance of the complex Big Data applications dubbed BDI Stacks. The BDI SL consists of the following steps:

1. **Development** includes the engineering of BDI Stack components such as, for example, Spark or Flink applications.
2. **Packaging** is the dockerization and publishing of the developed or existing third-party BDI Stack components.
3. **Composition** stands for the assembly of a BDI Stack, where we defined a BDI Stack as the integration of several BDI components (most commonly to address a particular data processing task).
4. **Enhancement** step is a process of extending BDI Stack with enhancement tools such as, for example, *Init Daemon* for the creation of dataflow, and *Logging* facilities for monitoring the health of a BDI Stack.

5. **Deployment** is the instantiation of a BDI Stack on physical or virtual servers.
6. **Monitoring** consists of observing the status of a running BDI Stack and can lead to a reiteration of the BDI components and architecture of the BDI Stack.

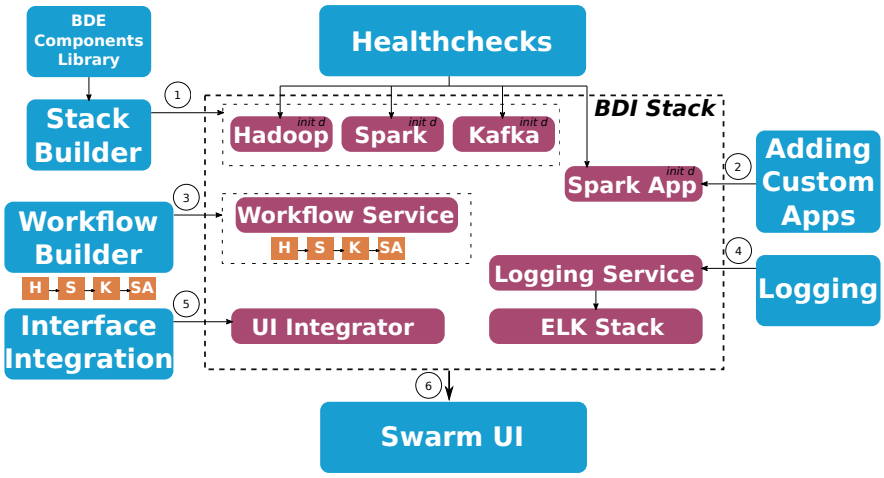

Fig. 2. BDI Stack Assembly and Architecture. (1) Picking up the ready-made BDI components from BDE Components Library. (2) Adding custom applications such as Spark applications. (3) Ensuring start-up of the BDI stack with BDI Workflows. (4) Adding logging facility. (5) Integrating interfaces with UI Integrator. (6) Deploying BDI Stack with Swarm UI.

To support the BDI SL methodology, we developed documentation and enhancement tools for each of the steps.

For the (1) **development** step we created application templates for Spark and Flink for the most common programming languages (Java, Scala, Python) [3]. We also provide Spark Notebook and Apache Zeppelin as a part of Spark/HDFS Workbench. Spark/HDFS Workbench is an integrated environment for developing, testing and running Spark applications. It can be deployed on any Docker host with only one CLI command [2]. Spark Notebook and Apache Zeppelin are the parts of the Spark/HDFS Workbench and can be used for interactive programming and execution of Spark jobs.

For the (2) **packaging** step we provide a library of ready-made components, which are created using best-practices on dockerization of Big Data technologies. More than 30 components are available in the BDE Github organization[20] at the time of writing. Also, we use the open source Docker technology, which has a community-driven components repository called Docker Hub. In case of a missing

[20] https://github.com/big-data-europe/

component or when BDI component does not meet project requirements, it is always possible to search the Docker Hub.

All the components from the BDE Github repository have example docker-compose snippets, which can be used for the (3) **composition** step. With this comprehensive library, the users are able to reuse existing BDI Stacks and create new ones by a simple extension. Additionally, we provide a graphical user interface for assembling BDI Stacks dubbed BDI Stack Builder.

For the (4) **enhancement** step we provide a set of enhancement applications aimed to improve usability of a BDI Stack. The enhancements include UI Integrator, Init Daemon, Healthchecks, Workflow Monitor, and Logging together with ELK Stack for logs visualization. We describe the enhancements in more detail in section 4.

The (5) **deployment** step can be done by using both widely adopted *docker-compose* application or with our Swarm UI interface.

The (6) **monitoring** step is performed using Logging facility and visualized with ELK stack.

4 BDI Stack Assembly and Architecture

The central steps of BDI Stack assembly are (3) **composition** and (4) **enhancement** steps. In this section we describe them in more detail.

Table 1. Use Cases Summary.

Pilot Name	**SC4**	**SC6**	**SC7**
Domain	Transport	Social Sciences	Security
Data Format	Sensor Data	Tabular Data (CSV)	Satellite Images
Number of components in BDI Stack	9	8+	11

In Figure 2 we show the process of BDI Stack assembly as well as resulting architecture of an example BDI Stack. As shown in Figure 2, user employs *Stack Builder* application[21] (1) to select required BDI components from *BDE Components Library* and have an initial version of BDI Stack with Hadoop, Spark and Kafka[22]. Then, the user adds custom applications (2) such as Spark App in the example, which performs the use case specific tasks. Often it is desired to ensure the execution order of a BDI Stack, for example, data processing should happen after data acquisition. The execution order can be controlled using docker native HEALTHCHECK Docker facility. However, due to the limitations of the native solution, we introduce an approach based on Init Daemon. With Init Daemon enabled components, it is possible to assemble a workflow (3) using

[21] `https://github.com/big-data-europe/app-stack-builder`

[22] `https://kafka.apache.org/`

Workflow Builder application[23]. The Workflow Service[24] executes a workflow during the BDI Stack deployment and ensures the correct execution order of the components of the stack. The logging facility[25] needs to be added as a Logging Service (4) together with ELK Stack for logging visualization. Interfaces from all the BDI Stack components (e.g. Hadoop, Spark, Kafka, Spark App) are aggregated using UI Integrator[26] (5) and can be accessed with a common GUI [4]. The resulting BDI Stack, which has docker-compose format, is deployed with Swarm UI[27] (6).

The enhancement applications such as *mu-bde-logging* and *mu-pipeline-service* are developed using `mu.semte.ch` semantic technology stack [19].

5 Use Cases

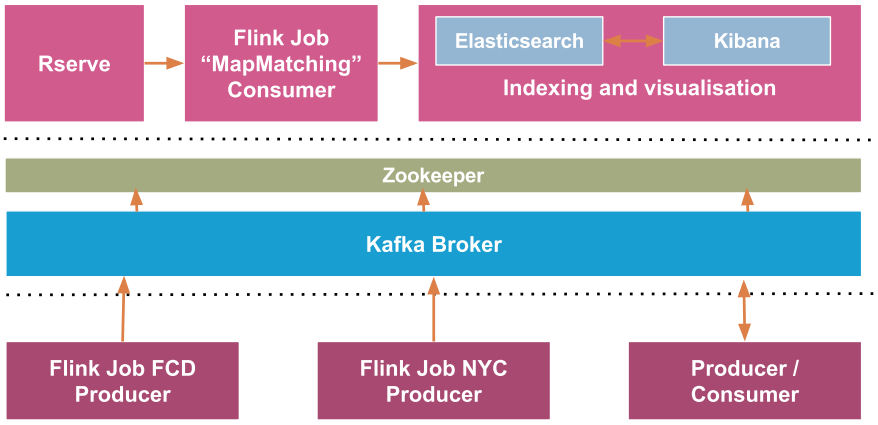

Fig. 3. Transport pilot architecture.

In this section we present three selected pilots developed with the BDI SL methodology. Each of the pilots correspond to a societal challenge. As shown in the summary Table 1, the pilots process the data in various formats and have different number of components in a BDI Stack. In the following, we describe the pilots in more detail.

[23] https://github.com/big-data-europe/app-pipeline-builder
[24] https://github.com/big-data-europe/mu-pipeline-service
[25] https://github.com/big-data-europe/mu-bde-logging
[26] https://github.com/big-data-europe/app-integrator-ui
[27] https://github.com/big-data-europe/app-swarm-ui

5.1 Transport

The H2020 Societal Challenge 4[28], Smart Green and Integrated Transport, covers a broad topic ranging from urban mobility, to safety, logistics, transport system integration, infrastructure monitoring and planning. Transport systems consume

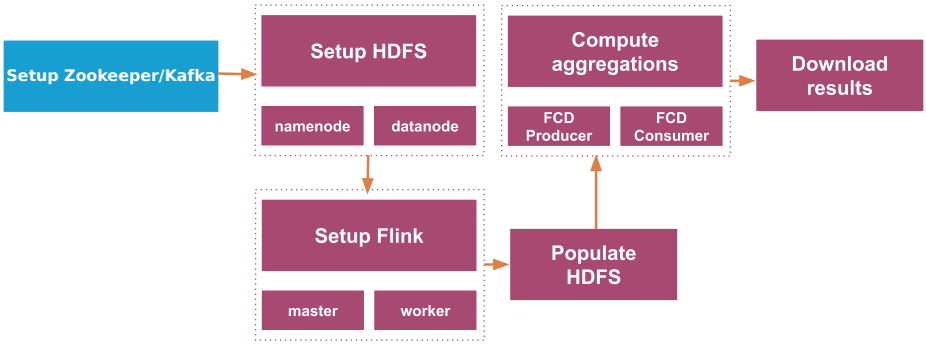

Fig. 4. Transport pilot initialization workflow.

huge flows of data to provide services, monitor infrastructures and discover the usage patterns in order to forecasting what will be the status in the near or distant future. All these systems consume streams of data from different sources and in different formats. In the SC4 pilot we have therefore decided to build a pilot that can ingest, transform, integrate and store streams of data that have spatial and temporal dimensions. One of the project partner, CERTH-HIT, is managing a system that monitors the traffic flow in Thessaloniki, Greece, using floating car data from a transport company. The legacy system is based on a relational database, stored procedures and R scripts to map-match the location of the vehicles to the road segments and compute the traffic flow and average speed among other statistical parameters. The result of the computation is used for monitoring and as input for forecasting the value of the parameters in the near future and is made available through a web service. The aim of the pilot is to address the scalability issues of the current system leveraging the availability of distributed frameworks and the containerization technology for the deployment of services in different environments.

The pilot is based on the microservices architecture where different software components, producers and consumers, communicate through a messaging system connecting data sources to data sinks. Producers and consumers are implemented as Flink jobs while Kafka has been chosen as the messaging system. The producer fetches the data every two minutes from the web service, stores the records sets into Hadoop HDFS, transforms the records into a binary format, using a schema shared with the consumer, and finally sends the records to a

[28] https://www.big-data-europe.eu/pilot-transport/

Kafka topic. The consumer reads the records from the Kafka topic and process them at event time applying the map matching function. The consumer must connect to an R server where an R script has been installed to perform the computation for the map matching using the road network data from Open Street Map stored in a PostGIS database. The consumer adds the identifier of the road segment as an additional field to the original record and finally aggregates the records per road segment and in time windows to compute the traffic flow and the average speed in each road segment. The result of the aggregation can be sent to Hadoop HDFS or to Elasticsearch[29]. From Elasticsearch different visualizations can be created easily with Kibana[30]. The records with the aggregated values stored in Elasticsearch will be used as input to a forecasting algorithm to predict the traffic flow. All the components are available as Docker images and a docker-compose file has been created adding the initialization service and the UI provided by the BDI Stack in order to start the services in the right sequence from the browser (e.g. Zookeeper before Kafka and PostGIS as well as Elasticsearch before the consumer).[31]

5.2 Social Sciences

The H2020 societal challenge 6[32], "Europe in a changing world - Inclusive, innovative and reflective societies" roughly covers topics improving the understanding of European societies in the context of the public sector. The Big Data Europe pilot implementation for this societal challenge exposes the foundation of making budget data comparable across European municipalities. Additionally the pilot architecture (see Figure 5) can be used as a base for document processing at scale. The basic implementation is ingesting budget data from three Greek municipalities: Athens, Thessaloniki and Kalamaria. These datasets (in CSV format) are collected on a daily basis and transformed to RDF using ELOD's schema[33][34]. The transformed datasets are used to calculate financial ratios[35] and compare incomes/expenses of the municipalities.

The entry point for input data is Apache Flume agents[36], which can be configured according to the availability and format of budget datasets. All Flume agents are configured to store raw data into Hadoop HDFS and create an Apache Kafka message for each file, containing the name of the source document as key and the contents of the file as an array of bytes. Apache Spark acts as a consumer of Kafka messages. The Spark job consumes messages in parallel with the possibility of being scaled to any number of nodes. SC6 pilot makes use of SPI[37] to determine a suitable parser for a given file, which in turn will

[29] https://www.elastic.co/products/elasticsearch
[30] https://www.elastic.co/products/kibana
[31] https://github.com/big-data-europe/pilot-sc4-fcd-applications
[32] https://www.big-data-europe.eu/pilot-social-sciences/
[33] http://linkedeconomy.org/en
[34] https://github.com/LinkedEcon/LinkedEconomyOntology-ELOD
[35] http://www.accountingverse.com/managerial-accounting/fs-analysis/financial-ratios.html
[36] http://flume.apache.org
[37] https://docs.oracle.com/javase/tutorial/sound/SPI-intro.html

transform the source file into RDF and upload the resulting triples to a Virtuoso Triple Store[38]. After a successful upload financial ratios[39] are calculated for the newly added data. Those aggregations are done using the SPARQL[40] query language. Both steps, the creation of the initial RDF dataset and the calculation of the financial ratios make use of PoolParty Semantic Suite's[41] capabilities as a clearing house for all literal terms that are involved. This means that financial terms, unknown to a wider public, can be translated, annotated and unified for municipalities of different languages. Finally a dashboard of financial ratios is created based on PoolParty's GraphSearch[42]. This way financial ratios are easily comparable. Financial ratios and periods of time as well as municipalities can be chose to be compared.

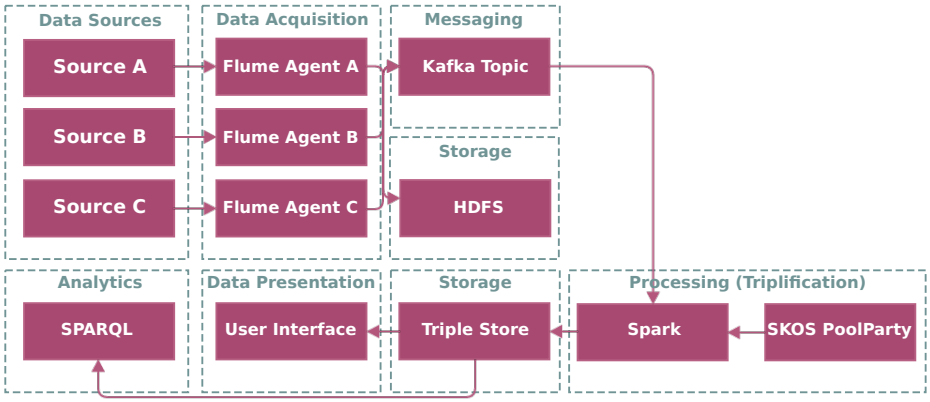

Fig. 5. Social sciences pilot architecture.

Note that the pilot setup can also be used as a template to get started with the relevant Big Data technologies. It comprises core Big Data tools such as Flume, Kafka, HDFS and Spark and shows how these can be setup to work together. The architecture is driven by the requirement of making the cooperation between a technical expert and a domain expert as easy as possible in that both have distinct points of control, for example, in the selection of source data and the transformation of the source files into RDF.

The pilot's stack is configured as a whole using a single docker-compose file, which can be run by Docker Swarm.[43] Although all involved docker images support BDE's Init-daemon, docker-compose "depends_on" feature is sufficient in

[38] https://virtuoso.openlinksw.com/
[39] https://en.wikipedia.org/wiki/Financial_ratio
[40] https://www.w3.org/TR/sparql11-query/
[41] https://www.poolparty.biz/
[42] https://www.poolparty.biz/poolparty-semantic-graph-search-server/
[43] https://github.com/big-data-europe/pilot-sc6-cycle2

the case of SC6. The pilot, however, makes use of Docker HEALTHCHECK[44] facility. The BDI Logging collects the health information about docker containers and store it in the ELK stack, thus the applications, that do not expose a graphical user interface can be monitored (e.g. Apache Kafka). Apache Spark's BDE UI as well as unified logging[45] make sure the pilot's general working can be followed easily.

5.3 Security

The security pilot combines data relating to security domain[46], the H2020 Societal Challenge 7 (SC7). Sources of such data are Earth Observation products and the combination of news articles with user-generated messages from social media. In the first case, the pilot processes satellite images in order to detect changes in land cover or land use. In the second one, it processes news from the web sites and social media in order to detect events. Combining the outcomes, we achieve an integration of remote sensing sources with social sensing ones.

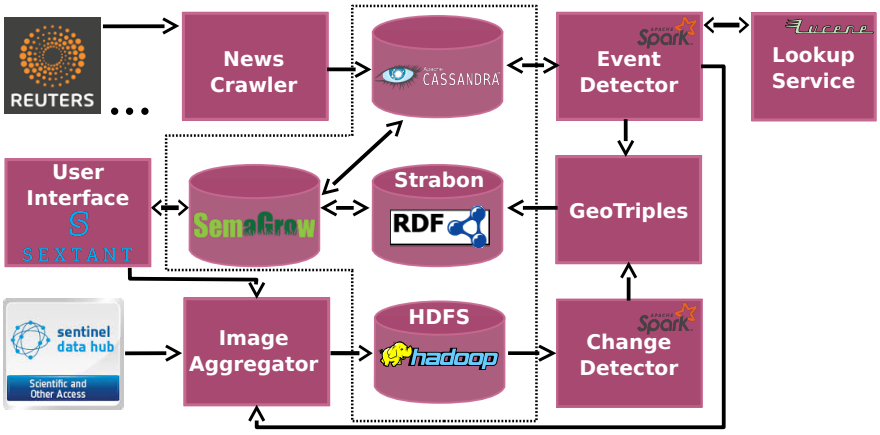

Fig. 6. Security pilot architecture.

The high-level architecture is depicted in Figure 6 and represents three workflows. the top components (News Crawler, Cassandra, Event Detector and Lookup Service) implement event-detection workflow. The middle layer (Sextant, SemaGrow, Strabon and Geotriples) implements the activation-workflow. The bottom layer (Image Aggregator, HDFS and Change Detector) implements the change-detection workflow.

For the event-detection workflow, the News Crawler runs periodically and ingests data from various public news streams like Twitter and Reuters. All

[44] https://docs.docker.com/engine/reference/builder/#healthcheck
[45] https://github.com/big-data-europe/mu-bde-logging
[46] https://www.big-data-europe.eu/security/

these news are stored in Cassandra[47] where they are processed by the Event Detector. Event Detector periodically executes a Spark job to cluster the news items into events and to associate them with one or more geo-locations with the help of the Lookup Service. The Lookup Service is based on Lucene index. It accepts location names in plain text and returns their geo-coordinates by indexing 180,000 location names from the GADM dataset[48].

The change-detection workflow processes satellite images from ESA's Copernicus Open Access Hub[49] (Open Hub). The Image Aggregator downloads large files that contain images with their metadata. These images cover a certain area of interest during a specific time period as requested by the user. The images are stored in Hadoop HDFS[50] being accessible to Spark nodes and thus available to the Change Detector. After each processing request[51], two satellite images are ingested in Change Detector and areas of highly possible changes are returned.

Sextant [12] is the basic component of the activation workflow and the entry point for the pilot. It has been widely extended for the pilot's needs and provides a graphic interface for the user to select either an event or a change detection triggering the corresponding workflow. SemaGrow, Strabon and GeoTriples provide support for the event- and change-detection workflows and complement the activation one. Geotriples [11] receives descriptions of areas (the output of Change Detector) or summaries of events (the output of Event Detector) and converts them into RDF. The output of the Geotriples is then stored in Strabon [10] which is a spatio-temporal triplestore that efficiently executes GeoSPARQL and stSPARQL queries. Sextant provides access and visualization the data that are stored either in Cassandra and Strabon through Semagrow [9].

All components of the pilot are provided as Docker images.[52] They run as Docker containers within Docker Swarm in order to be enhanced as a BDI Stack. The whole pipeline is deployed in the BDE Platform by running a docker-compose file that describes all the services of the pilot.

Executing a Docker Compose application will launch all Docker containers simultaneously. In order to avoid the immediate launch of all Docker containers and to control the execution order we enrich each service of the pilot with the native HEALTHCHECK Docker facility. Thus, we manage to keep a certain order of the initiation of the services and we make sure that every service will start running after it is confirmed that the services it is depended on, have completed a healthy start.

Ensuring the correct execution order of the components of the stack we have a ready pipeline for use. The user is able to select the provided functionality from Sextant choosing either of the two workflows described earlier. Aggregating the interfaces of the BDI Stack components (Sextant, Hadoop and Spark), which the

[47] `http://cassandra.apache.org/`
[48] `http://www.gadm.org/`
[49] `https://scihub.copernicus.eu/`
[50] The typical loading time for a set of images: 400 seconds.
[51] The typical Spark job execution time for Change Detector: 1000 seconds.
[52] `https://github.com/big-data-europe/pilot-sc7-change-detector`

SC7 pilot uses with the UI Integrator allows the user to monitor the progress of the selected workflow.

6 Conclusions and Future Work

In this paper, we presented a Big Data Integrator Stack Lifecycle methodology for creation, management and maintenance of Big Data applications. We showed how the six steps of the lifecycle are supported within the BDI platform. Three pilots are showcased as an evaluation of the presented lifecycle. The core of the BDI SL methodology is a composition step, which depends on the underlying technology (i.e. Docker). Thus, it will be hard to transfer the methodology to generic environments or adapt it for usage in High Performance Clusters (HPC) or vendor specific environments, which do not run on Docker or can not execute docker containers. The linear initialization workflows as presented in the pilots are tolerant to cascading failures. In the future, we will address this issue by further developing Docker images for all the components to be fault-tolerant and not dependent on the execution order. Additionally, we plan to test and evaluate not only three, but all 7 pilots and report on the evaluation results.

7 Acknowledgments

This work was supported by grant from the European Union's Horizon 2020 research Europe flag and innovation program for the project Big Data Europe (GA no. 644564).

References

1. Auer, S., Scerri, S., Versteden, A., Pauwels, E., Charalambidis, A., Konstantopoulos, S., Lehmann, J., Jabeen, H., Ermilov, I., Sejdiu, G., Ikonomopoulos, A., Andronopoulos, S., Vlachogiannis, M., Pappas, C., Davettas, A., Klampanos, I.A., Grigoropoulos, E., Karkaletsis, V., de Boer, V., Siebes, R., Mami, M.N., Albani, S., Lazzarini, M., Nunes, P., Angiuli, E., Pittaras, N., Giannakopoulos, G., Argyriou, G., Stamoulis, G., Papadakis, G., Koubarakis, M., Karampiperis, P., Ngomo, A.C.N., Vidal, M.E.: The BigDataEurope Platform - Supporting the Variety Dimension of Big Data. In: 17th International Conference on Web Engineering (ICWE2017) (2017), http://jens-lehmann.org/files/2017/icwe_bde.pdf
2. Ermilov, I.: Scalable spark/hdfs workbench using docker. https://www.big-data-europe.eu/scalable-sparkhdfs-workbench-using-docker/, Retrieved on May 21, 2017 (2016)
3. Ermilov, I.: Developing spark applications with docker and bde. https://www.big-data-europe.eu/developing-spark-applications-with-docker-and-bde/, Retrieved on May 21, 2017 (2017)
4. Ermilov, I.: User interface integration in bdi platform (integrator ui application). https://www.big-data-europe.eu/user-interface-integration-in-bdi-platform-integrator-ui-application/, Retrieved on May 21, 2017 (2017)

5. Grady, N.W.: Kdd meets big data. In: Big Data (Big Data), 2016 IEEE International Conference on. pp. 1603–1608. IEEE (2016)
6. Harney, J., Lim, S.H., Sukumar, S., Stansberry, D., Xenopoulos, P.: On-demand data analytics in hpc environments at leadership computing facilities: Challenges and experiences. In: Big Data (Big Data), 2016 IEEE International Conference on. pp. 2087–2096. IEEE (2016)
7. Heit, J., Liu, J., Shah, M.: An architecture for the deployment of statistical models for the big data era. In: Big Data (Big Data), 2016 IEEE International Conference on. pp. 1377–1384. IEEE (2016)
8. Jabeen, H.: Bde vs. other hadoop distributions. `https://www.big-data-europe.eu/bde-vs-other-hadoop-distributions/`, Retrieved on May 21, 2017 (2016)
9. Konstantopoulos, S., Charalambidis, A., Mouchakis, G., Troumpoukis, A., Jakobitch, J., Karkaletsis, V.: Semantic web technologies and big data infrastructures: Sparql federated querying of heterogeneous big data stores. ISWC Demos and Posters Track (2016)
10. Kyzirakos, K., Karpathiotakis, M., Koubarakis, M.: Strabon: a semantic geospatial dbms. The Semantic Web–ISWC 2012 pp. 295–311 (2012)
11. Kyzirakos, K., Vlachopoulos, I., Savva, D., Manegold, S., Koubarakis, M.: Geotriples: a tool for publishing geospatial data as rdf graphs using r2rml mappings. In: Proceedings of the 2014 International Conference on Posters & Demonstrations Track-Volume 1272. pp. 393–396. CEUR-WS. org (2014)
12. Nikolaou, C., Dogani, K., Bereta, K., Garbis, G., Karpathiotakis, M., Kyzirakos, K., Koubarakis, M.: Sextant: Visualizing time-evolving linked geospatial data. Web Semantics: Science, Services and Agents on the World Wide Web 35, 35–52 (2015)
13. Rahman, F., Slepian, M., Mitra, A.: A novel big-data processing framwork for healthcare applications: Big-data-healthcare-in-a-box. In: Big Data (Big Data), 2016 IEEE International Conference on. pp. 3548–3555. IEEE (2016)
14. Rodriguez, P., Haghighatkhah, A., Lwakatare, L.E., Teppola, S., Suomalainen, T., Eskeli, J., Karvonen, T., Kuvaja, P., Verner, J.M., Oivo, M.: Continuous deployment of software intensive products and services: A systematic mapping study. Journal of Systems and Software 123, 263–291 (2017)
15. Sebrechts, M., Borny, S., Vanhove, T., Van Seghbroeck, G., Wauters, T., Volckaert, B., De Turck, F.: Model-driven deployment and management of workflows on analytics frameworks. In: Big Data (Big Data), 2016 IEEE International Conference on. pp. 2819–2826. IEEE (2016)
16. Sezer, O.B., Dogdu, E., Ozbayoglu, M., Onal, A.: An extended iot framework with semantics, big data, and analytics. In: Big Data (Big Data), 2016 IEEE International Conference on. pp. 1849–1856. IEEE (2016)
17. Shearer, C.: The crisp-dm model: the new blueprint for data mining. Journal of data warehousing 5(4), 13–22 (2000)
18. Tsakalozos, K., Johns, C., Monroe, K., VanderGiessen, P., Mcleod, A., Rosales, A.: Open big data infrastructures to everyone. In: Big Data (Big Data), 2016 IEEE International Conference on. pp. 2127–2129. IEEE (2016)
19. Versteden, A., Pauwels, E.: State-of-the-art web applications using microservices and linked data. In: Maleshkova, M., Verborgh, R., Keppmann, F.L. (eds.) 4$^{\text{th}}$ Workshop on Services and Applications over Linked APIs and Data (SALAD). pp. 25–36. No. 1629 in CEUR Workshop Proceedings, Aachen (2016), `http://ceur-ws.org/Vol-1629/paper4.pdf`