Grant Agreement N° 872592

PLATOON

Digital platform and analytic tools for energy

# Deliverable D5.5

# Testing and documentation

Contractual delivery date:

M27

Actual delivery date:

31/03/2022

Responsible partner:

P03: UBO, GERMANY

| | |
|---|---|
| **Project Title** | PLATOON – Digital platform and analytic tools for energy |
| **Deliverable number** | D5.5 |
| **Deliverable title** | Integration and Testing Results |
| **Author(s):** | UBO, ENGIE, TECN, ENG, IMP, TIB, PDM, MINSAIT, CS |
| **Responsible Partner:** | UBO (Partner nº3) |
| **Date:** | 18/03/2022 |
| **Nature** | Other |
| **Distribution level (CO, PU):** | PU |
| **Work package number** | WP5 – WP5-Big data sharing and analysis reference implementations |
| **Work package leader** | UBO, Germany |
| **Abstract:** | This task focuses on the integration testing of components of the reference architecture. The tests evaluate how the components operate together. The precondition for the tests is that each component has been already tested in isolation and is working properly. This has been done in other tasks of WP5, WP2, WP3 and WP4. For the purposes of testing, we conceived different testing scenarios. The results of the tests are presented and documented in this deliverable. |
| **Keyword List:** | Integration Testing, Reference Architecture |
| **Reviewer(s):** | Juan Prieto (IND)<br>Erik Maqueda (TECN)<br>Philippe Calvez (ENGIE) |
| **Approved by:** | |
| **Recommended/mandatory readers:** | All WP5 partners. WP4 and WP6 partners for reference for implementation and testing. WP8 and WP9 partners for exploitation and dissemination. |

**The research leading to these results has received funding from the European Community's Horizon 2020 Work Program (H2020) under grant agreement no 872592.**

## Document Revision History

| Version | Date | Modifications Introduced | |
|---|---|---|---|
| | | Modification Reason | Modified by |
| v0.1 | 28/01/2022 | 1st version of TOC | UBO |
| v0.2 | 18/03/2022 | First finished version for internal review | All |
| v0.3 | 19/03/2022 | Final version for internal Review | ENG, UBO |
| v0.4 | 25/03/2022 | Corrections based on internal review | UBO |

# Table of Contents

## List of Figures

## List of Tables

# Terms and abbreviations

**Table 1: Terms and Abbreviations**

| | |
|---|---|
| API | Application Programming Interface |
| CA | Consortium Agreement / Certification Authority |
| CO | Confidential |
| DAPS | Dynamic Attribute Provisioning Service |
| DM | Dissemination Manager |
| EC | European Commission |
| EM | Exploitation Manager |
| GA | Grant Agreement |
| GAM | General Assembly Meeting |
| HW | Hardware |
| IOT | Internet of Things |
| IDS | International Data Spaces |
| IT | Information Technology |
| PaaS | Platform as a Service |
| PKI | Public Key Infrastructure |
| PM | Project Manager |
| PU | Public |
| QA | Quality Assurance |
| RE | Restricted |
| SC | Steering Committee |
| SLA | Service Level Agreement |
| SaaS | Software as a Service |
| SW | Software |
| TM | Technical Manager |
| WP | Work package |
| WPL | Work package Leader |

## Executive Summary

Task T5.5 is focusing on integration testing of components of the PLATOON reference architecture and the documentation of the test results. The tests evaluate how the components operate together and potential issues are communicated with the responsible partners. One precondition of the tests is that each component has been tested in isolation and are working properly. These tests have been conducted in other tasks of this WP (T5.2-T5.4) and corresponding tasks from WP2, WP3 and WP4.

This deliverable also presents typical scenarios which are used to test the reference architecture. The scenarios include an Edge-Scenario, a Non-Edge-, and a Data Sharing (IDS) scenario. The result of running our tests on these scenarios is documented in this deliverable.

In addition to the test result, the deliverable provides step-by-step instructions for the purpose of replication of the testing scenarios. This provides users of the PLATOON architecture a reference point to validate their PLATOON setup.

All discovered issues have been communicated to the responsible partners. In an iterative manner, the issues have been resolved which resulted in updated components with improved quality before implementation and validation in large scale pilots in WP6.

The tests in this deliverable focus on integration tests using representative, anonymised datasets on fictitious scenarios to guarantee a broad coverage of different use-cases. For other scenarios involving real datasets and real-world use-cases, the reader is referred to WP6.

# 1 Introduction

The purpose of Task T5.5 is the testing of components of the PLATOON reference architecture and the documentation of the test results. This task focuses on integration testing of the different components. This means that the tests evaluate how the components operate together according to the PLATOON reference architecture. These integration tests are an important part of the development cycle of the PLATOON reference architecture as they ensure that the individual components developed in other Work Packages can be combined in a meaningful way to support different scenarios in the energy domain. It is important to note that the isolated test of individual functionalities of the different components is not part of Task T5.5. Hence, the precondition to the tests performed in this task is that each component has been already tested in isolation. Such tests have been performed during the development cycle of individual components in other tasks of this WP (T5.2-T5.4) and corresponding tasks from WP2, WP3 and WP4.

For the purposes of integration testing, we conceived different scenarios to represent typical setups commonly encountered in the energy domain. The scenarios are inspired by the different pilots in the PLATOON project.

The first scenario for our tests is the Edge-Scenario. This scenario focuses on real time or streaming data coming from IoT or edge computing devices. This data is transformed and stored in a database specifically designed for storage and retrieval of time-series streaming data. This data is then processed in the edge and presented to the user by the visualization component.

The second scenario is the Non-Edge-Scenario. As the name suggests, this scenario does not operate on time-series data from edge computing devices but rather on more traditional batch data scenario using static data sources. The data in this scenario is semantified using the respective components and then made available to consumers through a federated query engine incorporating data from different sources as per explained in WP2 and T5.2 and T5.3. The analytics part is done by SANSA which accesses the data from the federated query engine. Finally, a visualizer component is used to present the results of the whole workflow.

To cover the Data Sharing scenario according to IDS reference architecture defined in WP2 and WP3, we defined a third scenario that focuses on the Meta-Data registry, Vocabulary Provider, and corresponding IDS connectors for data producers and consumers.

With these scenarios, Work Package 5 can cover a large number of possible setups of the PLATOON reference architecture. The result of these tests is documented in this deliverable and are forwarded to the responsible partners to help improve the existing component.

The rest of this document is structured as follows: Section 2 introduces the different scenarios in detail. In Section 3, the results of the tests are presented and documented. Finally, in Section 4, we conclude this deliverable by summarizing the outcomes of our tests.

# 2 Testing Scenarios

## 2.1 Edge-Scenario

The Edge-Scenario focuses on real time streaming data coming from IoT or edge computing devices. This data is then transformed and stored in a database specifically designed for storage and retrieval of time-series data. This data is then presented to the user by the visualization component. The following image depicts the architecture of the Edge-Scenario.

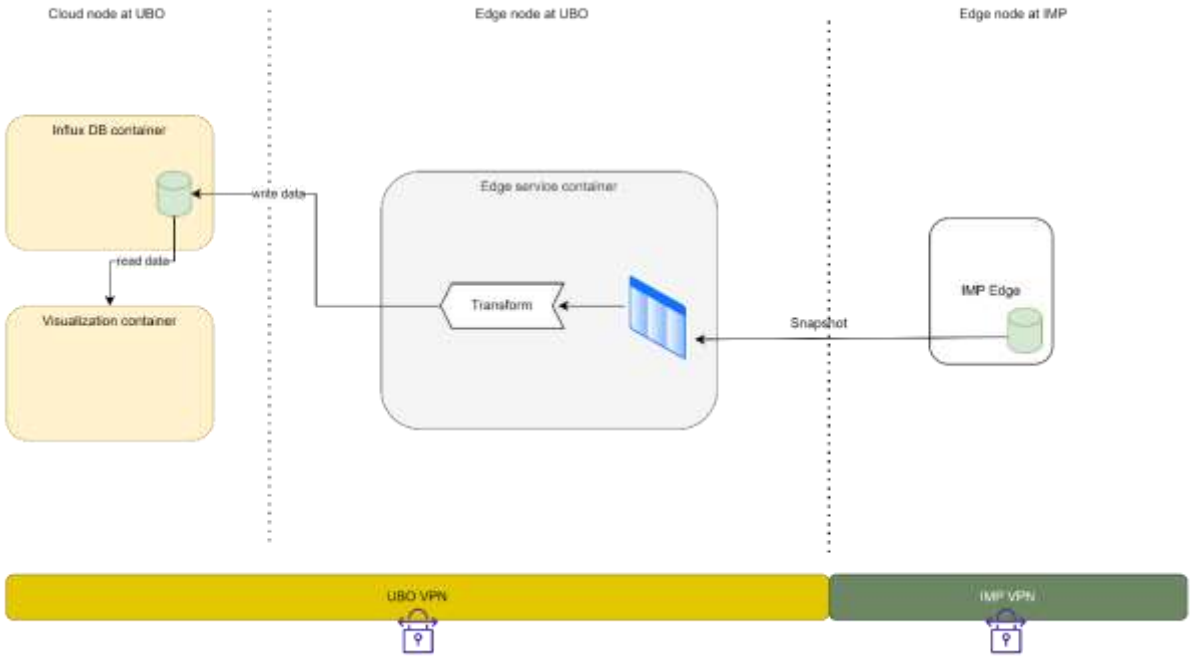


**Figure 1: Edge Scenario architecture**

### 2.1.1 Edge service

Container at the Edge node uses InfluxDB python client library to write data to the InfluxDB that is positioned in the Cloud or Central node. The exported csv file used by the Edge service has data for around 45 days with 15 min resolution. The edge service pushes one data array to InfluxDB every minute, timestamp is changed to current timestamp.

The container image was built for the Edge service. We also used docker-compose to set the deployment location and authentication of InfluxDB for unit testing. The content of the docker-compose.yml file is as follows:

```
version: "3.9"
services:
    unittes_container:
        image: andrejcampa/unittest_edge
        environment:
        - INFLUXDB_ADMIN_USER=user
        - INFLUXDB_DB=demo
        - INFLUX_DB_IP=xxx.xxx.xxx.xxx
        - INFLUXDB_ADMIN_PASSWORD=pass
        ports:
        - "8086:8086"
```

### 2.1.2 Data and Database

For reasons of confidentiality of the information, the data in this scenario cannot be processed in real time by the edge nodes of distributed computing, for its management a connectivity via VPN has been established (between SAMPOL and IND) where the node loads the set of data once the SAMPOL database (Legacy BD) has validated that information.

Currently and in the built test environment, said database is replicated in a relational database internal to the node, where through an automatic process (connector to access the MySQL-Connector) it is read and published on the edge-node-messaging data bus.

This information that is published in the middleware is stored in the InfluxDB database, this database is integrated in a collection of technologies called Tick Stack, a collection of open-source components that combine to offer a platform to store, visualize and easily monitor time series data such as metrics and events. The components are: Telegraf, a server agent for collecting and reporting metrics; InfluxDB, a high-performance time series database; Chronograf, a user interface for the platform; and Kapacitor, a data processing engine that can process, stream, and pool InfluxDB data.

In particular, InfluxDB is a database oriented to time series, given the computing capabilities of the Edge Node are limited, its use is highly recommended since it shows good performance when working with sensors and for data analysis, it has a language SQL-like query, is compatible with JSON and is specifically designed for metrics and events, type of information that the Edge Node will process. Of these components, Chronograf and InfluxDB as fundamental elements for the storage, monitoring and validation of information.

The data that is being managed and finally published in JSON format in order to be consumed by any other process in a standardized way are the following:

- Identification (meter/concentrator)
- Date and time (acquisition)
- Import Energy
- Export Energy
- Reactive energies in the four quadrants

As has been detailed, the edge node deploys an InfluxDB container that stores the information published with the SAMPOL centralized database connector. This information consists of hourly records of the defined energies. In the following figure, they are shown for a certain interval of time.

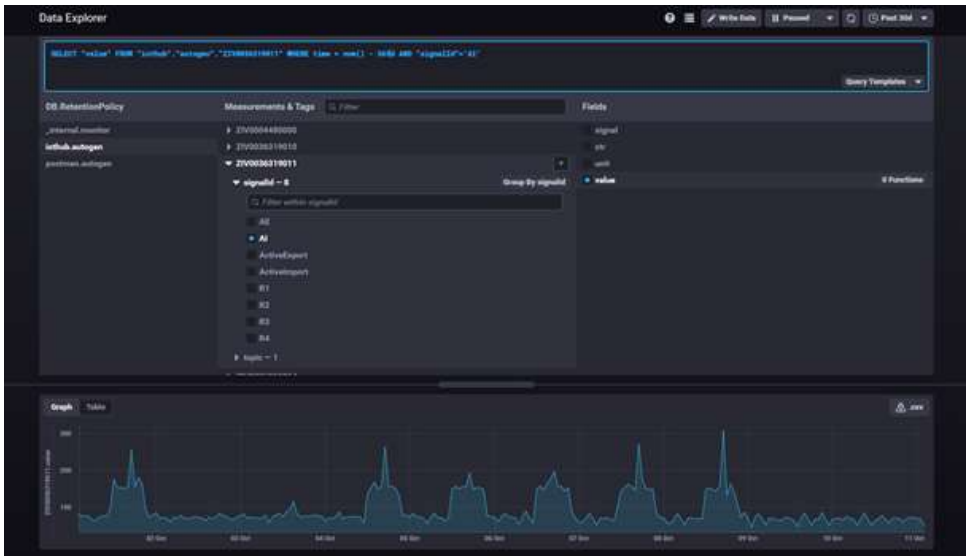


**Figure 2: Snapshot of Energy data used in Edge scenario**

In the architecture built and deployed at the edge node level we also used docker compose to set the deployment location and authentication of InfluxDB for unit testing. The content of the docker-compose.yml file is as follows:

influxdb:

container_name: influxdb

image: emslab.onesaitplatform.com/onesait-things/edge-influxdb:1.0.0.RELEASE

volumes:

- ./influxdb:/var/lib/influxdb/data

restart: always

environment:

- MQTT_HOST=mqtt
- MQTT_PORT=1883
- MQTT_TOPIC=platoon

Ports:

- 8086:8086
- 5300:5300
- 8888:8888

networks

edgenet

As a validation tool for data acquisition, analysis and integration in Platoon's edge architecture, the Node-RED tool has been used.

Node-RED, initially developed by IBM and currently part of the JS Foundation, is a stream-based programming tool. It is a free and open-source tool. In Node-RED, applications are described as a set of black boxes (nodes) connected to each other (creating a flow). Each node receives data, works with it, and then passes the result to the next node. One of the main advantages of Node-RED is that it can run on low-cost hardware (eg Raspberry Pi) and in the

cloud, and is designed to be used by a wider range of users (not just professionals). However, the functions must be written in JavaScript, which can be a barrier for less advanced users.

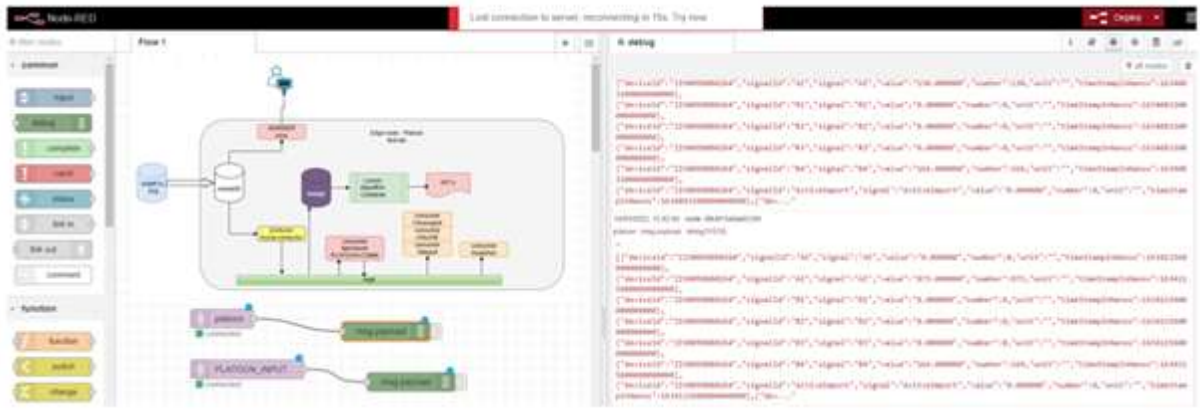

**Figure 3: Node-Red validation tests.**

### 2.1.3 Analytics Dashboard

For visualization of Edge data, we used Grafana (https://grafana.com/). Grafana is a multi-platform open source analytics and interactive visualization web application. It provides charts, graphs, and alerts for the web when connected to supported data sources. Grafana allows users to query, visualize, alert on and understand metrics irrespective of where they are stored. Grafana is available on GitHub (https://github.com/grafana/grafana) and also, Docker hub (https://hub.docker.com/r/grafana/grafana) and can be easily integrated in the PLATOON reference architecture.

More details about the development, functionalities and configuration of the dashboard are provided in T5.4.

## 2.2 Non-Edge Scenario

The Non-Edge scenario operates on historical data sources. The data in this scenario is semantified using the semantic adaptor and then made available to consumers through a federated query engine incorporating data from different sources. The analytics part is done by SANSA which accesses the data from the federated query engine. Finally, a visualizer component is used to present the results of the whole workflow. The following figure shows the high-level architecture of how components interact with each other.

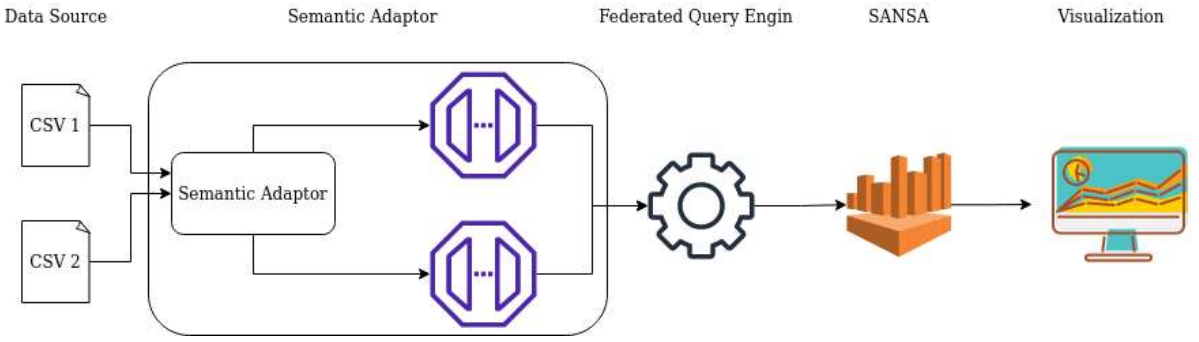


**Figure 4: Non-Edge Scenario architecture**

### 2.2.1 Data

For this scenario, Engie provided data related to Building occupancy prediction. The data is in CSV format and contains following columns:

- Id
- Date
- Hour
- Zone
- Source
- Connections

The “Source” column contains only the two values “WiFi” or “LAN”. The data has been anonymized and aggregated per hour. The file contains 31065 datapoints at a size of 1.4 MB. To test the pipeline with bigger data, some synthetic data has been generated with different sizes. The procedure is explained in Section 39.

### 2.2.2 Semantic Adapter & Federated Query Engine

The knowledge base creation pipeline presented in deliverables D5.2 and D5.3 integrates a semantic adapter and a federated query processing; it follows the PLATOON reference architecture presented in D2.1/D2.5. This pipeline is agonistic of the query engine used to implement the semantic connector and the federated query processing engine. The semantic adapter converts data sources in different formats, e.g., CSV, JSON or relational databases, into RDF; this conversion is guided by mapping rules specified in a mapping language, e.g., SPARQL-Generate or RDF Mapping Language (RML). These rules define declaratively concepts (i.e., classes and properties) in the PLATOON semantic data models (presented in D2.3) in terms of data collected from the PLATOON data sources. As a result, during the execution of these rules, the semantic adapter populates classes and properties in a unified knowledge base. Then, the pipeline uploads this unified knowledge base into a federation of SPARQL endpoints (e.g., in Virtuoso or Fuseki). The federated query processing engine provides an interface to these endpoints; it enables users to explore the knowledge graph and retrieve the answers to a SPARQL query. Two implementations of the pipeline are available; one resorts to SDM-RDFizer and the other one to SPARQL-Generator. The former, which is used in this deliverable, has a demo component as analysis tool which has been specifically designed to replicate the PLATOON reference platform as close as possible and make use of REST APIs. This enables us to validate the platoon reference architecture. Both pipelines are described in D5.2.

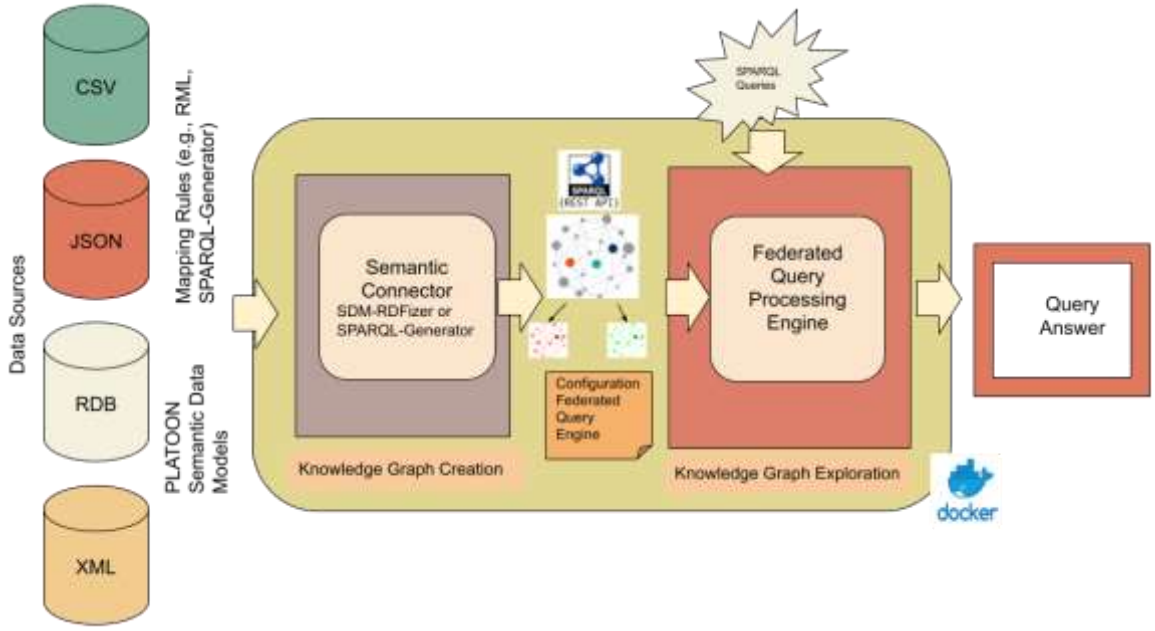


**Figure 5: Knowledge Graph Creation Pipeline**

### 2.2.2.1 A Semantic Adaptor based on SDM-RDFizer

The pipeline is implemented as a bash script that executes a series of Docker images, each one implements a component of the pipeline. As a result, the pipeline is comprised of three Docker images: a) the SDM-RDFizer image; b) the Virtuoso images; and c) the FQP image. This pipeline is available in a GitHub repository[1] and uses dockerized components.

README.md

# PLATOON Pipeline

This repository contains basic settings for PLATOON Pipeline.

- `scripts` - contains scripts used for transforming sources to RDF and loading it to triple store (Virtuoso) - `virtuoso-script.sh` - used to remotely connect and load data using `isql-v` tool of virtuoso on command line - `load_to_virtuos.py` - used to load the transformed RDF data to virtuoso using the `virtuoso-script.sh` script - `transform_and_load.py` - performs both transforming raw data to RDF and loading it virtuoso using the `virtuoso-script.sh` script - `mapping_parser.py` - generates the necessary file for the execution of DeTrusty
- `configuration_files` - contains configuration files for the execution of the pipeline - `config_rdfizer.ini` - configuration file for materializing the Knowledge Graph using SDM-RDFizer. - `config.ini` - configuration file for generating the DeTrusty configuration file.
- `docker-compose.yml` - docker compose setup for transforming data to RDF and load it to `Virtuoso` triple store.

**Figure 6: The PLATOON Pipeline**

The aforementioned GitHub repository contains required files and scripts for the execution of the PLATOON pipeline. These components are:

- ***Scripts:*** is a folder containing the scripts that transform mapping files and their corresponding data sources into RDF data, which is then loaded into the triple store (Virtuoso). These scripts are:
  - *Virtuoso-script.sh:* used to remotely connect and load data using the *isql-v* tool of Virtuoso.

- *Load_to_virtuoso.py: executes* the *virtuoso-script.sh* to upload transformed RDF data to the Virtuoso triple store.
- *Transform_and_load.py:* performs both the transformation of raw data into RDF data by invoking the SDM-RDFizer and uploads the data into a triple store by using *virtuoso-script.sh.*
- *Mapping_parser.py:* generates the input file necessary for the execution of FQP by associating the classes from the mapping files with its corresponding predicates and endpoint that contains the RDF data.

**Figure 7: Portion of transform_and_load.py illustrating the uploading of RDF data in the triples store.**

- ***Configuration_files:*** is a folder that contains the configuration files that are necessary for the execution of the SDM-RDFizer and mapping_parser.py script.

```
[default]
main_directory: .

[datasets]
number_of_datasets: 2

[dataset1]
endpoint:  http://pilot2akg:8890/sparql
mapping: ${default:main_directory}/sam/mapping.ttl

[dataset2]
endpoint:  http://localhost:8890/sparql
mapping: ${default:main_directory}/sam/mapping.ttl
```

**Figure 8: Example of Configuration File illustrating the execution of the SDM-RDFizer.**

- ***Docker-compose.yml:*** docker compose creates the docker images of the SDM-RDFizer, DeTrusty, and Virtuoso.

```yaml
version: "3.3"
services:
  sdmrdfizer:
    image: asakor/sdmrdfizier:4.0.1
    hostname: sdmrdfizer
    container_name: sdmrdfizer
    domainname: platoon
    volumes:
      - .:/data
    networks:
      - platoon
    depends_on:
      - pilot2akg
    environment:
      - SPARQL_ENDPOINT_IP=pilot2akg
      - SPARQL_ENDPOINT_USER=dba
      - SPARQL_ENDPOINT_PASSWD=dba
      - SPARQL_ENDPOINT_PORT=1111
      - SPARQL_ENDPOINT_GRAPH=http://platoon.eu/Pilot2A/KG
      - RDF_DUMP_FOLDER_PATH=/data

  detrusty:
    image: prohde/detrusty:0.2.0
    hostname: detrusty
    container_name: detrusty
    domainname: platoon
    volumes:
      - ./DeTrusty/Config/:/DeTrusty/Config/
    ports:
      - "5000:5000"
    networks:
      - platoon
    depends_on:
      - pilot2akg

  pilot2akg:
    image: kemele/virtuoso:6-stable
    hostname: pilot2akg
    container_name: pilot2akg
    domainname: platoon
    volumes:
      - ./rdf-dump:/data
    ports:
      - "8891:8890"
      - "1116:1111"
```

**Figure 9: Example of docker-compose.yml file illustrating the docker compose file used for the generation of the docker images that are required for the execution of the Pipeline.**

DeTrusty (a.k.a. FQP) is a query engine that implements federated query processing against SPARQL endpoints. It decomposes the input query into star-shaped sub-queries, i.e., all triple patterns in a sub-query share the same subject. If an RDF type statement is present in a sub-query, it will be used to identify the sources that contribute to the sub-query in question. If no such statement is present, DeTrusty selects all sources that contribute to RDF classes that contain all predicates of the sub-query. This allows DeTrusty to minimize the number of contacted SPARQL endpoints. Each sub-query is executed over the previously selected sources for the sub-query. The partial results retrieved from the endpoints are combined at the query engine level using non-blocking physical operators. DeTrusty creates bushy plans in order to speed up the query execution. These features enable DeTrusty to continuously generate complete and sound query results while minimizing the number of contacted endpoints and query execution time. The current version of DeTrusty is capable of executing SPARQL

SELECT queries. The SERVICE clause from SPARQL 1.1 is also implemented. Some SPARQL 1.1 features are not yet implemented, e.g., GROUP BY and aggregate functions. FQP can be run in a Docker container. After providing DeTrusty with the semantic source description of the set of SPARQL endpoints, it can be used via its HTTP API as if it was a regular SPARQL endpoint. Deliverable D2.8 provides a detailed description of the federated query engines integrated into the PLATOON framework; it also reports on the results of the performance empirical evaluation of DeTrusty on Pilot 2a knowledge base.

### 2.2.3 Analytics

In the work packages WP2, WP3, WP4, many technical standards, and tools were developed, which became part of the PLATOON reference architecture. Within work package WP5, these are evaluated. As part of the Generic Big Data Analytics Toolbox, the Semantic Data Analytics Stack SANSA was further developed. This is based on Apache Spark and Apache Jena natively using large-scale RDF Knowledge Graphs in standard downstream machine learning pipelines. The pipeline modules enable Semantic Similarity Estimation, Clustering, Classification, Regression, and anomaly detection. Developments have also been made to make these modules Dockerized and usable through REST APIs. Apache Livy, Hue, HDFS, and Docker are used for this purpose. All the work is published in scientific papers. The publications are the following:

- DistSim [1]
    - Similarity Estimation as backbone for clustering and unsupervised classification [2]
    - Machine Learning pipeline feature extraction on Knowledge Graphs [3]
    - Generic downstream machine learning pipelines for RDF Knowledge Graphs
- DistAD [4]
    - Numeric Outlier detection within RDF KG
- Semantic Web Analysis with Flavour of Micro-Services [5]
    - Rest Interface for SANSA and Spark based Pipelines

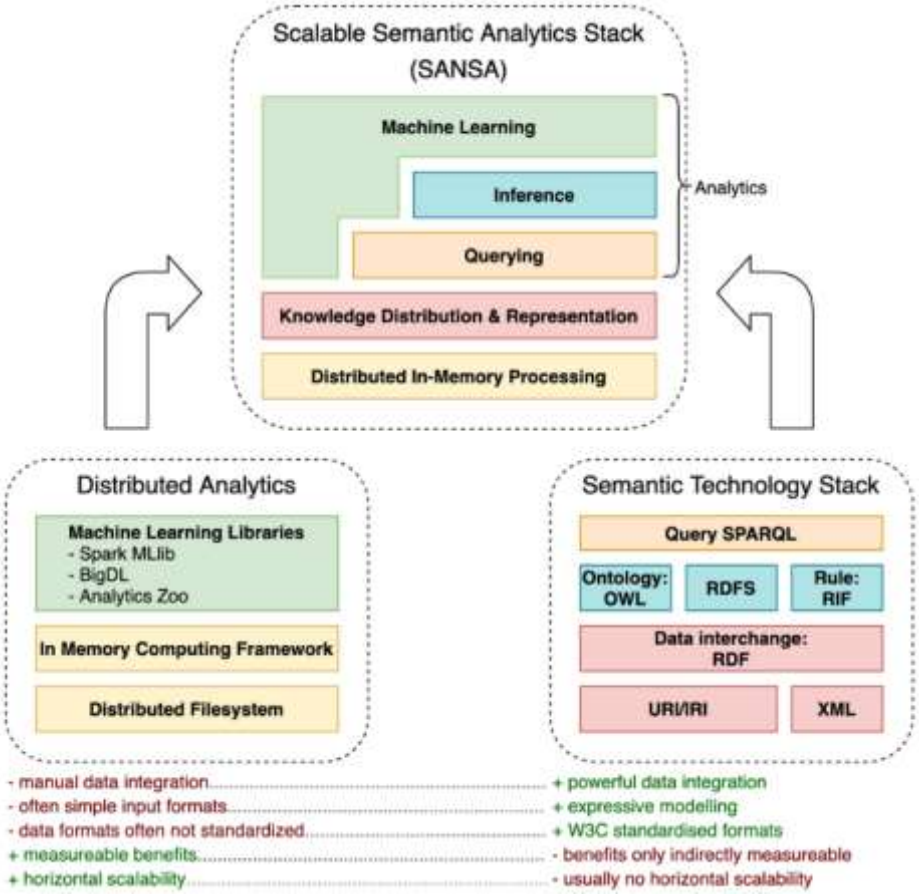


**Figure 10: The SANSA ecosystem**

More detailed examples can be found in the papers DistAD, DistRDF2ML, and DistAD. In the publications, you can also find further links to the software repositories. The entire content is fully open source under the Apache 2.0 license in the SANSA GitHub repository [5] [3] .

List of the modules which are evaluated in the context of SANSA are:

- Apache Livy
- Apache Spark (incl. MLlib)
- Apache Jena
- Hadoop File System (HDFS)
- HUE
- Apache Zeppelin

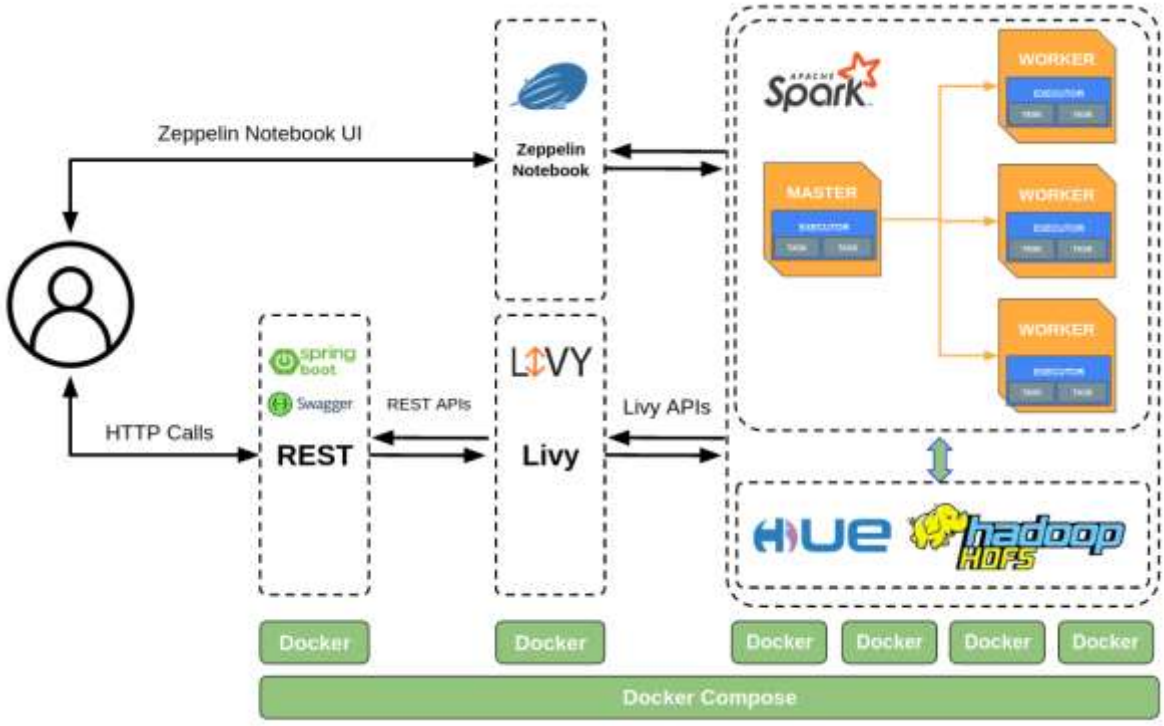


**Figure 11: The different modules of SANSA**

### 2.2.4 Analytics Dashboard

In task 5.4 a Data Analytics Dashboard has been developed for batch data visualisation that facilitates users to discover trends and get deeper insights into their data by combining and analysing different indicators produced by the data analytics tools/services defined in T5.2.

The target user of the tool are energy experts with high domain knowledge but low coding skills. Thus, the Dashboard has been defined to be intuitive and user-friendly providing a collection of reusable visualization templates/chart that can be easily integrated into customer-oriented cloud-based dashboard for predictive analytics and insights analysis.

The developed solution is based on the Generic Visualisation Toolbox built on the task T4.6. It is open source and it is available on: https://github.com/PLATOONProject/Analytics-Dashboard-WP5

More details about the development, functionalities and configuration of the dashboard are provided in T5.4.

## 2.3 Data Sharing Scenario

In this scenario Metadata registry is tested by registering two IDS Connectors to the deployed metadata registry and finally transferring a JSON between 2 IDS connector. Moreover, we test the Vocabulary Provider by testing the interaction between an IDS connector and Vocabulary Provider. Following images shows the architecture.

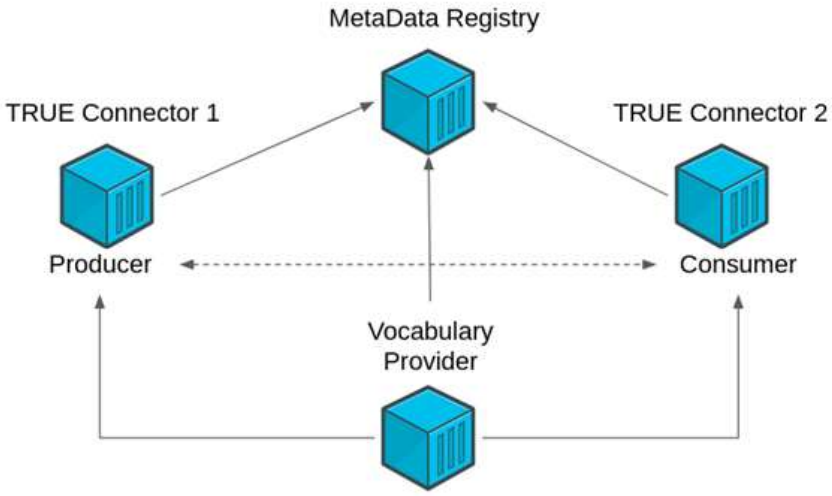


**Figure 12: Datasharing scenario overview**

### 2.3.1 IDS connectors

The Connector is the key building block of the International Data Spaces Architecture allowing the different entities that are part of the ecosystem to exchange, share and process digital content maintaining data sovereignty of the Data Owner. Depending on the type of configuration, the Connector's tamper- proof runtime hosts a variety of system services ensuring, for example, secure bidirectional communication, enforcement of content usage policies, system monitoring, and logging of content transactions for clearing purposes.

In particular, two different entities (Data provider/Data consumer) using two certified IDS connectors can exchange data, using secure transmission protocols, assuring the reliability of the parties involved defining specific usage control data policies. In PLATOON the pilots use, to exchange data in a trusted way, an open-source reference implementation of an IDS connector, the "TRUE" connector, developed by Engineering.

### 2.3.2 Metadata Registry

The Metadata Registry is a registry for resources and connectors derived from the International Data Spaces (IDS) Metadata Broker. In contrast to the IDS Metadata Broker, the Metadata registry has been tailored to keep the main functionalities of metadata handling for connectors and data/app resources and querying for the metadata. The Metadata registry is one of the main components in the PLATOON Marketplace. As shown in Figure 13, data consumers and data providers with an IDS connector can register their resources in the Metadata registry. The registry can be used to register, update, or unregister the connector or resource metadata. Note that the metadata registry does not serve the datasets themselves: querying is performed on metadata only.

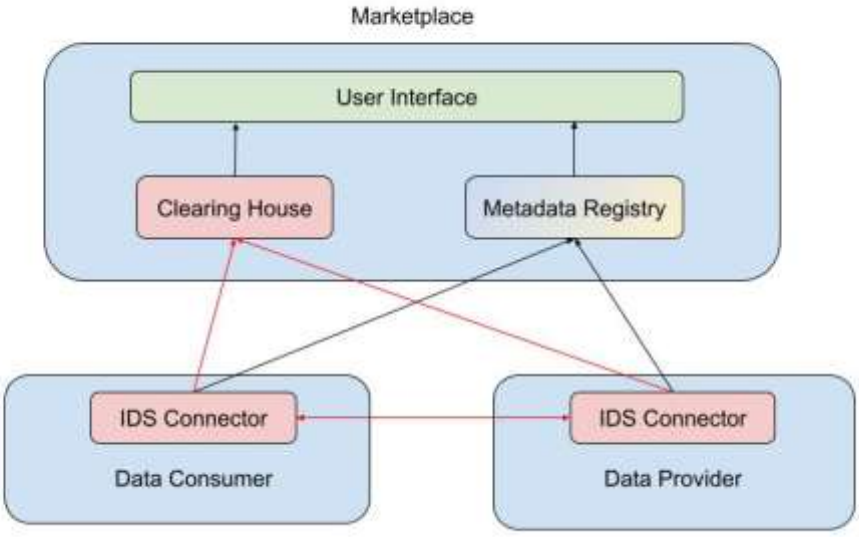


**Figure 13: Metadata registration**

### 2.3.3 IDS Vocabulary Provider

The PLATOON IDS Vocabulary Provider explained in D5.1 and D3.5 plays an essential role within the PLATOON reference architecture as it is the link between the Data Governance, Security, Privacy and Sovereignty layer (based on IDS reference architecture) and the Interoperability layer formed. The vocabulary provider provides direct Machine to Machine communication allowing to query the different vocabularies and exchange metadata through the IDS connectors. Several IDS messages have been implemented to allow to register to the vocabulary provider and send queries.

In addition, the Vocabulary Provider has a Graphical User Interface (GUI), where users can manage vocabularies (upload/upgrade/delete), search for specific terms, visualize the vocabularies in a network graph and execute SPARQL queries.

# 3 Testing Results

In this section, we briefly explain how the testing has been conducted and how it can be reproduced. Moreover, we explain all of our observations during testing process.

## 3.1 Edge-Scenario

### 3.1.1 Verification Plans

As already explained in this scenario, data should come from a (mock) edge device. The data should be transformed and stored in a persistent database, and finally a visualization tool should be able to visualize the streaming data. As mentioned earlier, the container mocking the edge device should be on a different network than database and visualization tool.

### 3.1.2 Integration

For integration, we had to install a virtual box to be able to have 2 different logical machines, one for the edge device and one for the database and visualization. For this, the tools Vagrant [6] and VirtualBox [7] needed to be installed on the server. Vagrant is an open-source software product for building and maintaining portable virtual software development environments. VirtualBox is a type-2 hypervisor for x86 virtualization developed by the Oracle Corporation. Vagrant requires VirtualBox to be able to create virtual machines.

After installing VirtualBox and Vagarant, the first step is creating a VagrantFile and running "vagrant up" to bring the virtual machine up and running. The content of the VagrantFile can be as follows:

```
Vagrant.configure("2") do |config|
      config.vm.box = "bento/ubuntu-20.04"
      config.vm.network "private_network", ip: "192.168.33.10"
end
```

After installing the virtual machine, one can connect to it via:

```
$ vagrant ssh
```

This command establishes a connection to the virtual machine. Now the edge device mock container can be started as a docker container via the following command:

```
$ docker run -e INFLUXDB_ADMIN_USER=admin -e INFLUXDB_DB=influx -e
INFLUX_DB_IP=xxx.xxx.xxx.xxx -e INFLUXDB_ADMIN_PASSWORD=password -p
8087:8086  andrejcampa/unittest edge
```

For configuration, one needs to know the IP, Username, Password, and Database name of InfluxDB. By running the above command, the mock device will start to submit data every minute to the specified database. The InfluxDB IP should be fetched after deploying database via the following procedure.

After running the Edge node, the database and visualisation node can be deployed. For this, we used docker-compose to bring those containers up and running. The content of the docker-compose.yml file is as follows:

```
version: '3.6'
services:
influxdb:
image: influxdb:1.8.10
container_name: influxdb
restart: always
environment:
- INFLUXDB_DB=influx
- INFLUXDB_ADMIN_USER=admin
- AUTH_ENABLED=false
```

```
ports:
- '8086:8086'
volumes:
- ./influxdb_data:/var/lib/influxdb

grafana:
image: grafana/grafana
container_name: grafana-server
restart: always
depends_on:
- influxdb
environment:
- GF_SECURITY_ADMIN_USER=admin
- GF_SECURITY_ADMIN_PASSWORD=admin
- GF_INSTALL_PLUGINS=
links:
- influxdb
ports:
- '3000:3000'
volumes:
- ./grafana_data:/var/lib/grafana
user: "root"

volumes:
grafana_data: {}
influxdb_data: {}
```

To bring these two containers up and running, we used the following command:

```
$ docker stack deploy -c docker-compose.yml edge
```

After running the above command, both containers will be deployed in docker swarm mode and will be available for Swarmpit. Note that Swarmpit also provides a user interface for deploying docker-compose files. The following figure shows Swarmpit after running the containers.

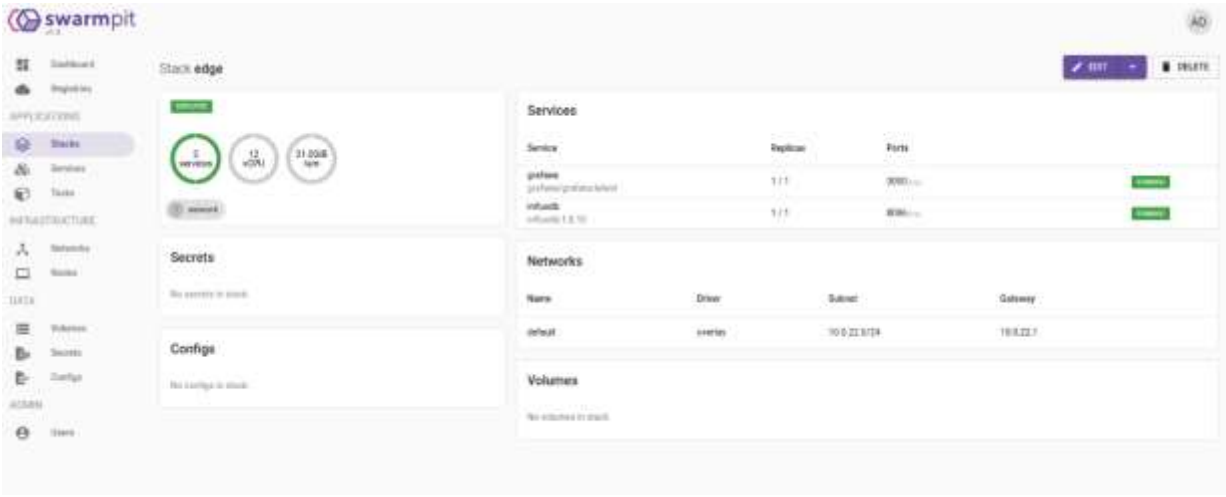


**Figure 14: Swarmpit after running the containers**

After deploying the containers, we need to connect Grafana to InfluxDB and configure a dashboard to visualize the streaming data. For this reason, first we need to:

- Access the Grafana UI via http://localhost:3000.
- In the left navigation of the Grafana UI, hover over the gear icon to expand the **Configuration** section. Click **Data Sources**.
- Click **Add data source**.
- Select **InfluxDB** from the list of available data sources.
- On the **Data Source configuration page**, enter a **name** for your InfluxDB data source.

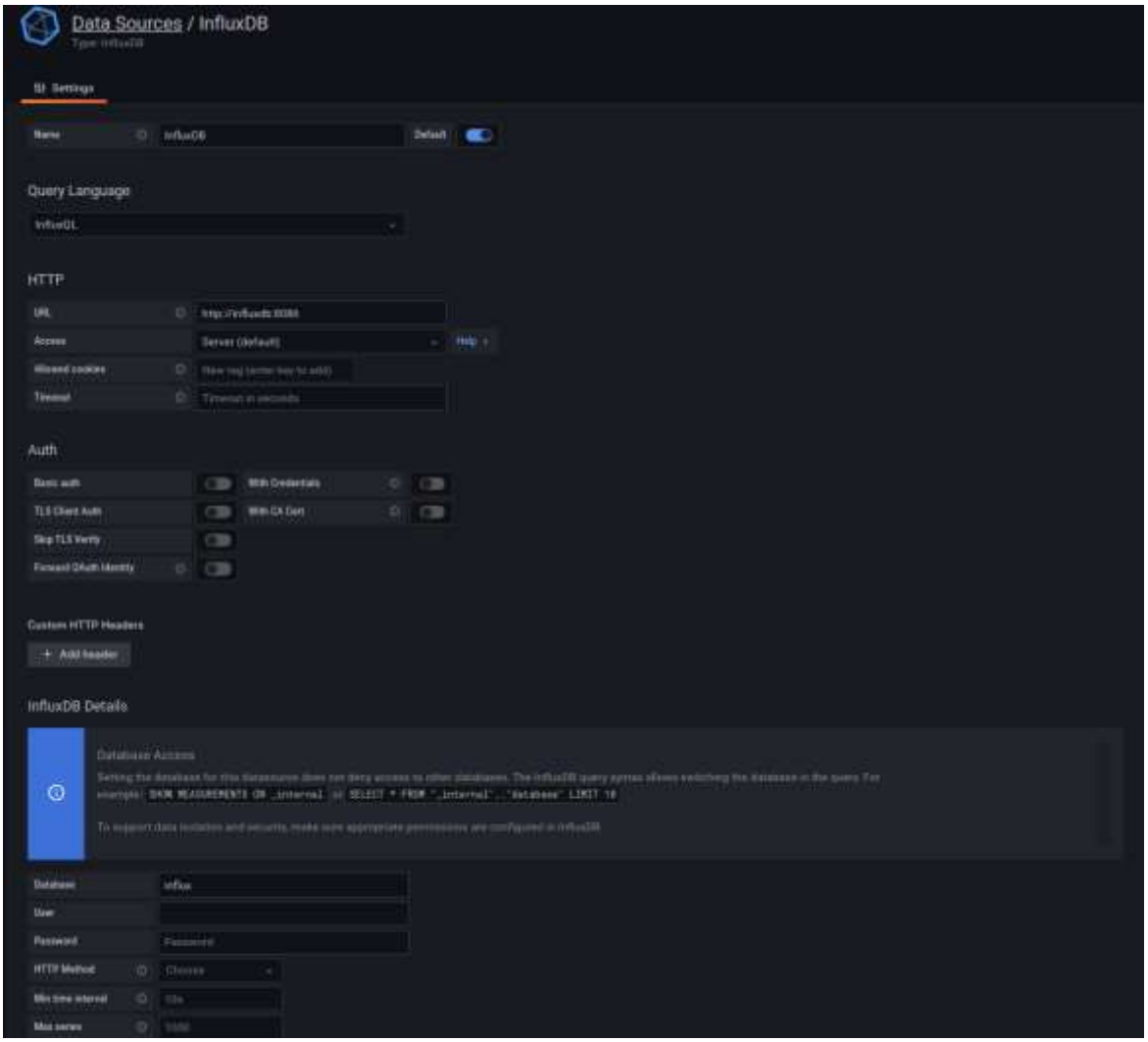


**Figure 15: InfluxDB data soruce**

After connecting Grafana to InfluxDB now one can create a panel to visualize data by selecting `Dashboard` from the left menu and selecting `Add a new Panel`. Figure 16 shows the query and configuration for creating a panel.

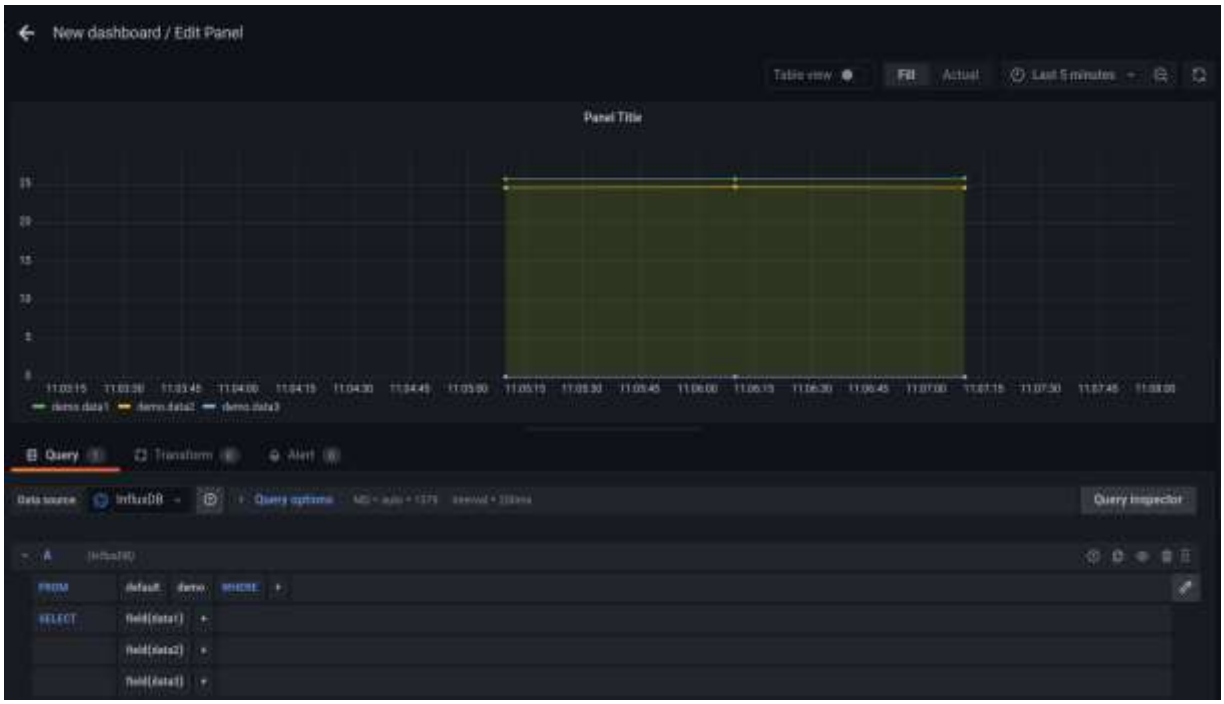


**Figure 16: Creation of a new panel**

### 3.1.3 Criteria and measures

For measurement, the following table summarizes the main characteristics of the components.

**Table 2: Main characteristics of the components**

| | Data Provider | InfluxDB | Visualization |
|---|---|---|---|
| Dockerized | Yes | Yes | Yes |
| Documentation | No | No | Yes |
| Open Source | Yes | Yes | Yes |
| Available on Platoon GitHub | No | No | Yes |
| Fully Configurable | Partially | Yes | No |

### 3.1.4 Results

After setting up all the containers, we can see the data is coming and being visualized in Grafana visualization tool [8].

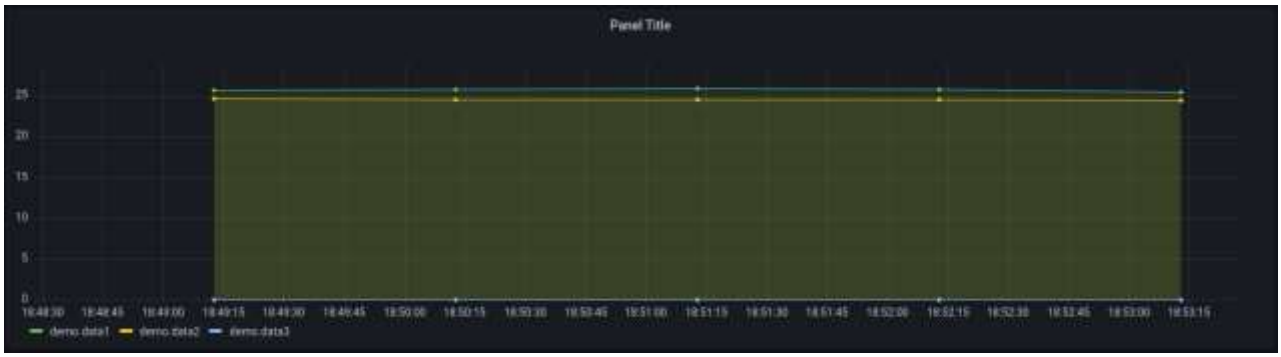


**Figure 17: Grafana Visualization tool**

The user can also check the logs of any tool via Swarmpit. The following image shows how the logs of InfluxDB container can be visualized.

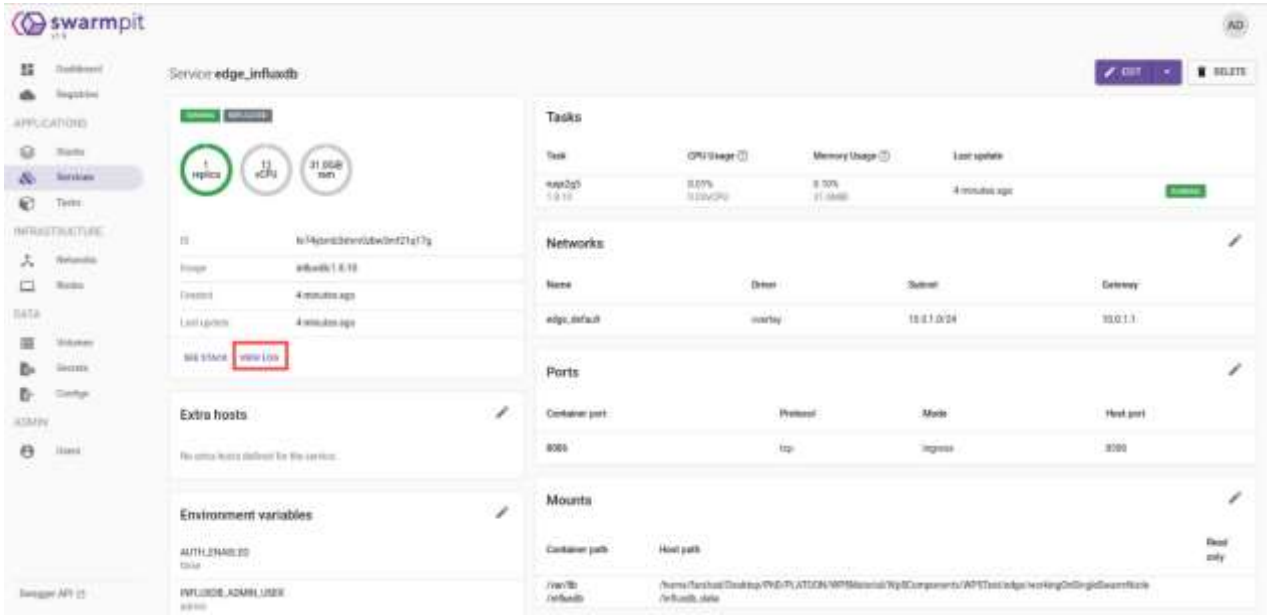


**Figure 18: Visualization of InfluxDB logs**

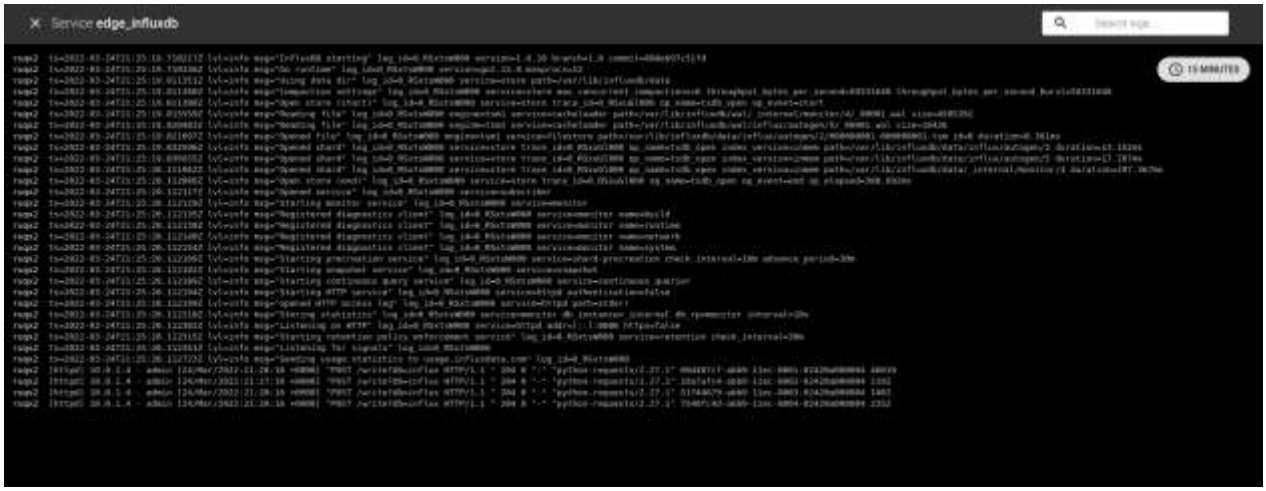


**Figure 19: InfluxDB logs**

The logs also approve that InfluxDB is receiving data from the edge node every minute.

## 3.2 Non-Edge Scenario

### 3.2.1 Verification Plans

For the Non-Edge scenario, historical data sources in CSV format should be converted to the semantified version via a semantic adaptor and served via a Federated Query Engine to the analytical tools. Finally, a visualizer component is used to present the results of the analytical tools.

However, as in this case only one data source exists (already explained in Section 2.2.1), we decided to divide the data vertically into 2 parts. The result was 2 files with different columns in each as follows:

CSV1:

- Id
- Date
- Hour

CSV2:

- Id
- Zone
- Source
- Connections

By applying such a division, the Federated Query Engine can be tested to see if it can merge results from multiple data sources. To test the pipeline with bigger data, some synthetic data has been generated with different sizes. The procedure is explained in Section 39.

### 3.2.2 Integration

For integration, one only needs to deploy all the containers via the docker-compose.yml file in the Appendix.

To make the process reproducible, we provided a Makefile with all the necessary commands as follows:

```
createFolders:
mkdir -p data/datanode
mkdir -p data/namenode
mkdir -p rdf-dump1
mkdir -p rdf-dump2
mkdir -p jars

downloadSANSA:
wget -O jars/sansa.jar
https://www.dropbox.com/s/syc9dv01867thtl/sansa.jar?dl=1
```

```
deploy:
docker-compose pull
docker network create -d overlay --attachable spark-net
docker stack deploy -c docker-compose.yml nonEdgeScenario
sleep 60
docker run -it --rm -v $(shell pwd)/jars:/data --net spark-net -e
"CORE_CONF_fs_defaultFS=hdfs://namenode:8020"
fmoghaddam/sansanamenode:v1 hdfs dfs -copyFromLocal -f /data /

semantify:
docker exec -it $(shell docker ps -q -f name=sdmrdfizer) python3 -
m rdfizer -c /data/config_rdfizer1.ini
docker exec -it $(shell docker ps -q -f name=sdmrdfizer) python3 -
m rdfizer -c /data/config_rdfizer2.ini
docker exec -it $(shell docker ps -q -f name=sdmrdfizer) python3
/data/scripts/load_to_virtuoso.py
docker exec -it $(shell docker ps -q -f name=sdmrdfizer) python3
/data/scripts/mapping_parser.py /data/config.ini
docker exec -it $(shell docker ps -q -f name=detrusty)
/DeTrusty/Scripts/restart_workers.sh

down:
docker stack rm nonEdgeScenario
docker network rm spark-net
```

The above commands orchestrate all the necessary steps to have all the containers up and running via:

```
$ make createFolders
$ make downloadSANSA
$ make deploy
$ make semantify
```

The first step of deployment is creating necessary folders which different containers will need during run time. So, the folder structure of the project should be as follows:

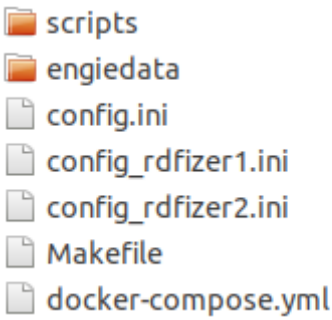


**Figure 20: Folder structure of Semantification Pipeline**

The "scripts" should be cloned from `https://github.com/SDM-TIB/PLATOONPipeline` and the rest of the files will be introduced later in this document.

The second step is downloading SANSA. This step downloads compiled version of SANSA which is needed for running analytics.

The third step is deploying all the containers to docker swarm. The following figure shows the running containers for the non-edge scenario.

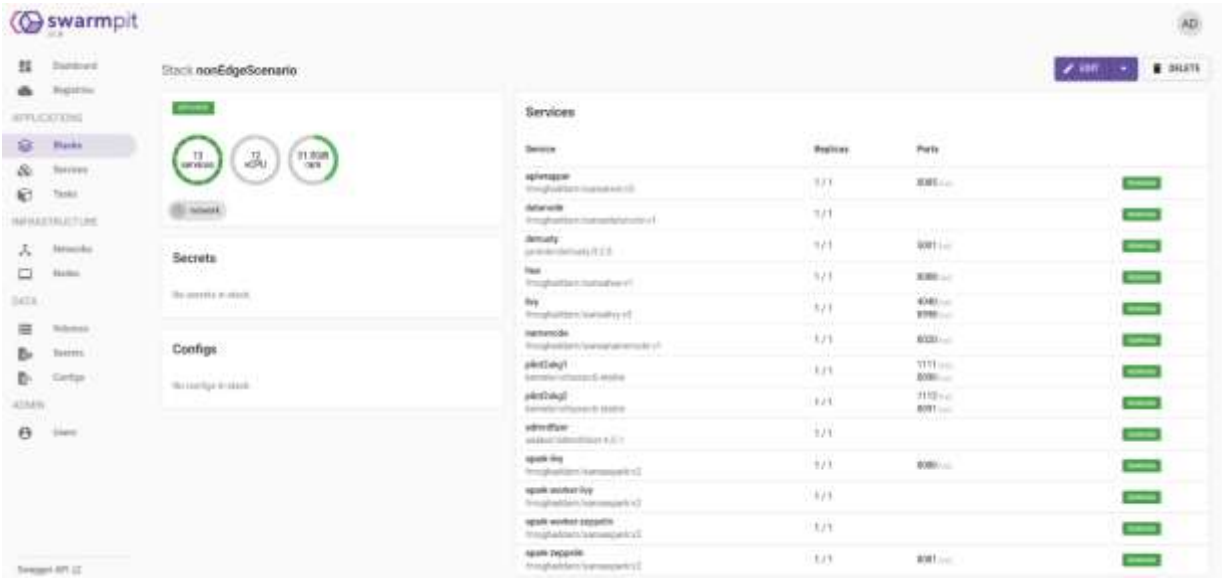


**Figure 21: Running containers for the non-edge scenario**

By having all the containers (13 containers in this case) up and running, the main process can be started by first semantifying the CSV data. For this reason, we used RDFizer [9] as semantic adaptor. RDFizer is able to convert CSV data to RDF by getting RML mapping rules and a config file. As we divided data into 2 parts, two similar config files (config_rdfizer1.ini and config_rdfizer2.ini) are needed as follows:

```
[default]
main_directory: /data/engiedata

[datasets]
number_of_datasets: 1
output_folder: ./rdf-dump1
```

```
all_in_one_file: yes
remove_duplicate: yes
name: EngieOccupancyPrediction
enrichment: yes
ordered: false
large_file: false

[dataset1]
name: Engie1
mapping: ${default:main_directory}/mapping1.ttl
```

```
[default]
main_directory: /data/engiedata

[datasets]
number_of_datasets: 1
output_folder: ./rdf-dump2
all_in_one_file: yes
remove_duplicate: yes
name: EngieOccupancyPrediction
enrichment: yes
ordered: false
large_file: false

[dataset1]
name: Engie2
mapping: ${default:main_directory}/mapping2.ttl
```

Moreover, 2 customized RML mapping rules (engiedata/ mapping1.ttl and engiedata/mapping2.ttl) are also needed as follows:

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ex: <http://example.com/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix ql: <http://semweb.mmlab.be/ns/ql#> .
```

```
@prefix engie: <http://engie/vocab/> .
@base <http://project-engie.eu/> .

<TripleMap1>
a rr:TriplesMap;
rml:logicalSource [
rml:source "./engiedata/occupancydata1";
rml:referenceFormulation ql:CSV
];
rr:subjectMap [
rr:template "http://project-engie.eu/building/{ID}";
rr:class engie:Building
];
rr:predicateObjectMap [
rr:predicate engie:DATE;
rr:objectMap [
rml:reference "DATE";
rr:datatype xsd:DATE
]
];


rr:predicateObjectMap [
rr:predicate engie:HOUR;
rr:objectMap [
rml:reference "HOUR";
rr:datatype xsd:int
]
].
```

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ex: <http://example.com/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix ql: <http://semweb.mmlab.be/ns/ql#> .
@prefix engie: <http://engie/vocab/> .
@base <http://project-engie.eu/> .

<TripleMap1>
a rr:TriplesMap;
rml:logicalSource [
rml:source "./engiedata/occupancydata2";
rml:referenceFormulation ql:CSV
];
rr:subjectMap [
rr:template "http://project-engie.eu/building/{ID}";
rr:class engie:Building
];
rr:predicateObjectMap [
rr:predicate engie:ZONE;
rr:objectMap [
rml:reference "ZONE";
rr:datatype xsd:String
]
];


rr:predicateObjectMap [
rr:predicate engie:SOURCE;
rr:objectMap [
rml:reference "SOURCE";
rr:datatype xsd:String
]
];


rr:predicateObjectMap [
rr:predicate engie:CONNECTIONS;
```

```
rr:objectMap [
rml:reference "CONNECTIONS";
rr:datatype xsd:int
]
].
```

By having the above-mentioned config files and mapping files, RDFizer is able to convert the respective CSV files to RDF via:

```
$ docker exec -it $(shell docker ps -q -f name=sdmrdfizer) python3
-m rdfizer -c /data/config_rdfizer1.ini
$ docker exec -it $(shell docker ps -q -f name=sdmrdfizer) python3
-m rdfizer -c /data/config_rdfizer2.ini
```

The above commands generated N-Triple (NT) files for the 2 CVS files. In this stage these NT files will be uploaded to 2 separate Virtuoso SPARQL endpoints. For this, the following command can be used:

```
$ docker exec -it $(shell docker ps -q -f name=sdmrdfizer) python3
/data/scripts/load_to_virtuoso.py
```

The last part of semantification is creating a config file for DeTrusty [10] (Federated Query Engine) and restarting the DeTrusty process via:

```
$ docker exec -it $(shell docker ps -q -f name=sdmrdfizer) python3
/data/scripts/mapping_parser.py /data/config.ini
$ docker exec -it $(shell docker ps -q -f name=detrusty)
/DeTrusty/Scripts/restart_workers.sh
```

At this stage, an endpoint will be accessible for any analytical tool to fetch data via a SPARQL query. The following command shows an example:

```
$ curl -X POST -d "query=SELECT ?s ?p ?o WHERE { ?s ?r
<http://engie/vocab/Building>. ?s ?p ?o. } LIMIT 1"
localhost:5001/sparql
```

By now an endpoint is accessible to serve data. In this scenario, SANSA has been used as an analytical tool. SANSA (as explained in 2.2.3) offers REST APIs for applying analytics to the data. In this scenario, SANSA provided 2 new endpoints, one for fetching data from the DeTrusty endpoint, applying analytics, and writing the result into HDFS, and the second endpoint for serving the analytical result as a JSON to the visualization tool. The following images show the provided APIs.

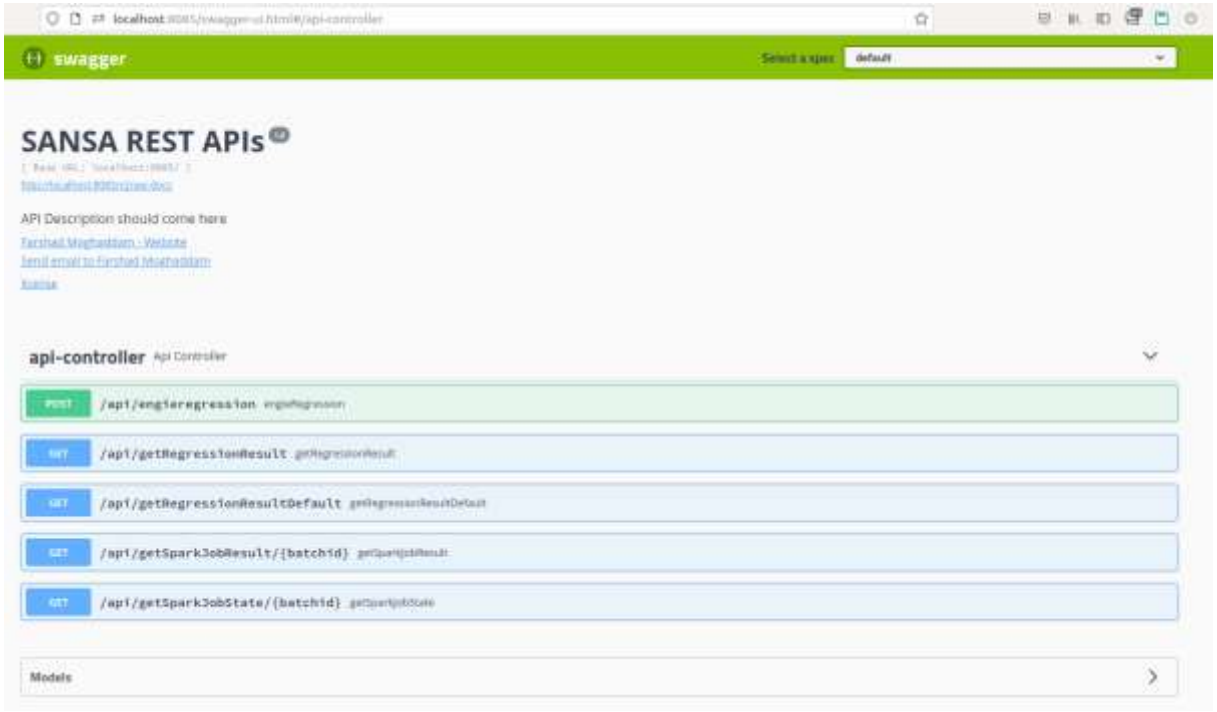


**Figure 22: APIs provided by SANSA**

To run Analytics, one should use the following endpoint:

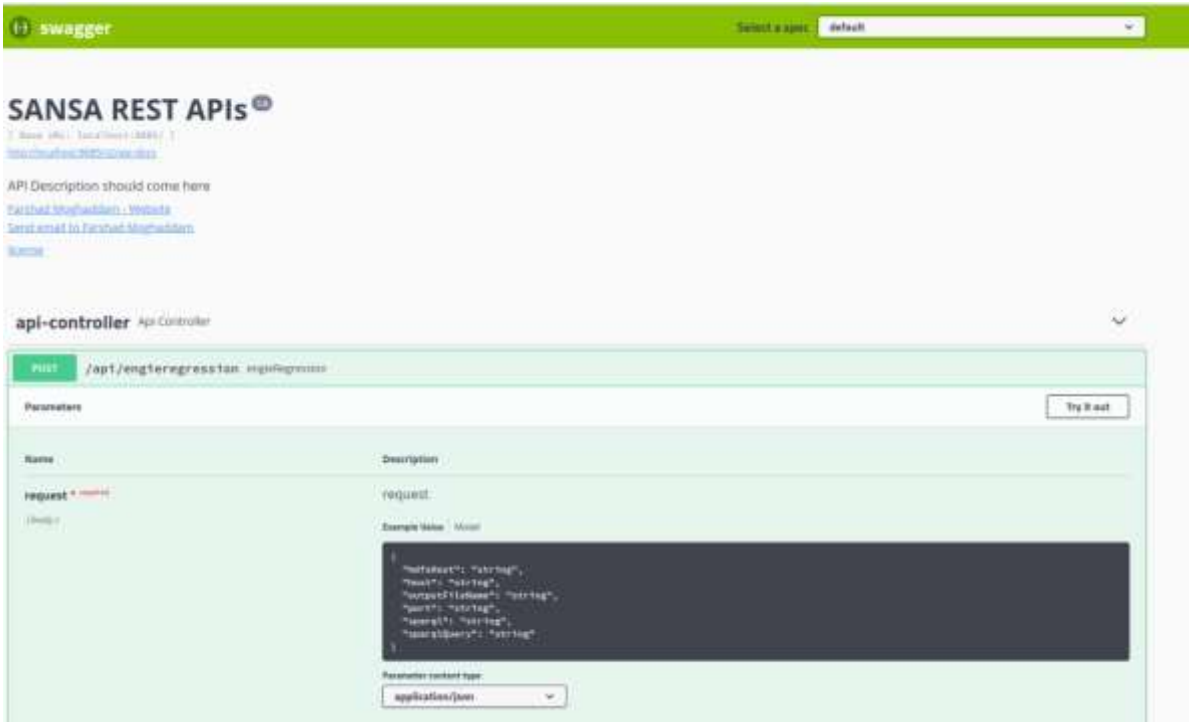


**Figure 23: Endpoint for analytics**

The input json sample is as follows:

```
{
  "hdfsHost": "hdfs://namenode:8020/",
```

```
"host": "http://detrusty",
"outputFileName": "output.json",
"port": "5000",
"sparql": "sparql",
"sparqlQuery": "SELECT ?s ?p ?o WHERE { ?s ?r <http://engie/vocab/Building>. ?s ?p ?o. }"
}
```


Running this API will produce a json file as a result of the anlaytics process. This API returns the address of the result file. Now for fetching the result one can use "`/api/getRegressionResult`". This API returns the result of analytics as JSON. In this stage, the mentioned API can be used by graphical dashboard for visualization. The following image shows the graphical dashboard.

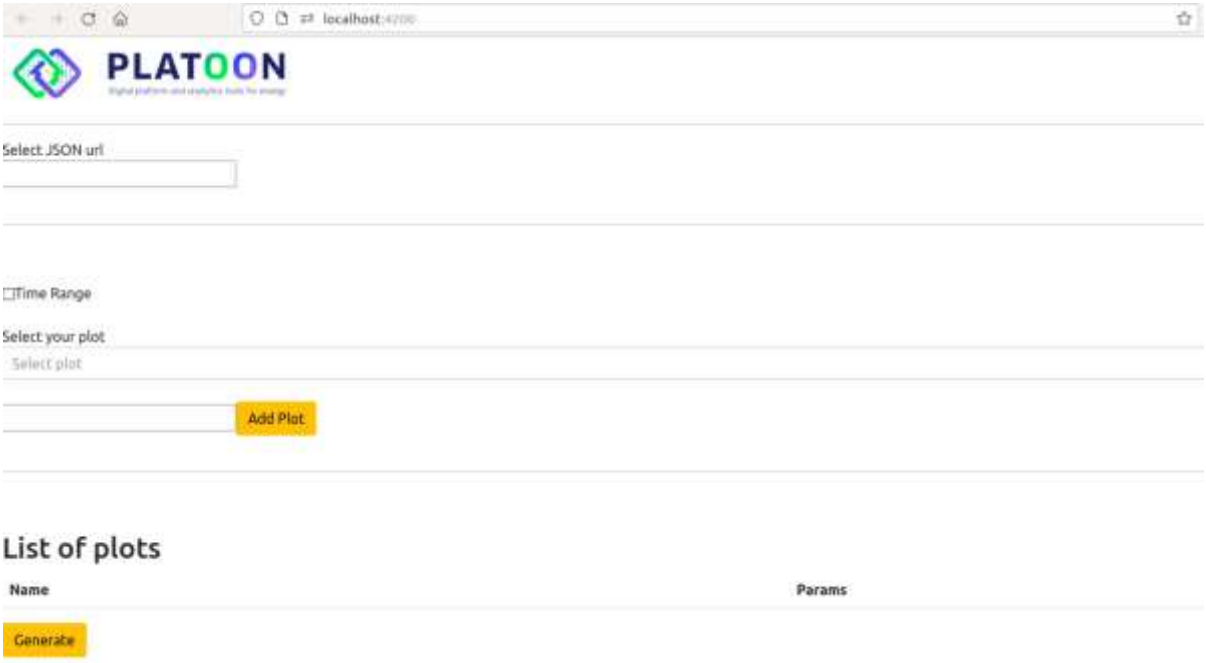


**Figure 24: Graphical Dashboard**

User should provide the API URL for fetching the results. In this scenario the URL is `http://apiwrapper:8085/api/getRegressionResultDefault`. Then user must select the dashboard type. Finally, by pressing generate can visualize the result


**Figure 25: Visualization of the result**

For more information about the visualization dashboard please visit https://github.com/PLATOONProject/Analytics-Dashboard-WP5

### 3.2.3 Criteria and measures

For measurement, the following table summarizes the main characteristics of the components.

**Table 3: Main characteristics of the components**

| | Dockerized | Documentation | Open Source | Available on Platoon GitHub | Fully Configurable |
|---|---|---|---|---|---|
| Apache Spark | Yes | Yes | Yes | No | Yes |
| Apache Hadoop | Yes | Yes | Yes | No | Yes |
| Apache Livy | Yes | Yes | Yes | No | Yes |
| Hue | Yes | Yes | Yes | No | Yes |
| SANSA REST | Yes | Yes | Yes | No | Yes |
| RDFizer | Yes | Yes | Yes | Yes | Yes |
| DeTrusty | Yes | Yes | Yes | Yes | Yes |
| Virtuoso | Yes | Yes | Yes | No | Yes |
| Visualization Dashboard | Yes | Yes | Yes | Yes | Yes |

### 3.2.4 Results

After setting up all the containers and running respective commands as expected the dashboard will show the analytical results.


**Figure 26: Dashboard for analytical result**

To show the performance of the semantification pipeline, we tried to generate some synthetic data from the original data to be able to put RDFizer [9] and Detrusty [10] under stress. For this reason, we generated more data by shifting the date in the original data. Following table shows the statistics of data.

**Table 4: Data statistics**

| | Number of Rows | File Size | Knowlege Graph Size |
|---|---|---|---|
| **Original Data** | 31066 ~ 30K | 1.4 MB | 30 MB |
| **Synthetic Data 1** | 341716 ~ 350K | 16.8 MB | 300 MB |
| **Synthetic Data 2** | 963016 ~ 1M | 47.8 MB | 1 GB |

Having data with different sizes can reveal the performance of the semantification pipeline. Following table shows the runtime (in second) for RDFizer semantifer, Loading data to triple store, and Querying FQP. We keep the SPAQL query as it is defined in section 3.2.2.

**Table 5: Runtime in seconds**

| | RDFizer | Loading to TripleStore | FQP |
|---|---|---|---|
| **Original Data** | 5.70s | 4.10s | 17.61s |
| **Synthetic Data 1** | 40.74s | 39.67s | 251.62s |
| **Synthetic Data 2** | 110.73s | 116.58s | 715.16s |

As it can be seen, increasing dataset size, increase the run time of semantification pipeline. However, it should be noted that, this change is not exponential nor even linear. Which means increasing the dataset size by factor of 10 does not increase the runtime by factor of 10, but less than 10.

## 3.3 Data Sharing Scenario

### 3.3.1 Verification Plans

As already explained in this scenario Metadata registry is tested by registering two IDS Connectors to the deployed metadata registry and finally transferring a JSON between 2 IDS connector. For this reason, we use 2 TRUE Connectors as IDS connector. One connector as producer and one as consumer. First the producer register itself to the metadata registry and then the consumer can search metadata registry for the provider and request a JSON content.

Moreover, we test the Vocabulary Provider by testing the interaction between an IDS connector and Vocabulary Provider.

### 3.3.2 Integration

For integration, one needs to first clone metadata registry and TRUE Connectors via:

```
$ git clone https://github.com/PLATOONProject/Metadata-Registry.git
$ git clone https://github.com/Engineering-Research-and-Development/true-connector.git
```

After cloning repositories, the user should follow the instructions from the respective GitHub repositories to bring both components up and running. After running, meta data registry will be available at `https://localhost` and producer connector will be available at `https://localhost:8083` and the consumer connector will be available at `https://localhost:8084`

#### 3.3.2.1 Registering IDS connector

The first step is registering the producer connector to the meta data registry. This step happens by running the following snippet:

```
curl --location --request POST 'https://localhost:8084/proxy' \
--header 'Content-Type: text/plain' \
--data-raw '{
"multipart": "mixed",
"Forward-To": "https://METADATA_REGISTRY_IP/infrastructure",
"messageType": "ConnectorUpdateMessage",
"payload": {
"@context": {
"xsd": "http://www.w3.org/2001/XMLSchema#",
"ids": "https://w3id.org/idsa/core/",
"idsc": "https://w3id.org/idsa/code/"
},
```

"@type": "ids:BaseConnector",

"@id": "https://w3id.org/engrd/connector/",

"ids:description": [{

"@value": "Data Provider Connector description",

"@type": "http://www.w3.org/2001/XMLSchema#string"

}],

"ids:title": [{

"@value": "Data Provider Connector title",

"@type": "http://www.w3.org/2001/XMLSchema#string"

}],

"ids:resourceCatalog": [{

"@type": "ids:ResourceCatalog",

"@id": "https://w3id.org/idsa/autogen/resourceCatalog/193f2e86-b44e-49d9-9cfb-ccb30d760c85",

"ids:offeredResource": [{

"@type": "ids:TextResource",

"@id": "https://w3id.org/idsa/autogen/textResource/9fb55fe4-b36c-48e6-9cab-0b72e61b91b3",

"ids:language": [{

"@id": "https://w3id.org/idsa/code/EN"

}, {

"@id": "https://w3id.org/idsa/code/IT"

}],

"ids:version": "1.0.0",

"ids:description": [{

"@value": "Default resource description",

"@type": "http://www.w3.org/2001/XMLSchema#string"

}],

"ids:theme": [],

"ids:title": [{

"@value": "Default resource",

"@type": "http://www.w3.org/2001/XMLSchema#string"

}],

"ids:created": {

"@value": "2022-03-15T09:50:40.951+01:00",

"@type": "http://www.w3.org/2001/XMLSchema#dateTimeStamp"

},
"ids:modified": {
"@value": "2022-03-15T09:50:40.951+01:00",
"@type": "http://www.w3.org/2001/XMLSchema#dateTimeStamp"
},
"ids:representation": [{
"@type": "ids:TextRepresentation",
"@id": "https://w3id.org/idsa/autogen/textRepresentation/5f952009-d300-4229-bff5-fbf2b144c268",
"ids:instance": [{
"@type": "ids:Artifact",
"@id": "http://w3id.org/engrd/connector/artifact/1",
"ids:creationDate": {
"@value": "2022-03-15T09:50:40.416+01:00",
"@type": "http://www.w3.org/2001/XMLSchema#dateTimeStamp"
}
}],
"ids:language": {
"@id": "https://w3id.org/idsa/code/EN"
},
"ids:created": {
"@value": "2022-03-15T09:50:41.733+01:00",
"@type": "http://www.w3.org/2001/XMLSchema#dateTimeStamp"
}
}],
"ids:contractOffer": [{
"@type": "ids:ContractOffer",
"@id": "https://w3id.org/idsa/autogen/contractOffer/7473805d-198b-4bb0-9178-03a21ddfc177",
"ids:permission": [{
"@type": "ids:Permission",
"@id": "https://w3id.org/idsa/autogen/permission/bc84532b-fc86-434c-aaea-d2d40f6a57f3",
"ids:target": {
"@id": "http://w3id.org/engrd/connector/artifact/1"
},

```
"ids:description": [{
"@value": "provide-access",
"@type": "http://www.w3.org/2001/XMLSchema#string"
}],
"ids:action": [{
"@id": "https://w3id.org/idsa/code/USE"
}],
"ids:title": [],
"ids:assignee": [{
"@id": "https://assignee.com"
}],
"ids:assigner": [{
"@id": "https://assigner.com"
}],
"ids:preDuty": [],
"ids:postDuty": [],
"ids:constraint": [{
"@type": "ids:Constraint",
"@id": "https://w3id.org/idsa/autogen/constraint/28ceba5b-b262-4acc-8da0-93c8c59dfdf2",
"ids:leftOperand": {
"@id": "https://w3id.org/idsa/code/POLICY_EVALUATION_TIME"
},
"ids:rightOperand": {
"@value": "2022-03-08T08:50:40Z",
"@type": "xsd:datetime"
},
"ids:operator": {
"@id": "https://w3id.org/idsa/code/AFTER"
}
}, {
"@type": "ids:Constraint",
"@id": "https://w3id.org/idsa/autogen/constraint/f3d6ab01-96c1-4767-a31f-
c1c1030d73c8",
"ids:leftOperand": {
"@id": "https://w3id.org/idsa/code/POLICY_EVALUATION_TIME"
```

```
},
"ids:rightOperand": {
"@value": "2022-04-15T08:50:40Z",
"@type": "xsd:datetime"
},
"ids:operator": {
"@id": "https://w3id.org/idsa/code/BEFORE"
}
}]
}],
"ids:provider": {
"@id": "https://provider.com"
},
"ids:consumer": {
"@id": "https://consumer.com"
},
"ids:contractDate": {
"@value": "2022-03-15T09:50:41.565+01:00",
"@type": "http://www.w3.org/2001/XMLSchema#dateTimeStamp"
},
"ids:prohibition": [],
"ids:obligation": []
}],
"ids:resourceEndpoint": [],
"ids:paymentModality": [],
"ids:sample": [],
"ids:resourcePart": [],
"ids:contentPart": [],
"ids:defaultRepresentation": [],
"ids:keyword": [{
"@value": "Engineering Ingegneria Informatica SpA",
"@type": "http://www.w3.org/2001/XMLSchema#string"
}, {
"@value": "TRUEConnector",
"@type": "http://www.w3.org/2001/XMLSchema#string"
```

```
}],
"ids:temporalCoverage": [],
"ids:spatialCoverage": [],
"ids:contentType": {
"@id": "https://w3id.org/idsa/code/SCHEMA_DEFINITION"
}
}],
"ids:requestedResource": []
}],
"ids:maintainer": {
"@id": "http://provider.maintainerURI.com"
},
"ids:curator": {
"@id": "http://provider.curatorURI.com"
},
"ids:hasEndpoint": [],
"ids:hasDefaultEndpoint": {
"@type": "ids:ConnectorEndpoint",
"@id": "https://88.152.187.228:8090/",
"ids:accessURL": {
"@id": "https://88.152.187.228:8090/"
},
"ids:endpointInformation": [],
"ids:endpointDocumentation": []
},
"ids:hasAgent": [],
"ids:securityProfile": {
"@id": "https://w3id.org/idsa/code/BASE_SECURITY_PROFILE"
},
"ids:extendedGuarantee": [],
"ids:inboundModelVersion": ["4.1.0"],
"ids:outboundModelVersion": "4.1.0"
}
}'
```

METADATA_REGISTRY_IP should be the IP address of the metadata registry. The payload part of the request is the description of the provider connector which can be access via `https://localhost:8090/`

The result of the above request would be:


--CY5lyunH_ZcoSvys8O4in8bPoJlPZa

Content-Disposition: form-data; name="header"

Content-Length: 2551

Content-Type: application/ld+json

{

"@context" : {

"ids" : "https://w3id.org/idsa/core/",

"idsc" : "https://w3id.org/idsa/code/"

},

"@type" : "ids:MessageProcessedNotificationMessage",

"@id" : "https://w3id.org/idsa/autogen/messageProcessedNotificationMessage/18feb970-0c09-4bf9-9106-e4b9cc75c179",

"ids:issuerConnector" : {

"@id" : "https://localhost/"

},

"ids:senderAgent" : {

"@id" : "https://www.iais.fraunhofer.de"

},

"ids:securityToken" : {

"@type" : "ids:DynamicAttributeToken",

"@id" : "https://w3id.org/idsa/autogen/dynamicAttributeToken/b5d58cc9-3524-45d5-a313-4872cc2b5281",

"ids:tokenValue" :
"eyJ0eXAiOiJhdCtqd3QiLCJraWQiOiJUQ1VGZUNOYXphbEtIZzlLenJ6TElBelJXVE1ERFdTYTdMY005WndITXlvIiwiYWxnIjoiUlMyNTYifQ.eyJhdWQiOiJpZHNjOklEU19DT05ORUNUT1JTX0FMTCIsImlzcyI6Imh0dHBzOi8vZGFwcy5haXNlYy5mcmF1bmhvZmVyLmRlIiwic3ViIjoiOTI6MTQ6RTc6QUM6MTA6MjI6RjI6Q0M6MDU6NkU6MkE6MkI6OEQ6REI6MTA6RDY6NEQ6QTA6QTE6NTM6a2V5aWQ6Q0I6OEM6Qzc6QjY6ODU6Nzk6QTg6MjM6QTY6Q0I6MTU6QUI6MTc6NTA6MkY6RTY6NjU6NDM6NUQ6RTgiLCJuYmYiOjE2NDczMzQ0NjUsImlhdCI6MTY0NzMzNDQ2NSwianRpIjoiTVRNMU5qQTJNVGN5T1RVeE16WTNNREExTXc9PSIsImV4cCI6MTY0NzMzODA2NSwiY2xpZW50X2lkIjoiOTI6MTQ6RTc6QUM6MTA6MjI6RjI6Q0M6MDU6NkU6MkE6MkI6OEQ6REI6MTA6RDY6NEQ6QTA6QTE6NTM6a2V5aWQ6Q0I6OEM6Qzc6QjY6ODU6Nzk6QTg6MjM6QTY6Q0I6MTU6QUI6MTc6NTA6MkY6RTY6NjU6NDM6NUQ6RTgiLCJzZWN1cml0eVByb2ZpbGUiOiJpZHNjOkJBU0VfU0VDVVJJVFlfUFJPRklMRSIsIn

JlZmVycmluZ0Nvbm5lY3RvciI6Imh0dHA6Ly9icm9rZXIuaWRzLmlzc3QuZnJhdW5ob2Zlci5kZS5kZW1vIiwiQHR5cGUiOiJpZHM6RGF0UGF5bG9hZCIsIkBjb250ZXh0IjoiaHR0cHM6Ly93M2lkLm9yZy9pZHNhL2NvbnRleHRzL2NvbnRleHQuanNvbmxkIiwidHJhbnNwb3J0Q2VydHNTaGEyNTYiOiI5NzRlNjMyNGYxMmYxMDkxNmY0NmJmZGViMThiOGRkNmRhNzhjYzZhNmEwNTY2MDMxZmE1ZjFhMzllYzhlNjAwIiwic2NvcGVzIjpbImlkc2M6SURTX0NPTk5FQ1RPUl9BVFRSSUJVVEVTX0FMTCJdfQ.RYYjc-bb0cx25oRUBqkY4FkguBZ5x68KK-LimshkV6Mc0ZAfxxl4WCFziw17NGS0rI9WCaFRiASMtgM8Qtkyv3XwjjZqaCS-SCATQVCenmETdZqjCBEvATJjmWBWq4qLRit4J3Cm5v_b3tVc5psq8zaMgVXoxUP22Qpk1wHb1E0RaPrVOO5b_Fv1_HMWm-sPgVSPnvDExQQyOBLU7PqZ9mSHq-TnskgoSrG5E4W4LSCBCpl9D8cqji_1EGZICpQGY44Zs5cuXRjY0_SmuwAixIIg41s-MZWxO1O2Kahk3M31-3vhKzy9CvvOdF3Eee5l9a8xa2O9YMhbRDzu2OogvQ",

"ids:tokenFormat" : {

"@id" : "https://w3id.org/idsa/code/JWT"

}

},

"ids:issued" : {

"@value" : "2022-03-15T09:02:29.139Z",

"@type" : "http://www.w3.org/2001/XMLSchema#dateTimeStamp"

},

"ids:correlationMessage" : {

"@id" : "https://w3id.org/idsa/autogen/connectorUpdateMessage/6db56a02-aa41-46a2-adb9-46294f2dc2cf"

},

"ids:recipientConnector" : [ ],

"ids:recipientAgent" : [ ],

"ids:modelVersion" : "4.0.3"

}

--CY5lyunH_ZcoSvys8O4in8bPoJlPZa--

`MessageProcessedNotificationMessage` shows that the connector has been registered.

### 3.3.2.2 Query Metadata Registry

For searching Metadata registry, the user should use IDS `QueryMessage`. The following snippets search metadata registry for any connector of type `BaseConnector`:

```
curl --location --request POST 'https://localhost:8084/proxy' \
--header 'Content-Type: text/plain' \
--data-raw '{
"multipart": "mixed",
```

```
"Forward-To": "https://METADATA_REGISTRY_IP/infrastructure",
"messageType": "QueryMessage",
"payload":"PREFIX ids: <https://w3id.org/idsa/core/>
SELECT DISTINCT ?subject
WHERE {
graph ?g {
?subject a ids:BaseConnector.
}
}"
}'
```

METADATA_REGISTRY_IP should be the IP address of the metadata registry.

The result of the above request would be:

```
--lUMXu4b--6LSuVgK7BSD2CEleGxlxN7Z7SfRZ
Content-Disposition: form-data; name="header"
Content-Length: 2497
Content-Type: application/ld+json



{
  "@context" : {
    "ids" : "https://w3id.org/idsa/core/",
    "idsc" : "https://w3id.org/idsa/code/"
  },
  "@type" : "ids:ResultMessage",
  "@id" : "https://w3id.org/idsa/autogen/resultMessage/348b5da0-5900-4847-ab98-9cc4d9fecb41",
  "ids:recipientConnector" : [ ],
  "ids:senderAgent" : {
    "@id" : "https://www.iais.fraunhofer.de"
  },
  "ids:recipientAgent" : [ ],
  "ids:securityToken" : {
    "@type" : "ids:DynamicAttributeToken",
```

"@id" : "https://w3id.org/idsa/autogen/dynamicAttributeToken/8ed8e067-3009-4712-ac8a-087640b04ffa",

"ids:tokenValue" :
"eyJ0eXAiOiJhdCtqd3QiLCJraWQiOiJUQ1VGZUNOYXphbEtIZzlLenJ6TElBelJXVE1E
RFdTYTdMY005WndITXlvIiwiYWxnIjoiUlMyNTYifQ.eyJhdWQiOiJpZHNjOklEU19D
T05ORUNUT1JTX0FMTCIsImlzcyI6Imh0dHBzOi8vZGFwcy5haXNlYy5mcmF1bmhvZ
mVyLmRlIiwic3ViIjoiOTI6MTQ6RTc6QUM6MTA6MjI6RjI6Q0M6MDU6NkU6MkE6
MkI6OEQ6REI6MTA6RDY6NEQ6QTA6QTE6NTM6a2V5aWQ6Q0I6OEM6Qzc6QjY6
ODU6Nzk6QTg6MjM6QTY6Q0I6MTU6QUI6MTc6NTA6MkY6RTY6NjU6NDM6NUQ
6RTgiLCJuYmYiOjE2NDc1MTI3MDksImlhdCI6MTY0NzUxMjcwOSwianRpIjoiTVRB
M09ETTVOemM1TmpFMk1UTXdORGt6TVRjPSIsImV4cCI6MTY0NzUxNjMwOSwiY
2xpZW50X2lkIjoiOTI6MTQ6RTc6QUM6MTA6MjI6RjI6Q0M6MDU6NkU6MkE6MkI6
OEQ6REI6MTA6RDY6NEQ6QTA6QTE6NTM6a2V5aWQ6Q0I6OEM6Qzc6QjY6ODU6
Nzk6QTg6MjM6QTY6Q0I6MTU6QUI6MTc6NTA6MkY6RTY6NjU6NDM6NUQ6RTgi
LCJzZWN1cml0eVByb2ZpbGUiOiJpZHNjOkJBU0VfU0VDVVJJVFlfUFJPRklMRSIsInJ
lZmVycmluZ0Nvbm5lY3RvciI6Imh0dHA6Ly9icm9rZXIuaWRzLmlzc3QuZnJhdW5ob2Z
lci5kZS5kZW1vIiwiQHR5cGUiOiJpZHM6RGF0UGF5bG9hZCIsIkBjb250ZXh0IjoiaHR0
cHM6Ly93M2lkLm9yZy9pZHNhL2NvbnRleHRzL2NvbnRleHQuanNvbmxkIiwidHJhbn
Nwb3J0Q2VydHNTaGEyNTYiOiI5NzRlNjMyNGYxMmYxMDkxNmY0NmJmZGViMT
hiOGRkNmRhNzhjYzZhNmEwNTY2MDMxZmE1ZjFhMzllYzhlNjAwIiwic2NvcGVzIjp
bImlkc2M6SURTX0NPTk5FQ1RPUl9BVFRSSUJVVEVTX0FMTCJdfQ.Lqe2HpN2U1Pl
UgB3VoAqiHGxvJK_qPzYBjuS83mQ-Vf7GlYwy4QwZdy-JrDb3h-PA0-
EbW0g8S9syxnO_2i-O6haS-i0yaRpRZSzLhI-gel-
5tCHqVDl8PaiQuYS_ufizxzMGRfGgI5DQS7_Od6cAGG8zJUOsHgJR07Jq1L3yH_zQnU
-edxAruZ6Mhgp-i_okm-
laQdbqfVeHLDjDSHlMsNz4KgY4grbXs_t0MFYnA1YBuUGA66XiC5c6mmYOQzj44y_
PuO4BR2XTXMcuckvuCbxZbjO7iEWBkuMACXr3qVx4AUSKDiyLyYPzF9KOrG7aUgt
BEVKt4SFjbvuelBLGw",

"ids:tokenFormat" : {

"@id" : "https://w3id.org/idsa/code/JWT"

}

},

"ids:correlationMessage" : {

"@id" : "https://w3id.org/idsa/autogen/queryMessage/e3c9faed-955c-4758-a7f8-e8bf6c79ab76"

},

"ids:modelVersion" : "4.0.3",

"ids:issued" : {

"@value" : "2022-03-17T10:58:36.237Z",

"@type" : "http://www.w3.org/2001/XMLSchema#dateTimeStamp"

},

"ids:issuerConnector" : {

"@id" : "https://localhost/"

```
  }
}
--lUMXu4b--6LSuVgK7BSD2CEleGxlxN7Z7SfRZ
Content-Disposition: form-data; name="payload"
Content-Length: 50



?subject
<https://localhost/connectors/-832458571>
--lUMXu4b--6LSuVgK7BSD2CEleGxlxN7Z7SfRZ--
```

As it can be seen the result of the query is a URL which represents the registered provider connector (https://localhost/connectors/-832458571). To see all the offered artifacts from this connector, the user can use the following snippet:

```
curl --location --request POST 'https://localhost:8084/proxy' \
--header 'Content-Type: text/plain' \
--data-raw '{
"multipart": "mixed",
"Forward-To": "https://METADATA_REGISTRY_IP/infrastructure",
"messageType": "QueryMessage",
   "payload":"PREFIX ids: <https://w3id.org/idsa/core/>
SELECT ?subject ?predicate ?object
WHERE {
  graph ?g {
    ?subject a ids:Artifact.
    ?subject ?predicate ?object
  }
}"
}'
```

METADATA_REGISTRY_IP should be the IP address of the metadata registry.

The result of the above request would be:

```
--am9x3PR5C0iBEcTCklvMXwS_R580_BTAO
Content-Disposition: form-data; name="header"
Content-Length: 2497
Content-Type: application/ld+json
```

```
{
  "@context" : {
    "ids" : "https://w3id.org/idsa/core/",
    "idsc" : "https://w3id.org/idsa/code/"
  },
  "@type" : "ids:ResultMessage",
  "@id" : "https://w3id.org/idsa/autogen/resultMessage/43d16868-6488-4442-89a7-f351cd7c61fc",
  "ids:recipientConnector" : [ ],
  "ids:senderAgent" : {
    "@id" : "https://www.iais.fraunhofer.de"
  },
  "ids:recipientAgent" : [ ],
  "ids:securityToken" : {
    "@type" : "ids:DynamicAttributeToken",
    "@id" : "https://w3id.org/idsa/autogen/dynamicAttributeToken/a90fa21b-1be8-499c-8e01-f6de2e81f06b",
    "ids:tokenValue" :
"eyJ0eXAiOiJhdCtqd3QiLCJraWQiOiJUQ1VGZUNOYXphbEtIZzlLenJ6TElBelJXVE1E
RFdTYTdMY005WndITXlvIiwiYWxnIjoiUlMyNTYifQ.eyJhdWQiOiJpZHNjOklEU19D
T05ORUNUT1JTX0FMTCIsImlzcyI6Imh0dHBzOi8vZGFwcy5haXNlYy5mcmF1bmhvZ
mVyLmRlIiwic3ViIjoiOTI6MTQ6RTc6QUM6MTA6MjI6RjI6Q0M6MDU6NkU6MkE6
MkI6OEQ6REI6MTA6RDY6NEQ6QTA6QTE6NTM6a2V5aWQ6Q0I6OEM6Qzc6QjY6
ODU6Nzk6QTg6MjM6QTY6Q0I6MTU6QUI6MTc6NTA6MkY6RTY6NjU6NDM6NUQ
6RTgiLCJuYmYiOjE2NDc1MTI3MDksImlhdCI6MTY0NzUxMjcwOSwianRpIjoiTVRB
M09ETTVOemM1TmpFMk1UTXdORGt6TVRjPSIsImV4cCI6MTY0NzUxNjMwOSwiY
2xpZW50X2lkIjoiOTI6MTQ6RTc6QUM6MTA6MjI6RjI6Q0M6MDU6NkU6MkE6MkI6
OEQ6REI6MTA6RDY6NEQ6QTA6QTE6NTM6a2V5aWQ6Q0I6OEM6Qzc6QjY6ODU6
Nzk6QTg6MjM6QTY6Q0I6MTU6QUI6MTc6NTA6MkY6RTY6NjU6NDM6NUQ6RTgi
LCJzZWN1cml0eVBvb2ZpbGUiOiJpZHNjOkJBU0VfU0VDVVJJVFlfUFJPRklMRSIsInJ
lZmVycmluZ0Nvbm5lY3RvciI6Imh0dHA6Ly9icm9rZXIuaWRzLmlzc3QuZnJhdW5ob2Z
lci5kZS5kZW1vIiwiQHR5cGUiOiJpZHM6RGF0UGF5bG9hZCIsIkBjb250ZXh0IjoiaHR0
cHM6Ly93M2lkLm9yZy9pZHNhL2NvbnRleHRzL2NvbnRleHQuanNvbmxkIiwidHJhbn
Nwb3J0Q2VydHNTaGEyNTYiOiI5NzRlNjMyNGYxMmYxMDkxNmY0NmJmZGViMT
hiOGRkNmRhNzhjYzZhNmEwNTY2MDMxZmE1ZjFhMzllYzhlNjAwIiwic2NvcGVzIjp
bImlkc2M6SURTX0NPTk5FQ1RPUl9BVFRSSUJVVEVTX0FMTCJdfQ.Lqe2HpN2U1Pl
UgB3VoAqiHGxvJK_qPzYBjuS83mQ-Vf7GlYwy4QwZdy-JrDb3h-PA0-
EbW0g8S9syxnO_2i-O6haS-i0yaRpRZSzLhI-gel-
5tCHqVDl8PaiQuYS_ufizxzMGRfGgI5DQS7_Od6cAGG8zJUOsHgJR07Jq1L3yH_zQnU
-edxAruZ6Mhgp-i_okm-
```

laQdbqfVeHLDjDSHlMsNz4KgY4grbXs_t0MFYnA1YBuUGA66XiC5c6mmYOQzj44y_PuO4BR2XTXMcuckvuCbxZbjO7iEWBkuMACXr3qVx4AUSKDiyLyYPzF9KOrG7aUgtBEVKt4SFjbvuelBLGw",

"ids:tokenFormat" : {

"@id" : "https://w3id.org/idsa/code/JWT"

}

},

"ids:correlationMessage" : {

"@id" : "https://w3id.org/idsa/autogen/queryMessage/49040fd5-ea7e-4e01-be4b-bf0b3babdc03"

},

"ids:modelVersion" : "4.0.3",

"ids:issued" : {

"@value" : "2022-03-17T11:00:35.545Z",

"@type" : "http://www.w3.org/2001/XMLSchema#dateTimeStamp"

},

"ids:issuerConnector" : {

"@id" : "https://localhost/"

}

}

--am9x3PR5C0iBEcTCklvMXwS_R580_BTAO

Content-Disposition: form-data; name="payload"

Content-Length: 571

?subject ?predicate ?object

<https://localhost/connectors/-832458571/1188459774/-605367243/-413319368/1939874764> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://w3id.org/idsa/core/Artifact>

<https://localhost/connectors/-832458571/1188459774/-605367243/-413319368/1939874764> <http://www.w3.org/2002/07/owl#sameAs> **<http://w3id.org/engrd/connector/artifact/1>**

<https://localhost/connectors/-832458571/1188459774/-605367243/-413319368/1939874764> <https://w3id.org/idsa/core/creationDate> "2022-03-17T10:23:51.824Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>

--am9x3PR5C0iBEcTCklvMXwS_R580_BTAO--

As can see the provider connector offers `http://w3id.org/engrd/connector/artifact/1` as an artifact.

#### 3.3.2.3 Request an Artifact

Now that consumer finds producer in metadata registry, it can check the available resources (for example `http://w3id.org/engrd/connector/artifact/1`) and request it via the following snippet:

```
curl --location --request POST 'https://localhost:8084/proxy' \
--header 'Content-Type: text/plain' \
--data-raw '{
   "multipart": "mixed",
   "Forward-To": "https://ecc-provider:8889/data",
   "messageType": "ArtifactRequestMessage" ,
   "requestedArtifact": "http://w3id.org/engrd/connector/artifact/1" ,
   "payload" : {
                  "catalog.offers.0.resourceEndpoints.path":"/pet2"
                  }
}' -k
```

The result of the above request would be:

```
--lorkGOd_45QOjnoKEoWrHXbwLc6iSDJXN4
Content-Disposition: form-data; name="header"
Content-Length: 1148
Content-Type: application/ld+json

{
  "@context" : {
    "ids" : "https://w3id.org/idsa/core/",
    "idsc" : "https://w3id.org/idsa/code/"
  },
  "@type" : "ids:ArtifactResponseMessage",
  "@id" : "https://w3id.org/idsa/autogen/artifactResponseMessage/d6f2603e-dfd0-491c-b76c-dcea65586eb6",
  "ids:correlationMessage" : {
    "@id" : "https://w3id.org/idsa/autogen/artifactRequestMessage/a6d3a52a-c44f-4dd6-b515-dd728a246fdb"
  },
  "ids:issuerConnector" : {
    "@id" : "https://w3id.org/engrd/connector/provider"
  },
```

```
  "ids:recipientConnector" : [ {
    "@id" : "http://w3id.org/engrd/connector"
  } ],
  "ids:senderAgent" : {
    "@id" : "https://w3id.org/engrd/connector/provider"
  },
  "ids:recipientAgent" : [ ],
  "ids:securityToken" : {
    "@type" : "ids:DynamicAttributeToken",
    "@id" : "https://w3id.org/idsa/autogen/dynamicAttributeToken/9c4bb106-4c79-45a4-
a6f4-d2dadaad752c",
    "ids:tokenFormat" : {
      "@id" : "https://w3id.org/idsa/code/JWT"
    },
    "ids:tokenValue" : "DummyTokenValue"
  },
  "ids:modelVersion" : "4.1.0",
  "ids:issued" : {
    "@value" : "2022-03-14T22:16:15.210Z",
    "@type" : "http://www.w3.org/2001/XMLSchema#dateTimeStamp"
  }
}
--lorkGOd_45QOjnoKEoWrHXbwLc6iSDJXN4
Content-Disposition: form-data; name="payload"
Content-Length: 160

{"firstName":"John","lastName":"Doe","address":"591  Franklin Street,
Pennsylvania","checksum":"ABC123 2022/03/14
23:16:15","dateOfBirth":"2022/03/14 23:16:15"}
```


#### 3.3.2.4 Vocabulary Provider Test

To test the vocabulary provider, one needs first to clone the raspatory from https://github.com/PLATOONProject/PLATOON_IDS-Vocabulary-Provider and running the containers via:

```
$ git clone https://github.com/PLATOONProject/PLATOON_IDS-Vocabulary-Provider
$ cd PLATOON_IDS-Vocabulary-Provider
```

```
$ docker-compose up -d
```

After making the Vocabulary Provider up and running, we need to run deploy the IDS connector as explained in section 3.3.2 with this difference that after cloning the TRUE Connector, the use needs to change a configuration which is in `.env` as follows:

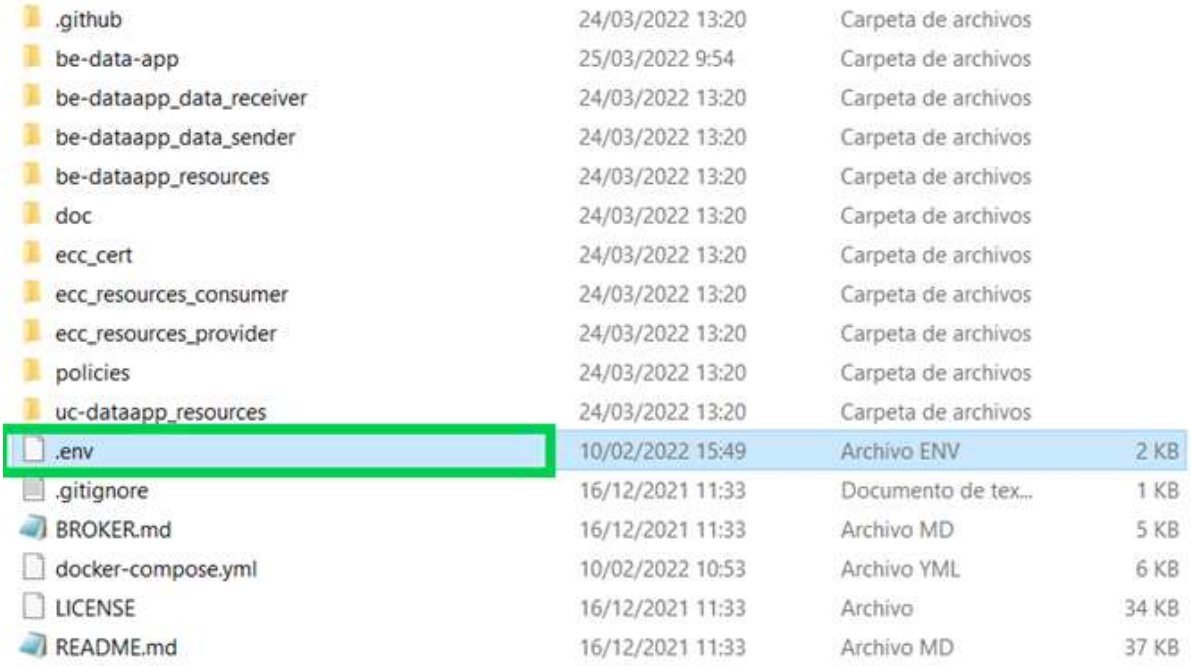


**Figure 27: The .env fine which contain TRUE Connector configurations**

```
# REST Communication type between ECC - mixed | form | http-header
MULTIPART_ECC=mixed
```

Should be replaced with:

```
# REST Communication type between ECC - mixed | form | http-header
MULTIPART_ECC=form
```

The rest of the deployment will be the same as section 3.3.2.

For interacting with Vocabulary Provider, the IDS connector can send different messages such as `DescriptionResponseMessage. DescriptionResponseMessage` allows a connector to call the PLATOON IDS Vocabulary Provider and obtain generic information from it; specifically, a "config.json" file is returned. To do so the user can run the following request:

```
curl --location --request POST 'https://localhost:8084/proxy' \
--header 'Content-Type: text/plain' \
--data-raw '{
"multipart": "form",
"Forward-To": "https://VOCABULARY_PROVIDER_IP:8080/api/ids/data",
"messageType": "DescriptionRequestMessage"
}'
```

VOCABULARY_PROVIDER_IP is the IP of the deployed Vocabulary Provider. The result of the above request will be:


```
--vCX149HzMbdxjrP_4z4BojtmQkC5N_M3NxY
Content-Disposition: form-data; name="header"
Content-Length: 1132
Content-Type: application/ld+json


{
  "@context" : {
    "ids" : "https://w3id.org/idsa/core/",
    "idsc" : "https://w3id.org/idsa/code/"
  },
  "@type" : "ids:DescriptionResponseMessage",
  "@id" : "https://w3id.org/idsa/autogen/descriptionResponseMessage/bbb6248f-1c46-44ed-b48e-c3b2f478b5d2",
  "ids:senderAgent" : {
    "@id" : "https://w3id.org/idsa/autogen/baseConnector/7b934432-a85e-41c5-9f65-669219dde4ea"
  },
  "ids:securityToken" : {
    "@type" : "ids:DynamicAttributeToken",
    "@id" : "https://w3id.org/idsa/autogen/dynamicAttributeToken/3a4e0053-41c8-4e7c-b62b-2ecb0d448a54",
    "ids:tokenValue" : "INVALID_TOKEN",
    "ids:tokenFormat" : {
      "@id" : "https://w3id.org/idsa/code/JWT"
    }
  },
  "ids:issuerConnector" : {
    "@id" : "http://w3id.org/engrd/connector"
  },
  "ids:modelVersion" : "4.0.0",
  "ids:issued" : {
    "@value" : "2022-03-25T15:55:52.320Z",
    "@type" : "http://www.w3.org/2001/XMLSchema#dateTimeStamp"
  },
  "ids:correlationMessage" : {
    "@id" : "https://w3id.org/idsa/autogen/descriptionRequestMessage/571c5972-544f-4f90-8e07-623f1217f0ca"
  },
  "ids:recipientConnector" : [ ],
  "ids:recipientAgent" : [ ]
}
--vCX149HzMbdxjrP_4z4BojtmQkC5N_M3NxY
Content-Disposition: form-data; name="payload"
Content-Length: 1162
```

```
{
  "@context" : {
    "ids" : "https://w3id.org/idsa/core/",
    "idsc" : "https://w3id.org/idsa/code/"
  },
  "@type" : "ids:BaseConnector",
  "@id" : "https://w3id.org/idsa/autogen/baseConnector/7b934432-a85e-41c5-9f65-
669219dde4ea",
  "ids:version" : "0.0.1-SNAPSHOT",
  "ids:description" : [ {
    "@value" : "IDS Connector for hosting the messages for the Vocabulary Provider",
    "@type" : "http://www.w3.org/2001/XMLSchema#string"
  } ],
  "ids:hasDefaultEndpoint" : {
    "@type" : "ids:ConnectorEndpoint",
    "@id" : "https://w3id.org/idsa/autogen/connectorEndpoint/711719ed-05fe-40c6-9137-
62c7599d2367",
    "ids:accessURL" : {
      "@id" : "https://vocabulary.daekin.tecnalia.com:8080/api/ids/data"
    }
  },
  "ids:securityProfile" : {
    "@id" : "https://w3id.org/idsa/code/BASE_SECURITY_PROFILE"
  },
  "ids:maintainer" : {
    "@id" : "https://www.tecnalia.com/"
  },
  "ids:curator" : {
    "@id" : "https://www.tecnalia.com/"
  },
  "ids:inboundModelVersion" : [ "4.0.0" ],
  "ids:outboundModelVersion" : "4.0.0",
  "ids:title" : [ {
    "@value" : "IDS Vocabulary Provider",
    "@type" : "http://www.w3.org/2001/XMLSchema#string"
  } ]
}
--vCX149HzMbdxjrP_4z4BojtmQkC5N_M3NxY--
```


As can be seen Vocabulary Provider returns `DescriptionResponseMessage` message. `DescriptionResponseMessage` is the corresponding reply message to the description request message that contains in the payload the "config.json" file.

### 3.3.3 Criteria and measures

For measurement, the following table summarizes the main characteristics of the components.

**Table 6: Main characteristics of the components**

| | Dockerized | Documentation | Open Source | Available on Platoon GitHub | Fully Configurable |
|---|---|---|---|---|---|
| TRUE Connector | Yes | Yes | Yes | No | Yes |
| Metadata Registry | Yes | Yes | Yes | Yes | Yes |
| Vocabulary Provider | Yes | Yes | Yes | Yes | Yes |

### 3.3.4 Results

As we showed, one producer connector could register itself in metadata registry and another consumer connector could find it and request a JSON content. Finally, the producer connector forwarded the JSON content to the consumer connector.

Moreover, regarding the Vocabulary Provider we showed that an IDS connector can send a request to Vocabulary Provider and get back the result to have a successful machine-to-machine communication. For this purpose, we used DescriptionRequestMessage. However, as explained in WP3 Vocabulary Provider offers more IDS Messages for retrieving data.

## 4 Conclusions

This deliverable focused on testing the components of the PLATOON reference architecture and the documentation of the test results. Using the tests, we evaluated how the components operated together.

For the tests, we conceived three different test and documented the results. As the results show, the tests ran successfully and therefore we conclude that the PLATOON architecture is working as expected. The deliverable provides the necessary step-by-step instructions to replicate our tests and compare them with the expected results.

During our testing, we communicated the discovered issues to the responsible partners and worked together on solving the respective issue. Thanks to the performed test, we were able to iteratively improve the quality of the different components before the implementation and validation in large scale pilots in WP6.

In this deliverable, the tests focused on integration tests using representative, anonymised datasets on fictitious scenarios to guarantee a broad coverage of different use-cases. Scenarios based on real-world use-cases are covered by the different pilots as part of WP6.

# 5 Internal Review

Mark with X the corresponding column:

| | | |
|---|---|---|
| **Y= yes** | **N= no** | **NA = not applicable** |

Name of reviewer: Erik Maqueda

Organisation: TECN

Date: 23/03/2022

| ELEMENT TO REVIEW | Y | N | NA | Comments | Author |
|---|---|---|---|---|---|
| **FORMAT**: Does the document …? | | | | | |
| …include editors, deliverable name, version number, dissemination level, date, and status? | x | | | | |
| … contain a license (in case of public deliverables)? | | | x | | |
| … include the names of contributors and reviewers? | x | | | | |
| … contain a version table? | x | | | | |
| … contain an updated table of contents? | x | | | | |
| … contain a list of figures? | x | | | | |
| … contain a list of tables? | x | | | | |
| … contain a list of terms and abbreviations? | x | | | | |
| … contain an Executive Summary? | x | | | | |
| … contain a Conclusions section? | x | | | | |
| … contain a List of References (Bibliography) in the appropriate format? | x | | | | |
| … use the fonts and sections defined in the official template? | x | | | | |
| … use correct spelling and grammar? | x | | | | |
| … conform to length guidelines (50 pages maximum (plus Executive Summary and annexes) | x | | | | |
| … conform to guidelines regarding Annexes (inclusion of complementary information) | x | | | | |

| | | | | | |
|---|---|---|---|---|---|
| … present consistency along the whole document in terms of English quality/style? (to avoid accidental usage of copy & paste text) | x | | | | |
| **About the content…** | | | | | |
| Is the deliverable content correctly written? | x | | | | |
| Is the overall style of the deliverable correctly organized and presented in a logical order? | x | | | | |
| Is the Executive Summary self-contained, following the guidelines and does it include the main conclusions of the document? | x | | | | |
| Is the body of the deliverable (technique, methodology results, discussion) well enough explained? | x | | | | |
| Are the contents of the document treated with the required depth? | x | | | | |
| Does the document need additional sections to be considered complete? | | x | | | |
| Are there any sections in the document that should be removed? | | x | | | |
| Are all references in the document included in the references section? | x | | | | |
| Have you noticed any text in the document not well referenced? (copy and paste of text/picture without including the reference in the reference list) | | x | | | |
| **TECHNICAL RESEARCH WPs (WP2-WP5)** | | | | | |
| Is the deliverable sufficiently innovative? | x | | | | |
| Does the document present technical soundness and its methods are correctly explained? | X | | | | |
| What do you think is the strongest aspect of the deliverable? | | | | Section 3 | |
| What do you think is the weakest aspect of the deliverable? | | | | Conclussions | |
| Please perform a brief evaluation and/or validation of the results, if applicable. | x | | | | |
| **VALIDATION WP (WP6)** | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Does the document present technical soundness and the validation methods are correctly explained? | | | X | | |
| What do you think is the strongest aspect of the deliverable? | | | X | | |
| What do you think is the weakest aspect of the deliverable? | | | X | | |
| Please perform a brief evaluation and/or validation of the results, if applicable. | | | X | | |
| **DISSEMINATION AND EXPLOITATION WPs (WP8 & WP9)** | | | | | |
| Does the document present a consistent outreach and exploitation strategy? | | | X | | |
| Are the methods and means correctly explained? | | | X | | |
| What do you think is the strongest aspect of the deliverable? | | | X | | |
| What do you think is the weakest aspect of the deliverable? | | | X | | |
| Please perform a brief evaluation and/or validation of the results, if applicable. | | | x | | |

**SUGGESTED IMPROVEMENTS**

| PAGE | SECTION | SUGGESTED IMPROVEMENT |
|---|---|---|
| | | see comments in the document |

**CONCLUSION**

Mark with X the corresponding line.

| | |
|---|---|
| | Document accepted; no changes required. |
| | Document accepted; changes required. |
| x | Document not accepted; it must be reviewed after changes are implemented. |

Please rank this document globally on a scale of 1-5.

*(1-Poor; 2–Fair; 3–Average; 4–Good; 5–Excellent)*

Using a half point scale.

Mark with X the corresponding grade.

| Document grade | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 | 5 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | x | | | | | |

Name of reviewer: Juan Prieto

Organisation: Minsait (Indra solutions TI)

Date:24/03/2022

| ELEMENT TO REVIEW | Y | N | NA | Comments | Author |
|---|---|---|---|---|---|
| **FORMAT**: Does the document …? | | | | | |
| …include editors, deliverable name, version number, dissemination level, date, and status? | | | | | |
| … contain a license (in case of public deliverables)? | | | | | |
| … include the names of contributors and reviewers? | | | | | |
| … contain a version table? | | | | | |
| … contain an updated table of contents? | | | | | |
| … contain a list of figures? | | | | | |
| … contain a list of tables? | | | | | |
| … contain a list of terms and abbreviations? | | | | | |
| … contain an Executive Summary? | | | | | |
| … contain a Conclusions section? | | | | | |
| … contain a List of References (Bibliography) in the appropriate format? | | | | | |
| … use the fonts and sections defined in the official template? | | | | | |
| … use correct spelling and grammar? | | | | | |
| … conform to length guidelines (50 pages maximum (plus Executive Summary and annexes) | | | | | |
| … conform to guidelines regarding Annexes (inclusion of complementary information) | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| … present consistency along the whole document in terms of English quality/style? (to avoid accidental usage of copy & paste text) | | | | | |
| **About the content…** | | | | | |
| Is the deliverable content correctly written? | | | | | |
| Is the overall style of the deliverable correctly organized and presented in a logical order? | | | | | |
| Is the Executive Summary self-contained, following the guidelines and does it include the main conclusions of the document? | | | | | |
| Is the body of the deliverable (technique, methodology results, discussion) well enough explained? | | | | | |
| Are the contents of the document treated with the required depth? | | | | | |
| Does the document need additional sections to be considered complete? | | | | | |
| Are there any sections in the document that should be removed? | | | | | |
| Are all references in the document included in the references section? | | | | | |
| Have you noticed any text in the document not well referenced? (copy and paste of text/picture without including the reference in the reference list) | | | | | |
| **TECHNICAL RESEARCH WPs (WP2-WP5)** | | | | | |
| Is the deliverable sufficiently innovative? | | | | | |
| Does the document present technical soundness and its methods are correctly explained? | | | | | |
| What do you think is the strongest aspect of the deliverable? | | | | | |
| What do you think is the weakest aspect of the deliverable? | | | | | |
| Please perform a brief evaluation and/or validation of the results, if applicable. | | | | | |
| **VALIDATION WP (WP6)** | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Does the document present technical soundness and the validation methods are correctly explained? | | | | | |
| What do you think is the strongest aspect of the deliverable? | | | | | |
| What do you think is the weakest aspect of the deliverable? | | | | | |
| Please perform a brief evaluation and/or validation of the results, if applicable. | | | | | |
| **DISSEMINATION AND EXPLOITATION WPs (WP8 & WP9)** | | | | | |
| Does the document present a consistent outreach and exploitation strategy? | | | | | |
| Are the methods and means correctly explained? | | | | | |
| What do you think is the strongest aspect of the deliverable? | | | | | |
| What do you think is the weakest aspect of the deliverable? | | | | | |
| Please perform a brief evaluation and/or validation of the results, if applicable. | | | | | |

**SUGGESTED IMPROVEMENTS**

| PAGE | SECTION | SUGGESTED IMPROVEMENT |
|---|---|---|
| | | |

**CONCLUSION**

Mark with X the corresponding line.

| | |
|---|---|
| | Document accepted; no changes required. |
| | Document accepted; changes required. |
| | Document not accepted; it must be reviewed after changes are implemented. |

Please rank this document globally on a scale of 1-5.

*(1-Poor; 2–Fair; 3–Average; 4–Good; 5–Excellent)*

Using a half point scale.

Mark with X the corresponding grade.

| Document grade | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 | 5 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

# 6 References


[1] C. F. Draschner, J. Lehmann and H. Jabeen, “DistSim - Scalable Distributed in-Memory Semantic Similarity Estimation for RDF Knowledge Graphs,” in *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, 2021.

[2] F. Bakhshandegan Moghaddam, C. Draschner, J. Lehmann and H. Jabeen, “Literal2Feature: An Automatic Scalable RDF Graph Feature Extractor,” in *Proceedings of the 17th International Conference on Semantic Systems, SEMANTICS 2021*, 2021.

[3] C. F. Draschner, C. Stadler, F. Bakhshandegan Moghaddam, J. Lehmann and H. Jabeen, “DistRDF2ML - Scalable Distributed In-Memory Machine Learning Pipelines for RDF Knowledge Graphs,” in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.

[4] F. Bakhshandegan Moghaddam, J. Lehmann and H. Jabeen, “DistAD: A Distributed Generic Anomaly Detection Framework over Large KGs,” 2022. [Online]. Available: https://hajirajabeen.github.io/publications/ICSC_2022_DistAnomalies_CR.pdf. [Accessed 18 March 2022].

[5] F. Bakhshandegan Moghaddam, C. F. Draschner, J. Lehmann and H. Jabeen, “Semantic Web Analysis with Flavor of Micro-Services,” in *LAMBDA Big Data Analytics Summer School. Doctoral Workshop*, 2021 forthcoming.

[6] HashiCorp, Inc., “HashiCorp Vagrant. Development Environments Made Easy.,” 2021. [Online]. Available: https://www.hashicorp.com/. [Accessed 27 February 2022].

[7] Oracle Corporation, “Welcome to VirtualBox.org!,” 2022. [Online]. Available: https://www.virtualbox.org/. [Accessed 27 February 2022].

[8] Grafana Labs, “Grafana,” 2022. [Online]. Available: https://grafana.com. [Accessed 27 February 2022].

[9] Scientific Data Management Group at TIB, “SDM-TIB/SDM-RDFizer,” 2022. [Online]. Available: https://github.com/SDM-TIB/SDM-RDFizer. [Accessed 27 February 2022].

[10] Scientific Data Management Group at TIB, “SDM-TIB/DeTrusty,” 2022. [Online]. Available: https://github.com/SDM-TIB/DeTrusty. [Accessed 27 February 2022].

## Appendix: Docker-compose.yml file for Non-Edge Scenario

```
version: '3.8'


services:
api:
image: fmoghaddam/tecndashboard:v1
ports:
- "8000:8000"
container_name: platoon-pynalia-api
hostname: platoon-pynalia-api
networks:
- spark-net


front:
image: fmoghaddam/tecndashboardui:v1
ports:
- "4200:4200"
container_name: platoon-pynalia-front
hostname: platoon-pynalia-front
networks:
- spark-net


spark-livy:
image: fmoghaddam/sansaspark:v2
environment:
- SPARK_MODE=master
- SPARK_DEPLOY_MODE=cluter
- SPARK_RPC_AUTHENTICATION_ENABLED=no
- SPARK_RPC_ENCRYPTION_ENABLED=no
- SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
- SPARK_SSL_ENABLED=no
```

```
volumes:
- ./jars/sansa.jar:/opt/bitnami/spark/jars/sansa.jar
ports:
- 8080:8080
networks:
- spark-net
deploy:
restart_policy:
condition: any


spark-worker-livy:
hostname: spark-worker-livy
image: fmoghaddam/sansaspark:v2
environment:
- SPARK_MODE=worker
- SPARK_MASTER_URL=spark://spark-livy:7077
- SPARK_WORKER_MEMORY=10G
- SPARK_WORKER_CORES=6
- SPARK_RPC_AUTHENTICATION_ENABLED=no
- SPARK_RPC_ENCRYPTION_ENABLED=no
- SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
- SPARK_SSL_ENABLED=no
volumes:
- ./jars/sansa.jar:/opt/bitnami/spark/jars/sansa.jar
networks:
- spark-net
deploy:
restart_policy:
condition: any


livy:
hostname: livy
```

```
image: fmoghaddam/sansalivy:v3
ports:
- 8998:8998
- 4040:4040
volumes:
- ./jars/sansa.jar:/opt/spark-3.0.2-bin-hadoop2.7/jars/sansa.jar
environment:
- SPARK_MASTER_ENDPOINT=spark-livy
- SPARK_MASTER_PORT=7077
depends_on:
- spark-livy
- namenode
- datanode
- hue
- spark-worker-livy
networks:
- spark-net
deploy:
mode: global
restart_policy:
condition: any
hue:
hostname: hue
image: fmoghaddam/sansahue:v1
ports:
- 8088:8088
environment:
- NAMENODE_HOST=namenode
- SPARK_MASTER=spark://spark-livy:7077
depends_on:
- namenode
networks:
- spark-net
deploy:
```

```
restart_policy:
condition: any


namenode:
image: fmoghaddam/sansanamenode:v1
hostname: namenode
ports:
- 8020:8020
environment:
- CLUSTER_NAME=test
- CORE_CONF_fs_defaultFS=hdfs://namenode:8020
- CORE_CONF_hadoop_http_staticuser_user=root
- CORE_CONF_hadoop_proxyuser_hue_hosts=*
- CORE_CONF_hadoop_proxyuser_hue_groups=*
- HDFS_CONF_dfs_webhdfs_enabled=true
- HDFS_CONF_dfs_permissions_enabled=false
-
HDFS_CONF_dfs_namenode_datanode_registration_ip___hostname___check=fal
se
healthcheck:
interval: 5s
retries: 100
start_period: 10s
volumes:
- ./data/namenode:/hadoop/dfs/name
networks:
- spark-net
deploy:
restart_policy:
condition: any


datanode:
image: fmoghaddam/sansadatanode:v1
```

```
hostname: datanode
volumes:
- ./data/datanode:/hadoop/dfs/data
environment:
- CORE_CONF_fs_defaultFS=hdfs://namenode:8020
depends_on:
- namenode
healthcheck:
interval: 5s
retries: 100
start_period: 10s
networks:
- spark-net
deploy:
mode: global
restart_policy:
condition: any


apiwrapper:
image: fmoghaddam/sansarest:v3
environment:
- LIVY_URL_PORT=http://livy:8998
- SANSA_JAR_LOCATION=hdfs://namenode:8020/data/sansa.jar
- NAMENODE=namenode
- DATANODE=datanode
- NAMENODE_HOST=hdfs://namenode:8020
- SPARK_WORKER_MEMORY=10g
- SPARK_EXECUTOR_CORES=6
ports:
- 8085:8085
depends_on:
- livy
networks:
```

```
- spark-net
deploy:
restart_policy:
condition: any


sdmrdfizer:
image: asakor/sdmrdfizier:4.0.1
container_name: sdmrdfizer
hostname: sdmrdfizer
volumes:
- .:/data
networks:
- spark-net
depends_on:
- pilot2akg1
- pilot2akg2
environment:
- SPARQL_ENDPOINT_IP_1=pilot2akg1
- SPARQL_ENDPOINT_PORT_1=1111
- SPARQL_ENDPOINT_IP_2=pilot2akg2
- SPARQL_ENDPOINT_PORT_2=1111
- SPARQL_ENDPOINT_USER=dba
- SPARQL_ENDPOINT_PASSWD=dba
- SPARQL_ENDPOINT_GRAPH=http://platoon.eu/Pilot2A/KG1
- RDF_DUMP_FOLDER_PATH=/data
deploy:
restart_policy:
condition: any


detrusty:
image: prohde/detrusty:0.2.0
container_name: detrusty
```

```
hostname: detrusty
volumes:
- ./DeTrusty/Config/:/DeTrusty/Config/
ports:
- 5001:5000
networks:
- spark-net
depends_on:
- pilot2akg1
- pilot2akg2
deploy:
restart_policy:
condition: any


pilot2akg1:
image: kemele/virtuoso:6-stable
hostname: pilot2akg1
volumes:
- ./rdf-dump1:/data
ports:
- 8090:8090
- 1111:1111
networks:
- spark-net
deploy:
restart_policy:
condition: any


pilot2akg2:
image: kemele/virtuoso:6-stable
hostname: pilot2akg2
volumes:
```

```
- ./rdf-dump2:/data
ports:
- 8091:8090
- 1112:1111
networks:
- spark-net
deploy:
restart_policy:
condition: any


networks:
spark-net:
external: true
```