This project has received funding from the European Union's Horizon 2020 research and innovation **programme under grant agreement No 780355**

FANDANGO DELIVERABLE

Deliverable No.:	D5.1
Deliverable Title:	Reference Architecture description
Project Acronym:	FANDANGO
Project Full Title:	FAke News discovery and propagation from big Data and Artificial iNtelliGence Operations
Grant Agreement No.:	780355
Work Package No.:	5
Work Package Name:	The FANDANGO software stack
Responsible Author(s):	ENG, Sindice, LIVETECH, VRT, CERTH, CIVIO
Date:	30.01.2019
Status:	V1.0 - Final
Deliverable type:	REPORT
Distribution:	PUBLIC

REVISION HISTORY

Version	Date	Modified by	Comments
V0.1	05.10.2018	ENG	First Draft
V0.2	16.11.2018	SIREN	Added SIREN contribution
V0.3	11.12.2018	ENG	Added CERTH, UPM and LVT contribution
V0.4	14.12.2018	SIREN	Peer Review
V0.5	18.01.2019	ENG	Revised version after Data Model (D2.2) and data processing pipelines update
V0.6	23.01.2019	SIREN	Peer Review
V1.0	30.01.2019	ENG	Revised and final version after peer review.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ABBREVIATION	DESCRIPTION
H2020	Horizon 2020
EC	European Commission
WP	Work Package
EU	European Union

1. EXECUTIVE SUMMARY

This document describes the reference architecture of the project and it has to be referred as main guideline for technologies in FANDANGO. The goal is to give to the reader a comprehensive overview of technologies implied in the project focussing on how they help to satisfy the requirements, motivations behind their choice and how their interaction is designed.

The definition of the architecture is crucial to determine how the system will support the workload and which tools are used to satisfy the requirements.

The project plans to monitor news producers websites so as to be able to determine the presence of fake news. This implies the need to have access of a possibly huge amount of data, since information on the web is created continuously and with high velocity.

The architecture design propose a big data architecture, with the intent of manage the ingestion, processing and analysis of data that traditional database can not handle. A big data approach is based on the parallelization of work on a cluster of computers allowing the system to handle the amount of data in an effective way.

In the world of big data, it is possible to choose between different software architectures depending on the workflow to be implemented. Whether the processes are analytical, batch, etl, streaming or that the response times should be near real time it is necessary to choose the correct architecture and consequently the tools for the implementation.

In the case of the functional requirements of FANDANGO, which will be analyzed in detail later, it is necessary to implement a streaming architecture that is able to process flows of data that come continuously from the web. In addition, the processed data must be accessed near real time in order to respond to journalists' queries. For this reason, over the computation layer, it has been prepared a presentation layer (Elastic) that contains all the data pre-processed and that is able to interact in near real time with the journalists. The storage is enriched with functionalities that allows to do advanced analysis like for affinity and similarity.

One of the pillars of the application is Kafka a distributed broker that allows to handle the stream of data and to give an access point to the back-end from front-end services, for example when journalist asks to crawl and analyse a specific news.

As mentioned in the previous paragraph FANDANGO is composed of a back end component, Big Data, and a front end application implemented through Angular that allows journalists to interact with the solution.

This model, associated with the use of big data tools, ensures not only efficient computation of large amounts of data, but also good horizontal scalability when the workload undergoes variations. This allows to achieve desired performance with adequate cost management.

We have chosen Cloud infrastructures to achieve the advantages described, since the virtual environment allows to manage the number of resources used based on their consumption. With this approach we are able to easily set up a hardware configuration to support the services and size the total capacity of the cluster in a flexible way. When computational needs are not longer required it is possible to shut down nodes of the cluster, saving their costs.

Requirements reported in deliverable "D2.3 User requirements" are transformed in technical functionalities and then matched with the best technologies to satisfy them. Following table reports the requirements associated with the chosen tool fulfil them.

REQUIREMENT	TOOL
Ingest data from web sites, claims, open data	Apache Nifi
Analyze news article to find information that can help user to detect malicious contents	Apache Spark, Apache Spark MLlib
Analyze videos or images for malicious modification detection	TensorFlow
Graph analysis for news correlation, information spreading	Neo4J
Visualization tool for statistics about current state of data in the system and graph visualization	Siren Investigate
Web application that permit user to retrieve useful informations about a specific article Claim verification tool	Python, Flask Siren Investigate

Table 1: Functional requirements and architecture mapping

Some non functional requirements necessary to handle the data trough the process are listed in the following table. This tools allows all the framework listed before to efficiently communicate and perform as expected.

REQUIREMENT	TOOL
Intermediate storage that allows stream computation	Apache Kafka
Final storage for fast data retrieving and comparisons	Elasticsearch
Raw data storage	Hadoop HDFS

Table 2: Non functional requirements and architecture mapping

We have also identified some non-functional requirements that allow the system to be more efficient and easier to manage, about scalability and portability of the platform.

REQUIREMENT	TOOL
System scalability	Cloud Microsoft Azure
System portability	Docker, Kubernetes, Azure

Table 3: Non functional requirements and architecture mapping

In this document its described a list of utilities available optionally for the partners, specific for team collaborations during the project life cycle.

UTILITY	SERVICE
Shared source-code repository for team collaboration	GitHub
Bug tracker and ticketing service for formal internal communications	GitHub issues tracker

Table 4: Collaboration tools

The architecture proposed and the list of frameworks described in this document can slightly change in order to handle problematic and needs that can rise during the implementations. However, the proposed solution has been designed in order to support additional use case and new features without substantial changes given the choice of easily customizable tools and the easy extensibility of the system.

2. INTRODUCTION

Fake news become one of the main threat of modern society, contributing into spreading misinformation, propaganda and hoaxes. FANDANGO is a project that aim to face this problem by offering a platform that allows to validate the reliability of information and sources found on the web. With this system it will be possible to identify sources on the web that make up malicious information, verify news considered suspicious and as final goal, stopping the spreading of this fake news.

There is a substantial effort to reach this purpose that requires the choice of specific tools to permit the finalization of the system, that have to support all desired functionalities. Underestimating the complexity of a requirement or chose not suitable tools to perform the operations may cause the failure of the project. Is indeed fundamental to understand the requirement of the project, evaluate the possible solutions and identify a set of software and frameworks to support the development of the whole product.

The reference architecture of the project is described in this document and it has to be referred as main guideline for technologies in FANDANGO. It explains the analysis done to chose the set of tools needed in the project and it describes their functionalities and usage in our use cases.

To achieve this results, user and software requirements were tracked and associated with possible technological solutions. After that, evaluations were made to define the final choice of the tools, evaluating theirs strengths and weaknesses.

FANDANGO's main goal is to offer to journalists a tool to fight fake news. Full description of this user requirements is described in the deliverable D2.3. It is possible to categorize this requirements into two different technical needs:

1. analyze a single news and gather all information about it
2. analyze statistics based on multiple articles and compare information

Some of the requirements belonging to group one are:

- see in one place all the informations provided by FANDANGO about a news item
- analyze the language of a news item

Some of the requirements belonging to group two are:

- receive a list of relevant sources regarding a claim or statement
- see articles related to the one given, and their estimated trustworthiness
- know whether the publisher of a news item is related to fake news in the past
- know whether the author of a news item is related to fake news creators
- find analysis (fact-checks) of a claim already done by reputed third-parties
- know when and where a fake news article originated, and how and by whom was propagated
- receive an alert when a fake news article trends

To achieve such goals it is necessary to monitor new information released on internet, both from reliable sites and not, to validate their content, classify if the news are real or manipulated and finally shows the requested information.

From a technical point of view there are three main modules to allows the final goal of FANDANGO:

1. Ingestion module: a set of tools and frameworks that permit the system to collect all data needed, like news, articles, claims, open data and images and videos
2. Advanced analytics module: analytics tools that allow to gather all information required from collected data and elaborate them through machine learning techniques to evaluate the content of the information and discover every aspect of the data that can help the user to identify the reliability of the content
3. Visualization and web application module: tools to interact with the platform and check the results of the analysis

The main problem that appear in the initial analysis is that in FANDANGO it is necessary to handle a substantial amount of data, increasing everyday, and in a variety of format (news articles, images, videos) in the form of structured and unstructured data. Since traditional database system are limited to elaboration of structured and limited data, a solution that required Big Data tools is needed.

A Big Data architecture is designed to handle the insertion, processing and analysis of data that is too large or complex for traditional database systems. Data can be processed in batches or in real time. Big Data solutions typically involve the use of a large amount of non-relational data, such as key-value data, JSON documents, or Time Series data (requirement specified into the deliverable "D2.1 Data lake integration plan"). Traditional relational database systems (RDBMS) are not suitable for storing this type of data.

The choice of the correct architecture to apply to the big data system is fundamental. Most common big data architectures are defined around the concept of batch and real time processing, and based on what is needed different, solutions are possible.

The elaboration can be done completely in batch mode, scheduling elaboration in an ordered pipeline of process. The main problem with this approach is the process is forced to stick to a scheduling order: ingestion then analytics and then visualization, causing the delay of information availability. This approach can not be suitable to permit real time notification of spreading fake news, also there is a delay on information gathering due of pipeline time consumption and scheduling frequency.

Two of the most common solutions to face these problems are "Lambda architecture" and "Kappa architecture".

Lambda architecture keeps separated the elaboration of batch and stream data, forcing different logic during the elaboration and merging of data from these pipelines. This approach is effective when is needed to have different access time to data, and it can be used the batch layer to execute massive elaborations and a speed layer where data is processed on streaming to allow fast access to the results.

Kappa architecture is a simpler version, where the batch layer is truncated in favour of stream processing. This allow to have a single logic on data processing and on information retrieval. This is the solution that best fit the requirements, since allows the information's update in real time and at the same time it keeps the logic as easy as possible, since batch processing is not required in the main computation of FANDANGO (https://www.oreilly.com/ideas/questioning-the-lambda-architecture). With this approach the system can ingest new data as fast as possible, keeping information updated. Having synchronized data may permit the platform to perform better in the analysis with better predictions and correlations with current topics. Also permit to check fake news state allowing the possibility of notification of spreading of malicious information.

The choice of the architecture will support all the needs of FANDANGO project at the current state and in futures steps. At the beginning we are only consider a part of the data sources that will be available in the final state, this mean that computability need will increase according with the number of data present in the system.

The architecture is composed by different tools that have to comply with the goals expressed. Tools belonging to the “big data family” are designed specifically to face problems with a huge amount of data in an efficient and scalable approach. This means that the architecture can scale easily by adding new machines to the infrastructure to cover an eventual increment of data to elaborate, without impact on actual software. This is a fundamental feature of this approach considering that data may increase consistently during the project’s life. The extensibility of the architecture will provide a reliable environment to work on without the need of change the logic of the software due to hardware/ framework modification.

In the first part of this reference is possible to find a more detailed description of the data flow in the system: how data is gathered, stored and analyzed and which tools are used for each step.

Afterwards it is described a detailed presentation of each framework, its usage in the system.

The final part of the document covers cross tools, useful for team collaboration during the project.

3. DATA FLOW OVERVIEW

This aim of this chapter is to give an overview of the system, focusing on data transformations, without going into implementation details. Each operation is associated with a framework deemed suitable for the purpose, a detailed explanation for each component is available in the following sections of the document.

To achieve the requirements express into User Requirements documentation and also during FANDANGO's meeting, we get input data from a selected list of web sites, verified claims sources and open data repositories. Web sites represent the list of information the project is monitoring, and are used to retrieve articles that are spread into the web. Open data repositories are databases of trusted information that will be used to validate articles and retrieve additional information for specific topics, like immigration or climate change. Claims are finally data that will be ingested by the partners to allow claim analysis and claim verification. This process is still a work in progress and decision about this data flow may change in following iterations of the project. Data is gatherer by web crawlers (or simply crawlers), software programs that browse specific internet web sites to retrieve their content. This is the ingestion module of the project.

Data from web sites need to be processed and analyzed in a pipeline, while open data and claims are stored directly into the final database since they don't need to be pre-processed.

We identify Apache Nifi (https://nifi.apache.org/) as the best tool to perform the ingestion operation. It offers a set of functions to design reach data flows that retrieves data from the web and also allows custom programs to execute in his components. This will be particular useful if there will be the need to change data sources or to allow the correct ingestion of data, since is possible to adapt current solution with a customized component.

Raw data pass through Apache Kafka (*https://kafka.apache.org/*), a queue based storage that allows data to be retrieved from different application. Data is accessible by each analytic component allowing to share data and information between the elaborations. Kafka is a fundamental tools to permit the kappa architecture and supports stream processing.

Once data is stored in the system a process will elaborate the information, categorizing the content. Raw data is divided extracting text, images videos and metadata, which will be managed accordingly. Metadata includes additional information like authors, source, publish time and so on. Kafka's storage is organized in topic, distinct queues for different type of data. Each type of information (text, image, video, metadata, claim) will be saved in related topic.

For data pre-processing we use Apache Spark (https://spark.apache.org/) a fast, general purpose elaboration engine. It supports parallel processing paradigm, this mean that it can perform processing a lot faster than standard sequential jobs.

After these phase, different process will elaborate the information, with the aim of finding relevant clue to help the classification of the data. Every data process depends on the type of the topic and requires a specific tool, choose together the specialized technical partner.

Text is validated semantically and syntactically, aiming to find grammar errors or recurring text patterns typical of fake news articles. Text analysis will be performed in Apache Spark since it also support advanced analytics operation, utilizing specific machine learning libraries (https://spark.apache.org/mllib/).

Video and images will be analyze to check for possible counterfeits and modifications. Image and video elaboration will use TensorFlow (https://www.tensorflow.org/) a Python library specialized into machine learning with state of art functions related to image recognition.

Metadata will be used for graph analysis, specifically for finding correlation between sources or author and determine if a specific web site is not reliable or an author is biased in his opinions or writer of fake informations. Metadata will be processed again in Apache Spark and then moved into Neo4J (https://neo4j.com/) for specific graph analysis. Neo4J is the de-facto framework related to graph computation and represent a solid and stable choice for the requirement. Spark graph elaboration framework GraphX (https://spark.apache.org/graphx/) is a possible alternative, but this particular library has some limitation that can prevent more advanced analysis.

Final data is enriched with extracted information from the processes and saved into Elasticsearch (https://www.elastic.co/products/elasticsearch). This tool allows fast access to data thanks to an advanced indexing and a search engine that allows full text queries, fundamental for claim verification and rapid user interaction with the database. From Elasticsearch data made available for final user to be interrogated and satisfy user requirement specified.

Siren Investigate (https://siren.io/) will allows the final user to have access at data statistics and information graph. With this tool final user will be provided with the necessary information that allows to check the propagation and the correlation of fake news or a particular topic, and also be alerted of spreading of fake news.

FANDANGO web application is the other visualization tool provided by the project. User will use the web app to investigate reliability of specific articles with the possibility of selecting part of the text and verify a particular claim. These features will support the final user during the analysis of a specific news, providing all information extracted from the analysis with additional advice retrieved from analytical computation of historical data. Web application is developed in Python with Flask framework (http://flask.pocoo.org/). This choice is due partner's framework preference and represent a solid choice for service development, with no limitation for project requirements.

The architecture will make available support tools for future or development utilization. Apache Hadoop (https://hadoop.apache.org/), provides the system with a set of useful frameworks, like HDFS. In addition will be also available Apache Zeppelin (https://zeppelin.apache.org/), a tool for data visualization that has been required from partners for testing and visualization purpose. Another possibly really useful tool is Apache Oozie (http://oozie.apache.org/), a scheduler that permit to organize jobs in the system, like data cleaning to reduce memory consumption or periodical batch job needed to improve the performances of machine learning models.

Finally Docker (https://www.docker.com/) is a framework expected to be present in the system for containerisation of web application or micro-services that will be used in the system. This will be fundamental to ensure the portability and extensibility of the system. For container management we consider Kubernetes (https://kubernetes.io/) an orchestration tool that automate the deploy of docker images. This will simplify the administration of the cluster by monitoring the status of services and re-deploying them in case of failure.

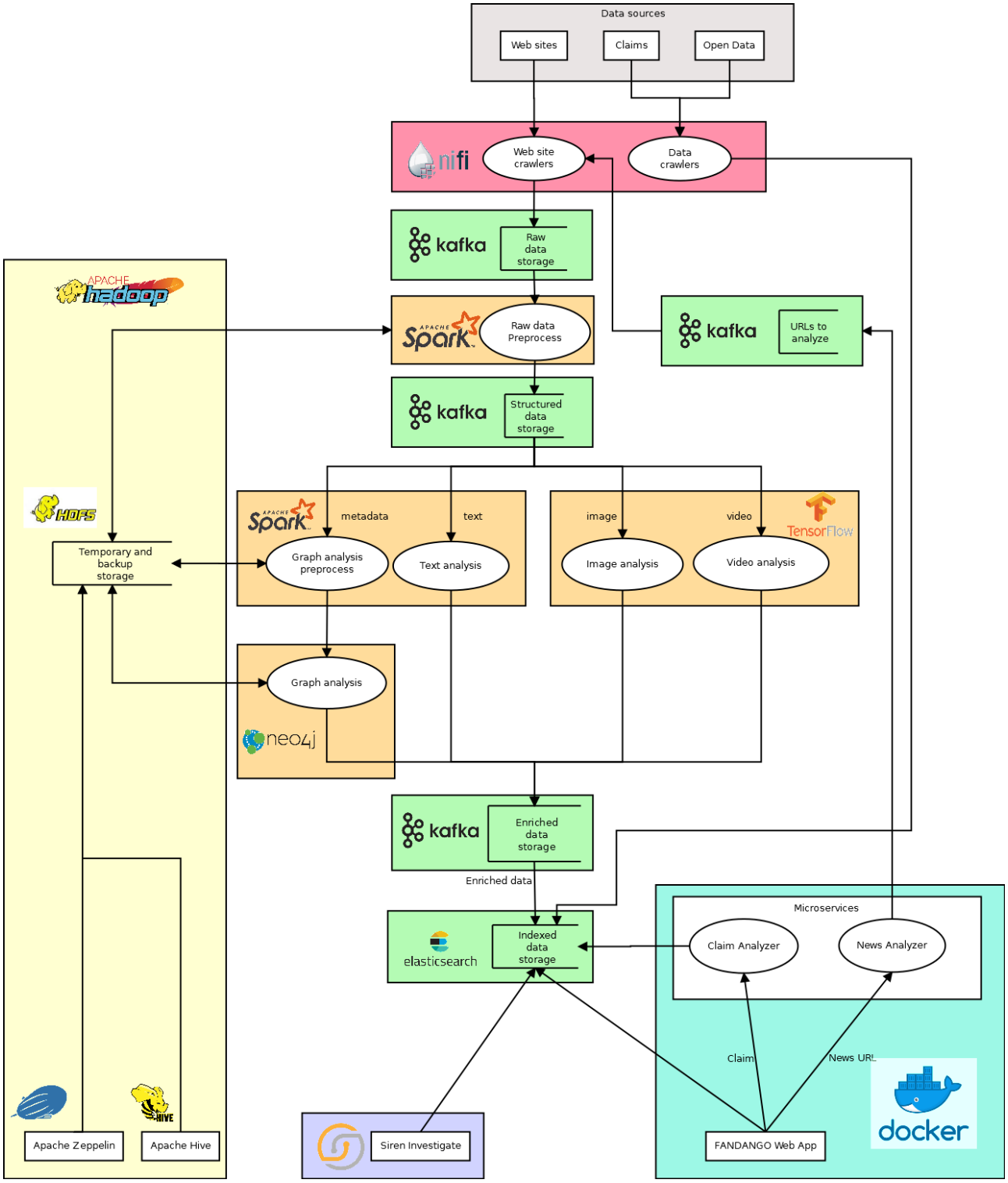

Figure 1: Data flow diagram

4. ARCHITECTURE COMPONENTS OVERVIEW

The Big Data infrastructure implemented for the FANDANGO project uses the Hortonworks suite, one of the biggest Big Data platforms currently on the market together with Cloudera and MapR.

The choice fell on the Hortonworks suite thanks to the exclusive presence of open-source tools, a difference in competing solutions. The HDP platform offers a wide range of tools for Hadoop Administrator to manage and monitor clusters, integrated in the Ambari web application. In addition Hortonworks is well integrated in several public clouds.

If the monitored services require more computational resources in terms of RAM or CPU, it is possible to scale them in order to meet the required performance requirements. The scalability of HDFS storage (Hadoop Distributed File System), on the other hand, is completely linear.

This flexibility is also made possible thanks to the fact that the architecture is deployed on the public cloud Azure. It is important to underline that all the tools used by FANDANGO are independent from the chosen cloud provider to guarantee portability as required by the non-functional requirement "architecture should be replicable and easy to maintain". Using a cloud infrastructure the system can be scaled up and down in order to handle peaks of usage or of data ingestion.

Hortonworks provides various security solutions, data governance and information security in the proposed architectural solution.

To provide all the functionalities of the project we need to extend the processing power of Hortonworks suite (including Spark and Kafka), and for these reasons FANDANGO architecture contains other tools. In particular the Siren Investigate suite that acts as storage for news and claims, but also as search engine (thanks to Elastic) and analytical tool and is fundamental for the requirement “analyze statistics based on multiple articles and compare information” defined in D2.3

To perform the sophisticated metadata analysis expected by T4.4 activities - “Source credibility scoring, profiling and social graph analytics”, such as the relations between fake news, authors, other news, the FANDANGO platform is empowered by Neo4J, a graph analysis DB. (for the user stories “know whether the publisher of a news item is related to fake news in the past, know whether the author of a news item is related to fake news creators, know when and where a fake news article originated, and how and by whom was propagated”).

In the following chapters we’ll illustrate the main components and their usage inside the platform. The data models is described in D2.2.

SOFTWARE	DESCRIPTION
NiFi	Data flow ingestion tool, open source, distributed and scalable, to model real-time pre-processing workflow from several different sources. It hosts the crawlers.
Kafka	Publish-subscribe distributed messaging system, that grants high throughput and back pressure management. This is the tool FANDANGO uses to connect the different components
Spark	Fast, in-memory, distributed and general engine for large-scale data processing with machine learning (Mllib), graph processing (GraphX), SQL (Spark SQL) and streaming (Spark Streaming) features. This is the processing engine of the project for pre-processing and text analysis.
TensorFlow	TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. This is the engine used to analyze multimedia data.
HDFS	The Hadoop distributed file system, open source, reliable, scalable, chosen as storage. Used in training phase and eventually for historical storage.
Hive	Query engine (SQL-like language) on HDFS (and HBase) with JDBC/ODBC interfaces. Used to read data from HDFS.
Oozie	Workflow scheduler. Used to schedule retraining of models.
Ambari	it acts as both a workflow engine and a scheduler. In this case, its main role is to manage the scheduling of Spark jobs and the creation of Hive tables.
Siren	Investigative Intelligence UI with connectivity to Elasticsearch, whose aim is to allow reporting, investigative analysis and alerting to users based on the indexed contents.
Elasticsearch + Siren Federate	Distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. Siren Federate plugin is added to Elasticsearch to allow data set semi-joins and seamless integration with different data sources.
Rest APIs, RSS, Web Sites, Open Data, Social network	Data sources of the FANDANGO project. Specific crawlers will connect to these sources of data to get the information needed to verify the news.
Zeppelin	The notebook dedicated to data scientists, to run in REPL mode scripts and algorithms on data stored in Hadoop.
FANDANGO Webapp	Access point to FANDANGO Infrastructure. The journalist will use the FANDANGO Web application to insert news and verify the fakeness of certain

	publications.
Docker	Docker is used to run software packages called "containers". Containers are isolated from each other and bundle their own application, tools, libraries and configuration files; they can communicate with each other through well-defined channels. Used to package the FANDANGO web app.
Kubernetes	Kubernetes is an open-source container-orchestration system for automating deployment, scaling and management of containerized applications

Table 5: FANDANGO Architecture's main components

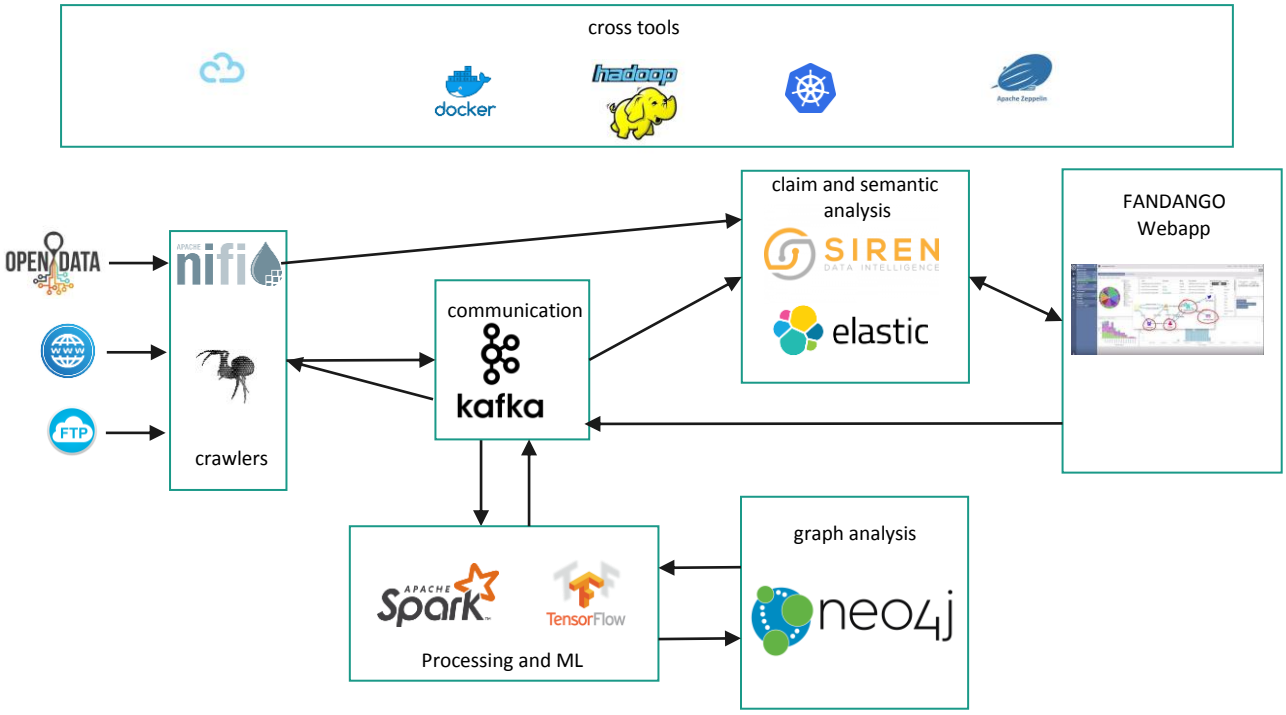

Figure 2: FANDANGO Architecture's main components

4.1 HADOOP

Hadoop is an open source distributed processing framework that manages data processing and storage for big data applications running in clustered systems. It is at the centre of a growing ecosystem of big data technologies that are primarily used to support advanced analytics initiatives, including predictive analytics, data mining and machine learning applications. Hadoop can handle various forms of structured and unstructured data, giving users more flexibility for collecting, processing and analyzing data than relational databases and data warehouses provide.

4.1.1 HADOOP USAGE IN FANDANGO

Hadoop technology is needed to support Big Data and distributed applications since this framework enables to elaborate and handle any kind of unstructured data, using a lot of nodes and huge amount of

information. Hadoop provides a stack of computational resources able to elaborate lot of petabyte through HDFS.

Within the FANDANGO project, several tools from the Hadoop ecosystem are taken in consideration. NiFi, Spark and Kafka are the heart of architecture, but Hadoop contains other tools that can also be used in a second phase. Among these we mention the administration tools of the platform as Ambari or Oozie for the management of workflows.

In the following chapters, the use case for each product of the Hadoop ecosystem will be detailed.

4.2 KAFKA

Kafka is a scalable messaging system, also included in HDP, which allows to manage the delivery of information in publish / subscribe mode thanks to distributed, redundant, resilient and with automatic recovery characteristics. This system allows to manage the back-pressure and high throughput towards the components that act as subscribers, guaranteeing the availability of data in the event of a fault. Kafka's consumers do not remove the events from the queue, which is instead saved until the end of the set retention period. The data are transmitted in key-value format, divided into replicated partitions and divided by topic on different brokers.

4.2.1 KAFKA USAGE IN FANDANGO

As mentioned in the introduction paragraphs, FANDANGO has an architecture stream oriented, as it constantly captures and processes the data flow captured by the crawlers. Kafka is the component that coordinate all the Data Flow phases, in particular Kafka allows the ingestion, processing and storage tools to communicate via a distributed and reliable queue. This ensures high performance, component decoupling and reliability.

As defined in the majority of the user stories of the D2.3, the web interface should interact with the crawlers and the processing components in order to analyse news never downloaded in previous iterations. Kafka will play the role of entry point for all the communications between front and back end components. In this way we'll decuple the presentation logic (javascript and REST) from the processing part.

In order to implement the Data Flow, as described in Figure 1, we create several Kafka topics, one for each type of data. We can group the topics in these categories

- Topics for raw data of type news
- Topics for raw data of type open data
- Topics for the pre-processed data (multimedia, text, metadata)
- Topics for processed data (after execution of models)
- Topics for the realtime interaction with the GUI

The main Producers of the Kafka instance are NiFi for the crawled data, and Spark streaming that gets the topics, apply processing and pushes the results again in Kafka.

On the other hand the main Consumers are Spark streaming and Siren.

4.3 NiFi

Apache NiFi is born of the NSA (National Security Agency - USA), and subsequently it is made open source. This component has the task of collecting data from different sources in a Data Flow mode in distributed and scalable mode. Apache NiFi offers several processors that can be modelled in specific workflows, using a graphical tool in drag and drop mode, in order to acquire data of various kinds, format, scheme, protocol, speed and size (for example, data from sensors, geo location devices, click streams, files, social feeds, log files and videos etc.). NiFi can upload data to Hadoop, but can also send it to other output applications, thanks to a variety of output processors.

The figure shows an example of a UI of NiFi, with a fictitious workflow:

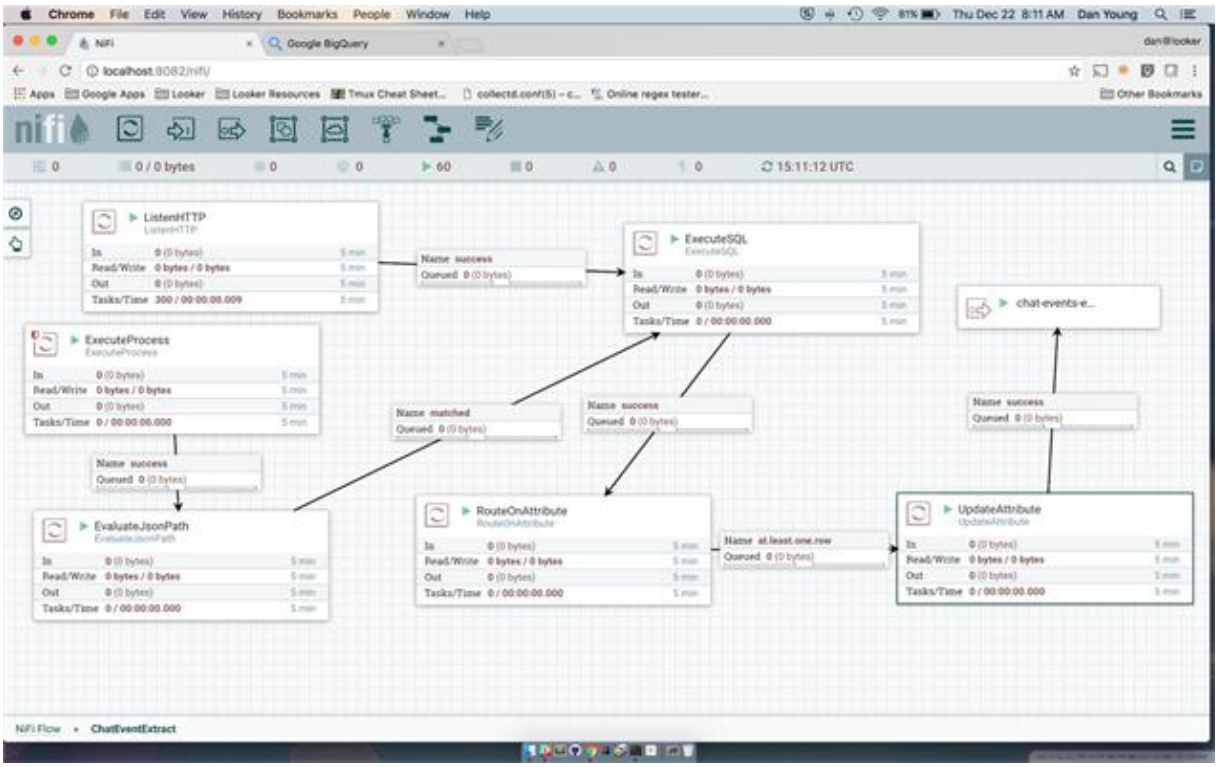

Figure 3: NiFi user interface

During the execution of these acquisition workflows, the data can be processed and pre-printed in real-time. The NiFi processors are customizable, and it is possible to code any missing connectors or intermediate processing components. Fault tolerance is also guaranteed, without loss of data and the ability to trace the data source and error handling.

NiFi can transport the data scheduling the processes of ingestion, but being always up & running also allows to set long running logic listening to certain events. The processes managed by NiFi are not limited to those of data acquisition but thanks to these characteristics, they can also be used to trigger jobs in the data lake or to transfer information to the outside world.

The following figure shows the architecture of NiFi and the possibility of being performed on multiple instances to cope with large volumes of data to be processed, through the coordination of Zookeeper (tool included in HDP for the coordination of various services such as Kafka).

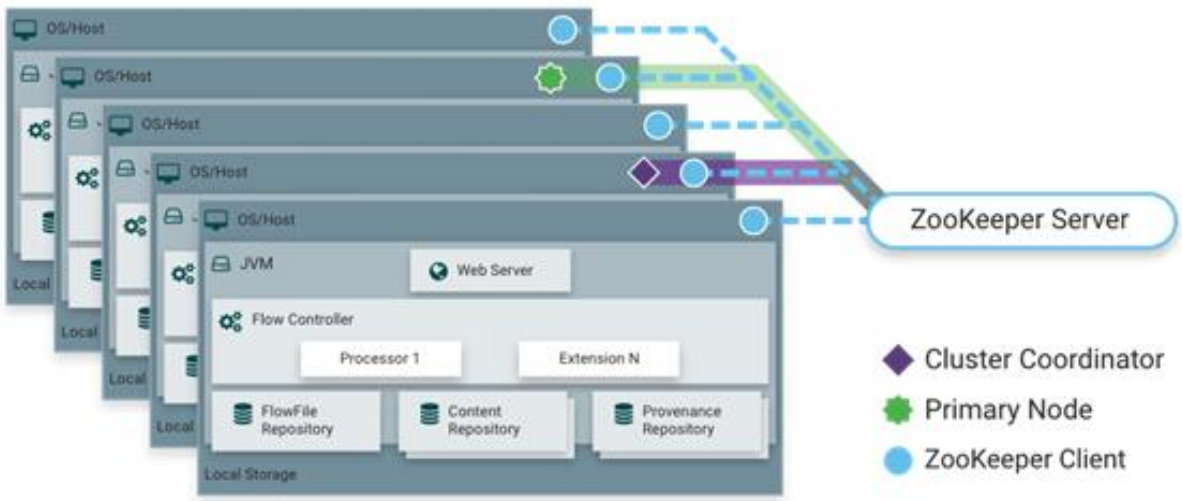

Figure 4: Distributed architecture of NiFi

NiFi is based on the concept of Flow Base Programming (FBP), a programming paradigm that defines applications as "black box" networks that exchange data through predefined connections through the passage of messages, where the connections are specified externally to the processes. These "black box" processes can be reconnected endlessly to form different applications without having to be changed internally. The FBP is oriented to the component (component-oriented), where the component can be reused

4.3.1 NiFi USAGE IN FANDANGO

NiFi is the main technology used for ingest data into the cluster. We've chosen NiFi instead of other products like Flume because it is not only an ingestion tool, but it is a data logistic platform. It allowed us to define complex workflows for the crawling of the data from different sources such as web sources and open data as defined in D2.3..

The scrappy crawlers have been implemented using ExecuteProcess processors that calls python procedures. Each crawler returns a document, in JSON format, that contains all the articles that were fetched in a iteration. The document will be than forwarded to a SplitText processor that creates a single document for each article crawled.

The NiFi user interface let to schedule the execution of the components and to configure the process allowing the administrator to manage the data ingestion.

4.4 CRAWLERS

One of the main focuses of data acquisition task in FANDANGO is data discovery and collection from social media streams, websites, open data and linked content. Although this task is conceptually easy, information extraction requires a potent implementation for discovering, acquiring, combining and handling data from the various sources, since some data can be superfluous and new content is generated with increasing velocity. More importantly, the majority of the news sites don't follow any data schema for easy parsing of their content and thus a laborious work is required.

A website crawler (also known as bot, web spider or web robot) is a computer program that systematically browses the website for purposes of indexing. Essentially, website crawlers are used to collect/mine data

from the Internet. Most search engines use it to see what is new on the Internet. A website crawler visits a website, extracts information from it and saves it for later use. Scheduled data retrieval from a website can be realized through software tools that take a URL as input, read through the website's code and store extracted data into the data lake.

For the purposes of FANDANGO, we selected Scrapy to implement the Website crawlers. Scrapy is a well-known open source web-scraping framework for Python programming language. It comes equipped with a wide range of features for working with proxies, cookies and authentication right out of the box. This allows writing very powerful web crawlers with minimal amount of code. However, it cannot interact with dynamic content in a website or execute Javascript code.

Scrapy is based on Selectors. A Selector "selects" certain paths of the HTML document specified either by XPath or CSS expressions. When working with Scrapy, what to get after crawling (item) must be specified. Extension mechanisms for Items provide additional protection against populating undeclared fields and preventing typos.

Once the model has been created, a "spider" which controls the "crawl" action has to be implemented. Spiders are classes that Scrapy uses to scrape information from a domain (or group of domains). They define an initial list of URLs to download, how to follow links and how to parse page contents to extract items.

4.4.1 CRAWLERS USAGE IN FANDANGO

The crawlers are used to ingest data in the FANDANGO data lake for further analysis. Different crawling scripts will be delivered for different sources so that different structure of information can be captured.

Currently, the identified sources are the Twitter streaming service, a wide list of web sites and news sites, as well as a list of open data portals in European and National level.

The Twitter social media platform has emerged in recent years as one of the de facto channels of news announcements from major government and political parties in both European countries and the Americas. It is also one of the biggest platforms for both valid and fake news propagation. The FANDANGO project will include crawlers of both the Twitter APIs, for monitoring Twitter streams of news agencies, politicians and political parties for the detection of emerging news and the validation of already detected events. Additionally, Twitter streams will be sampled for the detection of new sources to be crawled.

The web crawling In FANDANGO can be used for crawling specific news websites in order to collect published news events. Additionally, known satirical websites and fake news sites can be crawled in order to detect if a news item is a previously detected hoax. For this purpose a list of websites has been collected consisting of over 200 sites. Each one of these sites is crawled in a periodical basis for archived and new content. The content of each page is parsed in order to extract useful features including the title and full text, all images and video that might be included, the author of the article and the dates that the article was published and last edited.

4.5 SPARK

Spark consists of a large-scale in-memory distributed processing framework, able to take advantage of the data locality features on the Hadoop cluster. It is encoded in Scala, Java or Python and offers libraries for real time processing, with Spark Streaming, as well as SQL modules, graph processing with GraphX and various machine learning algorithms distributed through MLlib. Spark can be coded in Java, Python or Scala, even if it's internals are written in Scala which is a functional language compiled on the JVM, that also allows the use of Java libraries (although confined to the scope of non-distributed operations). Since Scala

is the native language for Spark, a factor that affects performance, if you use datasets distributed through RDD (Resilient Distributed Dataset) constructs. For data to which a scheme is applicable (fixed rows and columns), it is advisable to use the DataFrames, which further increase the speed of the processes.

Spark can simultaneously access data online (primary, preferential behaviour) natively by loading in memory the data partitions hosted in HDFS, on the same on which the task is executed, but it will also be able to access data stored in external repositories (via APIs or JDBC driver).

4.5.1 SPARK USAGE IN FANDANGO

Along the project, SPARK will be used in two main stages:

- Pre-processing,
- Development of Machine Learning (ML) and Deep Learning (DL) algorithms.

Firstly, once the data has been extracted from the different URL's using Crawlers, a pre-processing step is required to clean and prepare the data in a proper format to be used later, by the modules in charge of developing ML and DL models. Therefore, in the former case, the input of this module will be a large JSON file with several information collected from a certain URL. Subsequently, SPARK will be employed to read the data and to clean it due to its powerful properties regarding scalability and flexibility. Finally, SPARK will be employed to write back to Kafka the specific information required for the following modules of the system in different files. For instance, a message file will contain the data regarding the media content whereas another message will contain the semantic information of the article such as the text, the title or the keywords. Besides, all these files will have in common the same index that will be associated to a piece of news.

Secondly, SPARK provides a powerful library MLlib where several Machine Learning algorithms can be applied and trained in large scale databases such as FANDANGO's. Therefore, it will be used as a tool not only for pre-processing the data but for developing predictive models. Moreover, it offers the possibility of developing in other programming languages such as Python, Java and Scala so that, it can be adapted regarding the necessities of developers.

Another component very important for the project is Spark Streaming and the integration with Kafka.

Kafka is a potential messaging and integration platform for Spark streaming. Kafka, Indeed, acts as the central hub for real-time streams of data and are processed using complex algorithms in Spark Streaming. Once the data is processed, Spark Streaming could be publishing results into yet another Kafka topic or store in HDFS, databases or dashboards. The following diagram depicts the conceptual flow.

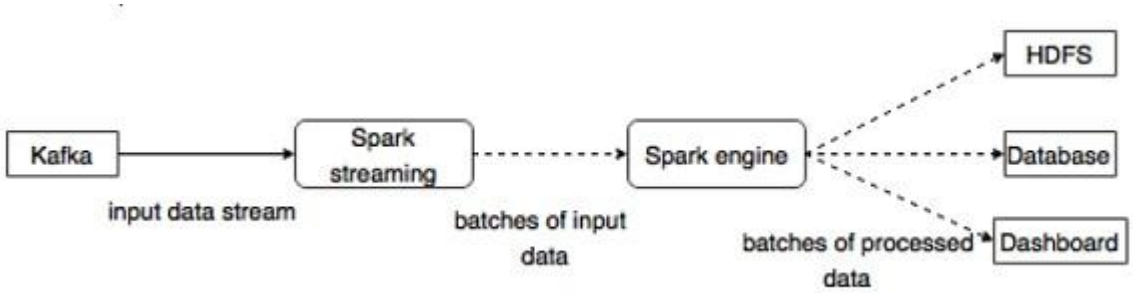

Figure 5: Spark Streaming and Kafka relation

In our case Spark engine will use the MLlib in ordered to apply the algorithms and the result will be stored again on Kafka or to the datastores.

SPARK can be integrated easily in a Big Data Platform such as FANDANGO where different components need to be constantly communicating among each other and processing large amount of data in real time.

4.6 HDFS

HDFS stands for Hadoop Distributed File System and it's the native Hadoop storage, consisting of a file system distributed across all cluster data nodes, which insists on the local Linux file system of each node. The features of this storage are to be append-only (HDFS is referred to as "immutable file system"). In fact, operations to modify the files stored in it are not allowed. On the other hand, HDFS guarantees high throughput for full-scan I / O operations. The files saved on HDFS are partitioned into blocks of fixed size and redundant on the various DataNodes and thanks to YARN the Spark and MapReduce processes (Yarn-compliant) are able to manage data in a distributed way, on each partition. The architecture of the HDFS, where the NameNode holds the information about the metadata of the stored files (in particular where the partitions and the replicas are located), implies that HDFS works in the most efficient way on large files (larger than block size-128 MB).

The storage deployed to host data is mainly HDFS. The data stored in HDFS are divided into special folders with permissions managed by ACLs for each partner/use-case and can be compressed thanks to the use of different formats: in this case the Avro formats will be adopted (when access to all rows and columns is required) and ORCFile for those data that prefer analytical access (only some columns) and that will be made available through Hive, whose reading performances on ORC files are optimal.

The types adopted can be traced to row-based and column-based formats:

- Avro (Row-based): characterized by having the schema of the records embedded in the file. Therefore, there is no need to pass the schema in the passage of files between applications. Avro also encodes data types based on type (not encoding as text) making it possible to compress the file more efficiently. This greatly reduces the bandwidth used and the processing time of the file. Another feature is to be evolving-schema, that is, using Avro, it is very easy to extend the existing schema by adding new columns without having to rewrite existing data. The applications that will read the data with the new schema, will still see the old records but will not have the value of the new column (on-read scheme).
- ORC (column-based): ORC is a self-descriptive columnar file format designed for Hadoop. It is optimized for large streaming readings, but with integrated support to quickly find the required lines. Storing data in a columnar format allows you to read, decompress and process only the values required for the current query. Because ORC files are type-aware, you can choose the most appropriate encoding for the type and create an internal index when writing the file. Thanks to the ORC type, (advanced compression, column projection, predicate push down and vectorization support) Hive is made more powerful than any other format on HDP.

It's nevertheless important to highlight the possibility to save encrypted data with transparent encryption at-rest of HDFS, to optimize security and never to keep any sensitive data in clear. This feature can be especially useful whether treating sensitive data or data having privacy issues.

This approach, encryption of HDFS at rest, encrypts selected files and directories stored ("at-rest") in HDFS using specially designated HDFS directories called "cryptographic zones".

Apache Hive is a SQL engine used to create metadata with a SQL-Like engine in order to access the data that resides in HDFS.

Hive, originally created and used by Facebook, was designated as the Hadoop Datawarehouse, since it's capabilities to perform operation to read files located on HDFS, by means of SQL-like code. Its main purpose is to read data, using query, therefore it uses a mapping between the table and the underlying file, that is memorized in a relational DB called Metastore (HCatalog).

Hortonworks Data Platform (HDP) supports Hive LLAP, which enables application development and IT infrastructure to perform queries that return real-time or NRT results. Use cases for implementing this technology include environments in which users of Business Intelligence (BI) tools or Web dashboards need to accelerate the analysis of data stored in an EDW Hive.

Hive in HDP 2.6 supports the MERGE SQL statement, which simplifies data refresh, data deletion, and data acquisition changes between tables. Transactional operations can be implemented through Hive QL, thanks to the recent feature of Hive on ACID, thus allowing to operate also in upsert on HDFS.

4.6.1 HDFS USAGE IN FANDANGO

The main Datastore in FANDANGO project is Siren/Elastic since it contains the processed news, the open data and the claims. HDFS is used from Spark and Kafka to store intermediate data.

In addition HDFS is used to store raw. This is well known Big Data design pattern because it allows to save the pure datum and consequently to have a complete history not affected by pre-processing. In this way, historical data can be reviewed in the future, analyzing them on different points of view.

Hive is used by the developers in order to read the raw data from HDFS and from the models in order to be retrained. Since Hive is the warehouse of Hadoop is easy to link to BI tools so it is a good extension point for future extensions of the platform.

4.7 ELASTICSEARCH & SIREN

The Siren platform provides relational cross index and cross system capabilities and investigative intelligence features, using Elasticsearch as its primary backend solution.

The platform comprises three components.

- Siren Investigate: A browser-based visualization tool that provides powerful graphical and analysis capabilities.
- Siren Alert: An alerting and reporting component for operational notifications and information.
- Siren Federate: A back end that provides the ability to search across Elasticsearch, Hadoop, and SQL databases.

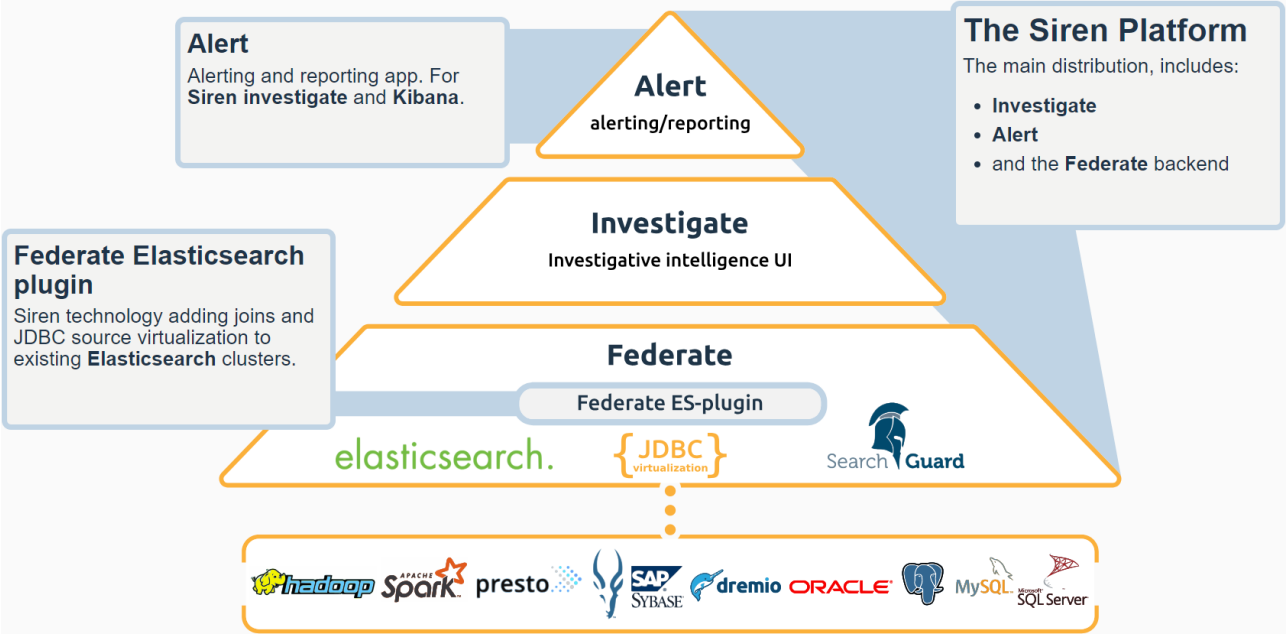

Figure 6: Siren architecture

4.7.1 SIREN INVESTIGATE

Siren Investigate is an application for interactive, exploratory investigative intelligence. It ties together the big data that resides in your infrastructure, without moving it, using a semantic data model that defines how data connects.

You edit the data model in Siren Investigate by listing the tables in the remote data sources, such as an RMDBS, an Elasticsearch index, or an Impala table, and how these refer to each other in a similar way to a primary/foreign key definition, but across systems.

You can cross boundaries of indices and back ends to discover how events and entities are connected. And you can use relational filtering to see time placements of events related to both single entities and groups.

Enterprise-level access control is provided at the index, record and field levels. And end-to-end encryption exists from the user interface down to the inter-cluster communications.

4.7.2 SIREN ALERT

Siren Alert provides alerts and reports (PDF emails) to be generated using logic, ranging from simple queries to advanced Complex Event Processing (CEP) scripts. The Siren Investigate user interface enables you to configure watchers (email recipients) and reports.

The Siren Alert scripting capabilities make it easy to implement statistical anomaly detection methodology to determine when alerts should be generated.

4.7.3 SIREN FEDERATE

Siren Federate is a plugin extension for Elasticsearch that adds cluster distributed and highly optimized cross index and cross back end data joins.

These capabilities are exposed by Siren Federate as an extended Elasticsearch API that is backward compatible with the Elasticsearch and Kibana plugin ecosystem.

Siren Federate makes full use of your current systems with the ability to translate analytic and join queries to the language supported by your existing databases and big data infrastructure or transparently using in-Siren-cluster-nodes memory joins as required.

4.7.4 ELASTICSEARCH

Elasticsearch is an open-source, RESTful, distributed search and analytics engine built on Apache Lucene. Since its release in 2010, Elasticsearch has quickly become the most popular search engine, and is commonly used for log analytics, full-text search, security intelligence, business analytics, and operational intelligence use cases.

Elasticsearch offers simple REST based APIs, a simple HTTP interface, and uses schema-free JSON documents, making it easy to get started and quickly build applications for a variety of use-cases. Moreover, the distributed nature of Elasticsearch enables it to process large volumes of data in parallel, quickly finding the best matches for search queries. Operations such as reading or writing data usually take less than a second to complete and enable real-time use cases such as news investigation.

4.7.5 ELASTICSEARCH & SIREN USER AND USAGE

Sometimes, the answer to a specific question may be simple, but the investigative process is typically complex and unique each time. While business intelligence, enterprise search, and knowledge graph are useful for specific tasks, investigative intelligence combines these elements to answer new questions. Siren platform provides a fully-fledged investigative platform, enabling users to move faster and go further when searching for content validation or evidence of false claims and misinformation.

In FANDANGO, Siren platforms is used to deliver fluid investigation capabilities to users trying to identify the trustworthiness of articles and claims. The main functionalities based on Siren platform are:

- Analytics about news articles with full text search capabilities and dashboarding, providing contextual analysis and drill-down mechanics to users investigating a group of articles.
- Analytics about claims with full text search capabilities and dashboarding, providing contextual analysis and drill-down mechanics to users investigating certain claims.
- Analytics for open data with full text search capabilities and dashboarding to improve user access to official information.
- Identification of related articles and claims that will support users by providing a bigger picture of the current scenario they are investigating.
- Knowledge-graph visualization to quickly identify related subjects, inter-connectivity, relevance, geographical spread and time-line progression of all aspects concerning article and claims publications.

While Siren platform does offer direct connectivity to multiple data sources for real-time analysis, given the search and relevance analysis requirements needed for FANDANGO, all the analytical data will be stored in Elasticsearch. Siren platform will connect to Neo4J Graph Platform but only to consolidate information in Elasticsearch before it can be fully used. Therefore, the are no other data flows required by the platform other than the data published by other systems being available in the Elasticsearch database.

Given the analytical nature of the platform, there will be no direct input from user through Siren platform, as those will be done through FANDANGO's purpose-built user interface.

4.8 ZEPPELIN

Apache Zeppelin is an open-source web-based notebook, adopted as Data Science Platform by Hortonworks, which allows you to perform operations of:

- Data ingestion
- Discovery of data
- Data analysis
- Data visualization and collaboration

Zeppelin benefits of a wide variety of interpreters (see the https://zeppelin.apache.org/documentation) and languages including Spark, performed using a specific Spark Driver called server.

The UI provides the ability to execute in REPL mode fragments of Spark code in various languages (Scala, Python, R) and display the results with different graphical widgets. The creation of algorithms can be done by importing special machine learning libraries into the code snippets that can be extracted from UI (example MlLib for Spark). Livy interpreter (Rest Service for Apache Spark https://livy.incubator.apache.org/) can be used to invoke Spark services, thus decoupling the front end from the backend.

4.8.1 ZEPPELIN USAGE IN FANDANGO

Zeppelin allows developers to interact with the FANDANGO data storages and to write algorithms in python, Scala or even in Hive QL. In this way partners can perform data exploration and, through the Python and Scala interpreters, write new algorithms and test them on subsets of data.

Zeppelin can be used not only by FANDANGO development team, but also by FANDANGO end user data scientists. In fact, data scientists can refine the algorithms, set parameters and test them among production data. Furthermore Zeppelin has a profiling system that allows access only to portions of controlled and certified data.

4.9 NEO4J

Neo4j is an open source graph database software developed entirely in Java. It is a totally transactional database, developed by Neo Technology, a startup of Malmö, Sweden and the San Francisco Bay Area.

The graph structure of Neo4j is extremely convenient and efficient in dealing with structures such as trees extracted for example from XML files, file systems and networks, which are obviously represented by a graph. Exploring these structures is usually faster than a table database because the search for nodes in relation to a certain node is a primitive operation and does not require multiple steps, usually three implicit in a SQL join, on different tables. Each node contains the index of incoming and outgoing relationships from it.

Neo4j does not support data sharding, so the entire graph must be stored in one machine. On the other hand, multiple instances of the same database can be activated simultaneously on different machines, creating a master-slave architecture managed through Zookeeper.

4.9.1 NEO4J USAGE IN FANDANGO

Neo4j will be the core in the graph analytics module, where two main operations will be performed including create the Graph Database and update it.

First of all, to create the Graph Database, NEO4J will load the metadata of the news, in order to map the database with the ontology of FANDANGO and generate the graph database. To do that, NEO4J provides its own query language named as CYPHER which allows developers to create optimal and intuitive queries by applying different operations in a similar way as SQL queries. Moreover, in the case of updating the Graph, NEO4J will also employ CYPHER to solve that purpose in an effective way.

Secondly, NEO4J will be used to perform a graph analysis regarding semantic relationships among organizations, authors, articles or publishers with the aim of measuring the credibility of these items based on its influence in the fake news propagation. To do so, NEO4J provides different plugins such as APOC and libraries to work with graph algorithms but for large-scale environments such FANDANGO is, where a graph may have millions of nodes and relationships and other products are not feasible to perform the analysis in the required time. Furthermore, the NEO4J community is also developing specific ontology and procedures to help developers and researchers in the task of detecting fake news using graphs by investigating for instance, communities of suspicious organizations or domains.

In Figure 8, a visualization of a graph database is presented, where three different items are presented including: authors (blue), domains (purple) and articles (yellow). The ontology presented in the example is a simplified version of FANDANGO's ontology to show how NEO4J can be used to detect fake news based on the semantic relationships between the different elements of the graph database.

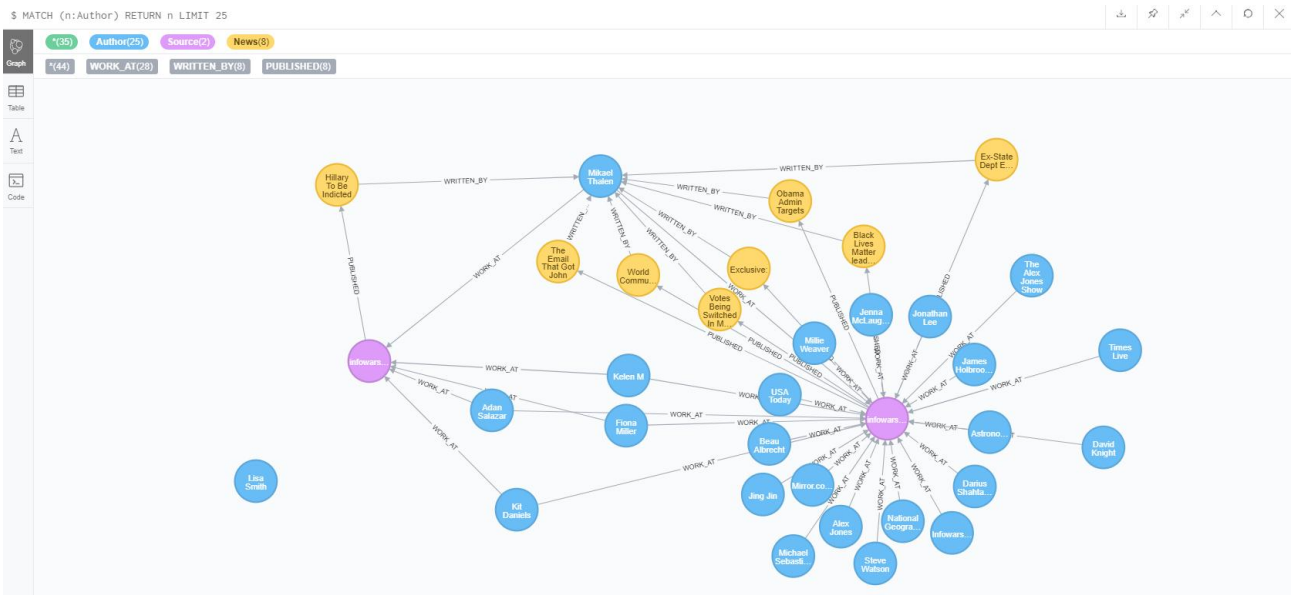

Figure 7: A graph database representation

Moreover, although NEO4J is implemented in JAVA, it provides a set of drives in other languages such as Python, C# or Javascript with the aim of encouraging developers to make use of this tool in Big Data solutions. In the case of FANDANGO, a python driver named as Py2neo will be used to perform the analysis and to obtain the source credibility measure for the different items involved in the graph.

Finally, NEO4J provides its own web application that will be used only in a development environment to check that the module works as expected and also, to review and solve possible errors in CYPHER queries. However, the data visualization will never be performed by the NEO4J web application but by FANDANGO's web application.

4.10 WEB APPLICATION

The aim of the FANDANGO web application is to show to the final user, the journalist in this case, the results of the analytical processes described in the document (from all the requirements but in particular "see in one place all the information provided by FANDANGO about a news item").

According to the iterative prototyping methodology adopted in FANDANGO to carry out research and development activities, the first implementation of the Web interface of FANDANGO is composed by a front end application written in Angular and a back end that support the GUI via REST services. In particular:

- Back-end side is developed in python, using Flask to provide services available on the web interface to answer final users requests.
- Given an URI, a flask request returns contents of the article from the URI (title, body and source)
- Front-end side is composed by an interface developed in Angular, that uses Ajax requests to send queries to API s. The exchange of data comes off through JSON object.

This is a web interfaces mock-up: therefore, some tools can change during construction.

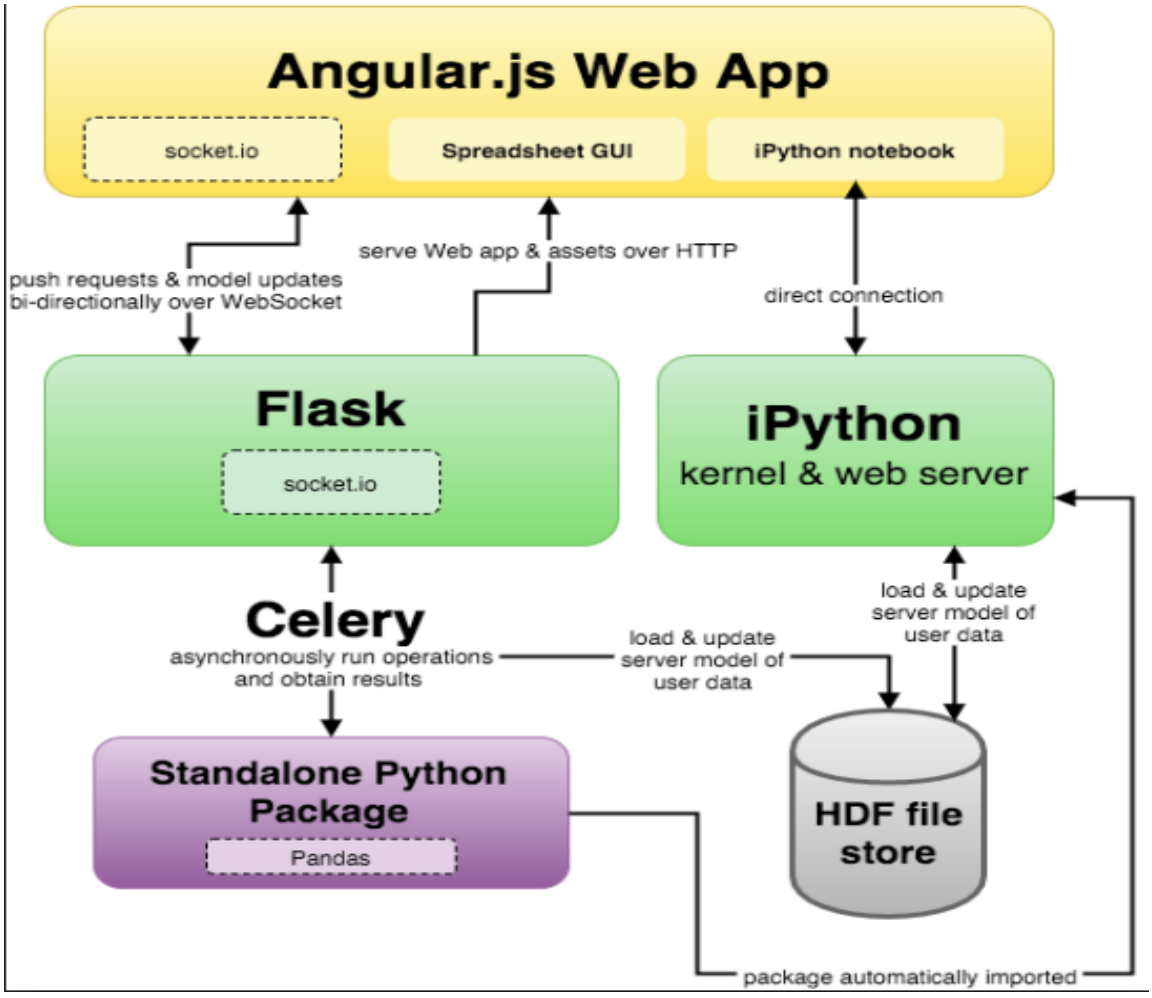

Figure 8: Web app architecture

The back end uses a micro-framework called Flask, is built on top of Jinja2 and WSGI toolkit. Running Flask by itself as a Web server on the internet is possible thanks to the WSGI layer. In our case, any services that the User Interface provides is a request from a services, from crawling using online to extract body, title and source to an URL to the score fakeness given to a news.

Flask will sit behind a web server and be called upon when the servers recognizes a request.

WSGI is the protocol used by Apache's httpd to manage the load on Flask app and takes the requests from web server and python back-end side application to translate actions between them.

The server security will be manage with SSL and TLS, this tools certificates allow servers to be verified by a trusted third-party based upon the domain name that a browser will be connecting to. In other words, the used server and the browser can't be sniffed.

4.10.1 WEB APPLICATION USAGE IN FANDANGO

FANDANGO web application will be the binder of all the micro-services provides by partners of the project. The final user will be able to insert an URL containing a news with different kind of medias (video, photos and text) and check the following features:

- Probability of fakeness
- Authenticity of associated pictures or videos.
- Propagation of the news through graph analysis.
- Visualization of related news.
- Visualization of claims related to a selected portion of the news.
- Suggestion of where to find related info on the Open Data available.

4.11 TENSORFLOW

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

TensorFlow uses Keras is a high-level neural networks API, written in Python. Keras puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.

A Keras model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as few restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that you can combine to create new models.

Keras makes available established deep learning models like VGG, ResNet and Inception, through the application library, alongside pre-trained weights in huge well known datasets like ImageNet and MS-Coco. These models can be used for prediction, feature extraction, and fine-tuning.

TensorFlow offers Serving which is a flexible, high-performance serving system for machine learning models, designed for production environments. TensorFlow Serving makes it easy to deploy new algorithms and experiments, while keeping the same server architecture and APIs. TensorFlow Serving provides out-of-the-box integration with TensorFlow models, but can be easily extended to serve other types of models and data.

4.11.1 TensorFlow usage in FANDANGO

TensorFlow will be used by WP4 machine learning (ML) modules both for training deep learning models as well as for testing (predicting) unseen content, after the model has been trained. As described before this library fit very well in our use case for the multimedia analysis, this because the nature of the multimedia datum.

For the training of the model and for a first bulk ingestion we are going to use a GPU architecture based on (Nvidia 1080ti) processors. In this way the training phase will be much more faster (10 times faster than normal CPU). Than on the normal life cycle of the project TensorFlow will run on normal CPU to balance the cost/benefits ratio. The cloud architecture we've chosen let the administrator to decide, also on running, to switch between different CPU/GPU architecture without a huge impact.

A service for each model will be developed that will use the TensorFlow runtime library to exploit the GPU or CPU enables server infrastructure. Data will be collected from a Kafka channel and will be send back to a Kafka channel to ensure easy communication among users of the ML outputs.

4.12 Ambari

Ambari provides granular control over all parts of the HDP cluster, allowing for increased performance, increased service quality and reduced administration costs. Furthermore, it is possible to easily deploy and centrally manage all the tools. In fact, it is possible to automate the installation of the various components and their management, also thanks to diagnostic reports on the status of the services.

4.12.1Ambari usage in FANDANGO

As introduced in the Hadoop description chapter Ambari will be used for monitoring the cluster and the installed tools. It allows the FANDANGO developers and than the final users to manage the infrastructure, to access data on HDFS like a classic file system, I provides interactive view to run Hive's queries and an Oozie interface to create and mange workflows and coordinators.

4.13 Oozie

Apache Oozie is used to schedule and coordinate processes in the Hadoop cluster. It relies on workflows, composed by several different actions, whose results can affect the next transitions. Workflow can be executed manually or through coordinators that can schedule wf. Both can be monitored by a console and parameterized. The coordinator can be scheduled according to the Oozie parameterization feature with variables, built-in constants and built-in functions (expressed by the "${coord:" syntax).

Thanks to the User-Retry capabilities provided by Oozie when an action is in ERROR or FAILED state, it's possible to design and implement worflows' retry policies. Depending on the nature of the failure, Oozie can define what type of errors allowed for User-Retry. In our scenario HDFSM can rely on controls among file-exists-error FS009 or FileNotFoundException JA008 in action executor, adding some specific actions at the end of the main SSH/subworkflow action, with the purpose of testing the overall result of the WF. User-

Retry allows user to give certain number of retries, so user can find the causes of that problem and fix them when action is in USER_RETRY state. If failure or error does not go away after max retries, the action becomes FAILED or ERROR and Oozie marks workflow job to FAILED or KILLED.

4.13.1 OOZIE USAGE IN FANDANGO

Oozie can be use to schedule and organize batch jobs in the Hadoop environment. In this iteration Oozie has not been used yet, but in the target version it will allow user to schedule the retraining of the model and to create, from a graphical tool, complex workflow for the data management of the platform.

5. CROSS FUNCTIONALITIES

Around the core components that enable data processing, a set of support tools for the sharing of the code have been included and which allow to apply, in some parts, DEVOPS techniques. Among these, particular importance is attributable to Git that allows the sharing of the code and the bug tracking.

For the web component, Docker containers are used. As already discussed in previous chapters, the architecture is implemented in a cloud infrastructure in order to meet the elasticity and replicability requirements of FANDANGO.

5.1. GITHUB

GitHub is a code hosting platform for version control and collaboration. It's a web-based platform that offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features.

Git is a version-control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source-code management in software development, but it can be used to keep track of changes in any set of files. As a distributed revision-control system, it is aimed at speed, data integrity, and support for distributed, non-linear workflows.

GitHub, moreover, provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project. It offers plans for both private repositories and free accounts which are commonly used to host open-source software projects.

The FANDANGO Git repository is available at: https://github.com/FANDANGOrg/FANDANGO.

In Git the version control is managed using branches. Branching means working on different versions of a repository at one time. By default our repository has one branch named master which is considered to be the definitive branch. We use branches to experiment and make edits before committing them to master. When you create a branch off the master branch, you're making a copy, or snapshot, of master as it was at that point in time. If someone else made changes to the master branch while you were working on your branch, you could pull in those updates. On GitHub, saved changes are called commits. Each commit has an associated commit message, which is a description explaining why a particular change was made. Commit messages capture the history of the changes, so other contributors can understand what you've done and why. To use Git, specific commands are used to copy, create, change, and combine code. These commands can be executed directly from the command line or by using an application like GitHub Desktop or Git Kraken.

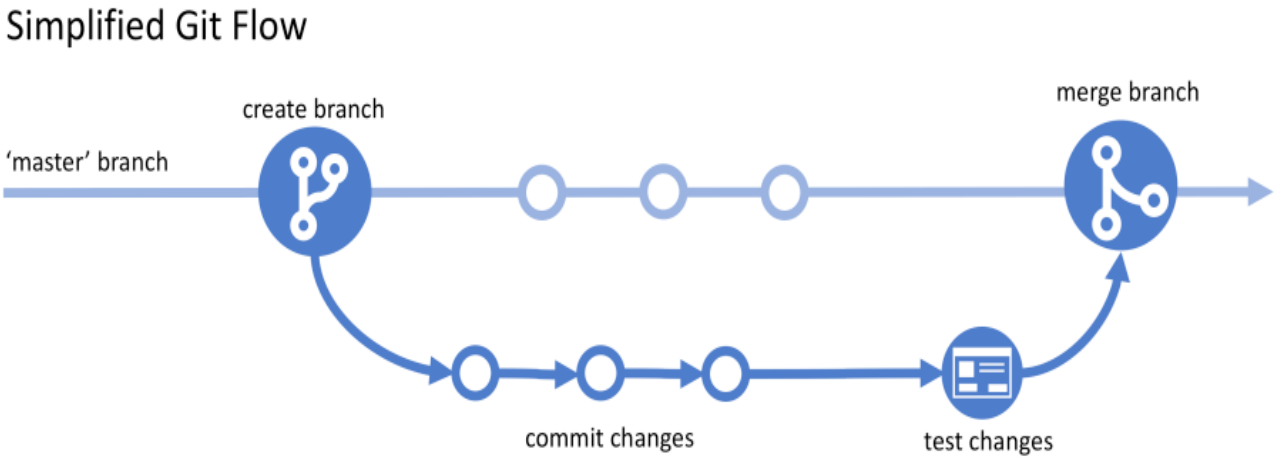

Figure 9: Git flow example

5.2. GitHub Issue tracking

An issue tracking system (also ITS, trouble ticket system, support ticket, request management or incident ticket system) is a computer software package that manages and maintains lists of issues.

In FANDANGO project the official bug tracker is the one provided by GitHub, and it can be accessed from the GitHub home page of the project. GitHub's tracker is called Issues, and has its own section in every repository. The monitoring of GitHub problems was chosen for attention to collaboration, references and text formatting.

5.3. Docker

In FANDANGO project some components are packaged using the technology of containers. A container is a set of one or more processes that are isolated from the rest of the system. All the files necessary to run them are provided from a distinct image, meaning that Linux containers are portable and consistent as they move from development, to testing, and finally to production. This makes them much quicker than development pipelines that rely on replicating traditional testing environments. The main difference of containers and virtualization is that Virtualization lets your operating systems (Windows or Linux) run simultaneously on a single hardware system. Containers share the same operating system kernel and isolate the application processes from the rest of the system.

The container platform used by FANDANGO project is Docker. With Docker, containers can be compared to extremely lightweight, modular virtual machines. Using Docker provides flexibility with to create, deploy, copy, and move containers from environment to environment; this helps to optimize apps for the cloud.

text formatting.

5.4. DockerHub

Docker Hub is a cloud-based registry service which allows to link to code repositories, build images and test them, stores manually pushed images, and links to Cloud; in this way the deployment of image to user's hosts is possible. Docker Hub also provides other useful functions: a centralized resource for container

image discovery, distribution and change management options, a system of user - team collaboration, and the workflow automation throughout a development pipeline.

For team projects, Docker Hub provides organizations, i. e. teams that gives team users access to shared image repositories. A Docker Hub organization can contain public and private repositories just like a user account. Access to push or pull for these repositories is allocated by defining teams of users and then assigning team rights to specific repositories. Repository creation is limited to users in the organization owner's group. This allows to distribute limited access Docker images, and to select which Docker Hub users can publish new images.

5.5. AUTOMATIC BUILD OF GITHUB TO DOCKERHUB

It is possible to set up an Automated Build of a repository of GitHub on Docker Hub. This is manageable linking the FANDANGO GitHub account on Docker Hub. This allows the registry to see GitHub repositories. Linking to GitHub grants Docker Hub access to all of the repositories. Once it is linked, it possible choose a source repository from which to create the Automatic Build. GitHub organizations and private repositories forked from organizations are made available for auto builds using the "Docker Hub Registry" application, which needs to be added to the organization - and then applied to all users. This is one of the steps for the continuous delivery process.

5.6. KUBERNETES

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. It is written in Go and, while originally developed as a closed source project by Google, the open-sourced project is hosted by the Cloud Native Computing Foundation (CNCF).

Among its features, Kubernetes offers service discovery and load balancing for containers, automatic bin packing for container placement and automated roll-outs, rollbacks and self-healing for the underlying applications and their configuration. Kubernetes supports any container runtime that implements the Open Container Initiative (OCI) runtime-spec, among them Docker.

Kubernetes adopts a Master-Worker architecture: the Master machine contains components that make global decisions about the cluster and handle cluster events, while the Worker machines, or Nodes, run one or more Pods, groups of one or more containers that share network and storage resources.

In FANDANGO project we'll create a Kubernetes cluster in order to manage the Docker instances that contains the micro services for the interaction with the backend. In this way they can be easily scaled horizontally.

5.7. AZURE CLOUD

According to this reference architecture, the FANDANGO Platform will be hosted on Azure cloud. Azure cloud computing service is a shared pools of configurable computer system resources and higher-level services that can be rapidly provisioned with minimal management effort. Cloud computing enable organizations to focus on their core businesses instead of expending resources on computer infrastructure and maintenance. Azure allows the creation and management of virtual machine both inside its web

application using a UI to handle the configuration's parameters and also with Power Shell script, a scripting language that permits to use Azure services via code.

For FANDANGO project we need to deploy different frameworks to enable the ingestion and elaboration of data. At this stage standard virtual machine are based on Linux and they have 4 vcpus and 16 GB memory. Based of task resource needs it is possible to easily scale the architecture, adding new virtual machine or increasing their capabilities, to fulfil the performance requirements. Number and size of virtual machines may will change if requirements of the project change.

The set FANDANGO cluster defined for the implementation is composed by:

- Apache NiFi
    - 1 machine 16GB Ram/8 cores
- Siren on Elasticsearch
    - 1 machine 64GB Ram/8 cores
- TensorFlow
    - 1 machine 16GB Ram/8 cores
- Hadoop and Apache Spark
    - 4 machines 32GB Ram/8 cores
- Neo4J
    - 1 machine 16GB Ram/8 cores
- Kafka
    - 3 machines 32GB Ram/8 cores
- Web application (Kubernetes cluster)
    - 1 machine 16GB Ram/8 cores

When possible we will deliver services in containers, to simplify the portability and the deployment of the system.

The azure portal interface makes very easy to scale vertically or horizontally the services. This is one of the reasons for the adoption of a cloud infrastructure and it fulfils the non functional requirement “architecture should be easy scalable in order to handle the introduction of new sources or handle peaks”. Suppose for example that the installation should analyze a specific event, for example the European Election, As the election data approach, the number of news will increase significantly. For this reason it will be necessary to strengthen the NiFi and Kafka clusters. When the resources will be no longer needed they will be released optimizing the costs of the installation.

In the next image we report a screenshot of the azure portal (administration page) of Azure.

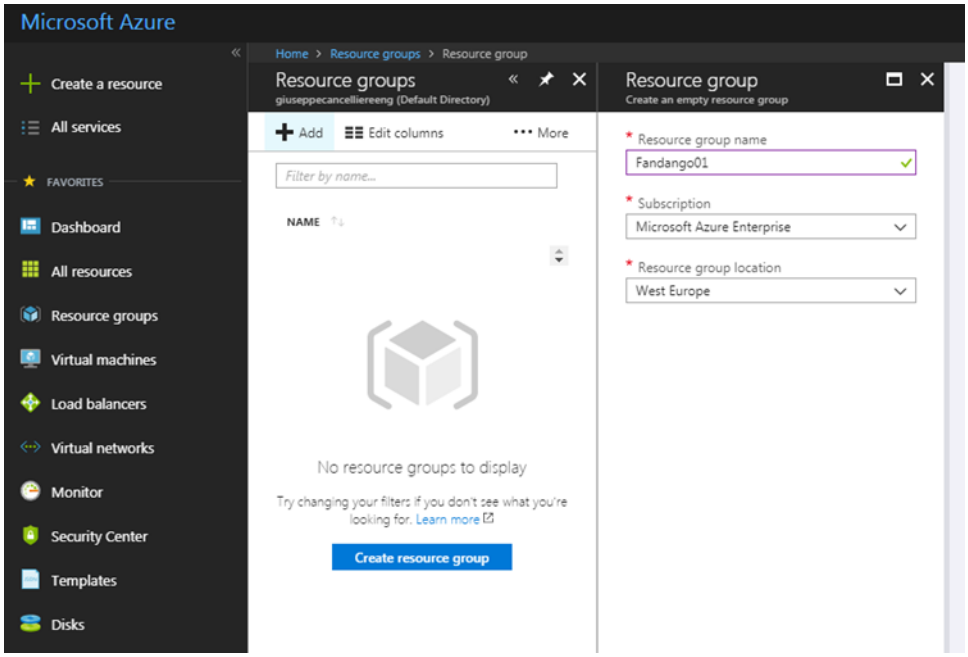

Figure 10: FANDANGO resource group definition on Azure

It is important to underline that the products that comprise the architecture are not dependent on the chosen cloud platform, therefore it is possible to replicate the architecture on other public, cloud or private cloud or on premise.

6. CONCLUSION

The aim of the document is to provide a description of the FANDANGO architecture, reasons for the choice of the various components and the relation between them.

The architecture chosen for FANDANGO project aims to host flows of continuous loading of data from the web and process them online. To guarantee this workflow, several tools have been introduced that allow the management of data like NiFi, Kafka and Spark streaming (in this case microbatch). These tools are the fulcrum around which other components perform specific processing required to satisfy the user requirements defined in D2.3.

NiFi allows to define loading flows and the preparation of initial data processing from the web. Kafka is a store of streams of records in a fault-tolerant durable way that stores data until the spark streaming ends up processing them.

After a first phase of pre-processing that extracts three types of multimedia, textual and metadata data, the various portions of data are managed in three different ways and with different technologies:

- Spark MLlib for the textual data
- TensorFlow for multimedia data
- Neo4J for metadata

These technologies have been chosen because their characteristics are consistent with the analysis to be performed on the different types of data. Spark MLlib contains high-quality algorithms that leverage iteration, and can yield better results than the one-pass approximations. TensorFlow is a library for high performance numerical computation very performing on GPU and CPU architecture; this is crucial for the multimedia analysis. Finally, the use of a graph storage and analysis system like Neo4j allows to analyze the relationships between the different entities present in the metadata (such as authors, publishers, etc.).

To allow quick access and perform similarity analysis between news, claim and open data, it was decided to use as final storage Elastic in the Siren platform that adds federation and advanced analysis functionality.

The iteration with the user is provided by a web interface based on Angular js and REST services implemented on a well-established python-based architecture.

In the document we have also described other tools that do not appear in the main flow, but which are fundamental for the execution of processes such as Hadoop / HDFS were we save all the history in raw format, zeppelin for prototyping, Ooozie for scheduling, Ambari for management.

To ensure the elasticity and portability of the solution, FANDANGO relies on a cloud environment that guaranties both horizontally and vertically scaling and the replicability of the solution. Where possible, container technology was chosen.

In order to provide a better description of the relations and of the communication paths between components an addendum related to D5.1 will be attached to D2.4.