



Policy Cloud  
Cloud for Data-Driven Policy Management

## CLOUD FOR DATA-DRIVEN POLICY MANAGEMENT

Project Number: 870675

Start Date of Project: 01/01/2020

Duration: 36 months

## D6.8 INTEGRATION OF RESULTS: POLICYCLOUD COMPLETE ENVIRONMENT M24

Dissemination Level	PU
Due Date of Deliverable	31/12/2021 (M24)
Actual Submission Date	03/01/2022
Work Package	WP6, Use Case Adaptation, Integration & Experimentation
Task	T6.1 Integration
Type	Demonstrator
Approval Status	
Version	V2.0
Number of Pages	p.1 – p.29

**Abstract:** This document contains complementary material for the PolicyCLOUD installation environment. It includes detailed instructions regarding installation and integration of the components as well as an analysis of the cloud environment that the platform is hosted.

The information in this document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided "as is" without guarantee or warranty of any kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/ her sole risk and liability. This deliverable is licensed under a Creative Commons Attribution 4.0 International License.



PolicyCloud has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 870675.

## Versioning and Contribution History

Version	Date	Reason	Author
1.1	23/11/2021	Update ToC	ICCS
1.2	03/12/2021	Integrate updated architecture design	ICCS
1.3	10/12/2021	Update Marketplace section	UPRC
1.4	11/12/2021	Update to sections 2.2.3 and 2.2.4 and had an overall view to the document	IBM
1.5	12/12/2021	Update section 2.4 and 2.7 and had an overview to the document	ATOS
1.6	12/12/2021	Update section 4	EGI
1.7	13/12/2021	First refinement	ICCS
1.8	15/12/2021	Minor changes to the first section	UPRC, IBM, ATOS
1.9	17/12/2021	First Review	IBM
1.10	21/12/2021	Second Review	LXS
1.11	23/12/2021	Address comments	ICCS
1.12	23/12/2021	Quality check	UPRC
1.13	24/12/2021	Version to Submit	ICCS
2.0	03/01/2022	Submitted version	ATOS

## Author List

Organisation	Name
ICCS/NTUA	Kostas Moutselos, Panayotis Michael, Vrettos Moulos, Christos Pavlatos, Marios Koniaris, Panagiotis Tsanakas, Aggelos Kolaitis
LXS	Jose María Zaragoza
IBM	Yosef Moatti
ATOS	María Ángeles Sanguino, Ana Luiza Pontual, Miquel Milá, Ricard Munné
EGI	Giuseppe La Rocca
UPRC	George Manias, Argyro Mavrogiorgou, Athanasios Kiourtis, Ilias Maglogiannis, Nikitas Sgouros, Vasilis Koukos
ITA	Rafael del Hoyo
UBI	Giannis Ledakis, Konstantinos Theodosiou
MAG	Armend Duzha, Nikos Achilleopoulos
SOF	Petya Nikolova, Iskra Yovkova
ITA	Vega Rodríguez

## Abbreviations and Acronyms

Abbreviation/Acronym	Definition
ABAC	Attribute-Based Access Control
DAA	Data Acquisition and Analytics
DORA	DevOps Research and Assessment
EBPM	Evidence Based Policy Making
EC	European Commission
ELSA	Entity-Level Sentiment Analysis
EOSC	European Open Science Cloud
IaC	Infrastructure as Code
IM	Incentives Management
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
NER	Named Entity Recognition
NLP	Natural Language Processing
PDT	Policy Development Toolkit
PME	Policy Model Editor
SLA	Service Level Agreement
URI	Uniform Resource Identifier

# Contents

Versioning and Contribution History.....	2
Author List.....	2
Abbreviations and Acronyms .....	3
Executive Summary.....	6
1 Introduction .....	7
1.1 Summary of Changes .....	7
2 Components .....	8
2.1 Data Acquisition.....	8
2.1.1 Cloud Gateways & APIs for Efficient Data Utilization .....	8
2.2 Data Analytics.....	11
2.2.1 The Data Acquisition and Analytics (DAA) .....	11
2.2.2 DAA API Gateway .....	11
2.2.3 Situational Knowledge Acquisition & Analysis.....	12
2.2.4 Opinion Mining & Sentiment Analysis Component.....	13
2.2.5 Operational Data Repository.....	13
2.2.6 Data Cleaning.....	15
2.2.7 Enhanced Interoperability .....	16
2.3 Policy Development Toolkit.....	17
2.4 Data Visualization.....	18
2.5 Authentication & Data Governance Model.....	19
2.6 Data Marketplace .....	21
2.7 Incentives Management .....	22
3 Source code management.....	23
4 Cloud Environment.....	26
5 Conclusion.....	28
References.....	29

## List of Figures

Figure 1 - Overall core component architecture (updated).....	7
Figure 2 - Data Acquisition Layer.....	8
Figure 3 - Streaming Component.....	9
Figure 4 - Data Analytics Layer.....	11
Figure 5 – Main repository (Ultrascable Operational Database).....	13
Figure 6 - Data Cleaning.....	15
Figure 7 – Enhanced Interoperability.....	16
Figure 8 - Policy Development Toolkit (PDT).....	17
Figure 9 - Data visualization.....	18
Figure 10 - Data Governance Model.....	19
Figure 11 - Keycloak Login.....	20
Figure 12 - EGI Check-in.....	20
Figure 13 – Data Marketplace.....	21
Figure 14 – Incentives Management.....	22
Figure 15 - Trivy comprehensive scanner.....	23
Figure 16 - Integration Server Analytics.....	25
Figure 17 - Utilization Graph.....	26

## Executive Summary

This deliverable has been released in December 2021, at M24 of the project, and its main objective is to specify the **updated** integration results between the PolicyCLOUD components. So, **this document is an updated version of the previous deliverable (D6.2 Integration of Results: PolicyCLOUD Complete Environment) [7]**.

This deliverable will follow the methodology of D6.2 that was submitted in M12 (December 2020) which has two main pillars:

1. Define common practices for integration and validation of the outcomes of the project
2. Detail the cloud environment the project will make use of to demonstrate the results

Regarding the former, GitLab will be the base code repository for the project, where the project already owns an organizational account. Over GitLab [1], the trunk-based development branching policy has been applied, as we considered it the most suitable policy given the project characteristics. Also, GitLab's issue reporting tool has been adopted, as it is fully integrated with GitLab's features. The test bed to support the demonstrators has been deployed over EGI's (EGI) infrastructure where flexibility is one of the critical features.

This deliverable abstractly incorporates all the changes and implementations that WP2, WP3, WP4 and WP5 had made during the second year of the project. More details about the components and the actual implementation can be found to the relative WP deliverables [8] [9] [10].

# 1 Introduction

In the following section the overall Architecture that was presented in D2.6 Conceptual Model & Reference Architecture M18 (the revised version) [8] is decomposed and analyzed. The main goal is to identify HOW each component can be deployed, WHERE it can be instantiated and WHAT are the API interfaces that are offered. Also, it analyzes the technical constraints for the cloud environment and the minimum requirements necessary for the full deployment.

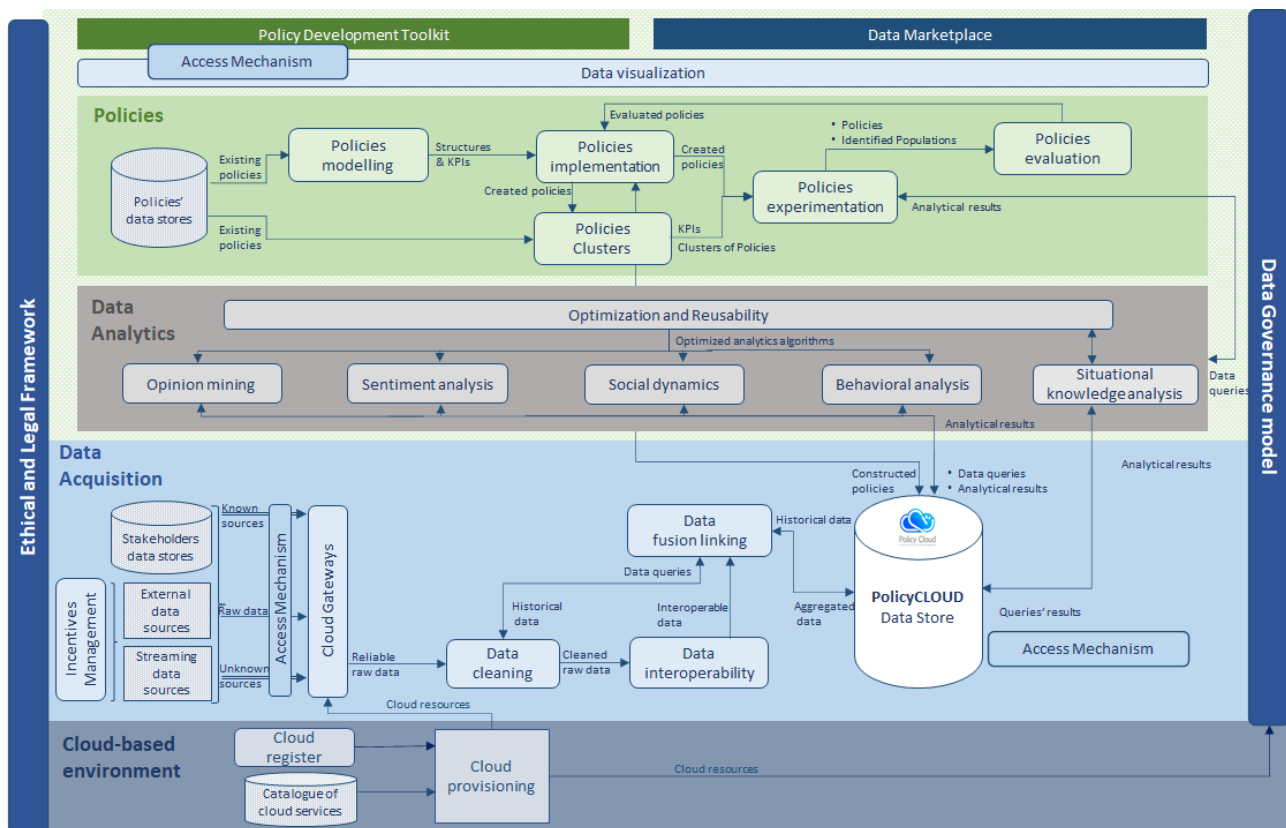


FIGURE 1 - OVERALL CORE COMPONENT ARCHITECTURE (UPDATED)

## 1.1 Summary of Changes

A summary of changes is provided in the following list:

1. Introduction of Data Marketplace (Section 2.6)
2. PDT integration (Section 2.3)
3. Data Acquisition layer component integration (Section 2.1)
4. Registry and CI new features (Section 3)
5. Repository integration with new datasets (Section 2.2.5)
6. Introduction of the Authentication & Data Governance Model platform (Section 2.5)
7. Data Analytics layer component integration (Section 2.2)

## 2 Components

In the following section a description of the component and a reference to the source code is provided.

### 2.1 Data Acquisition

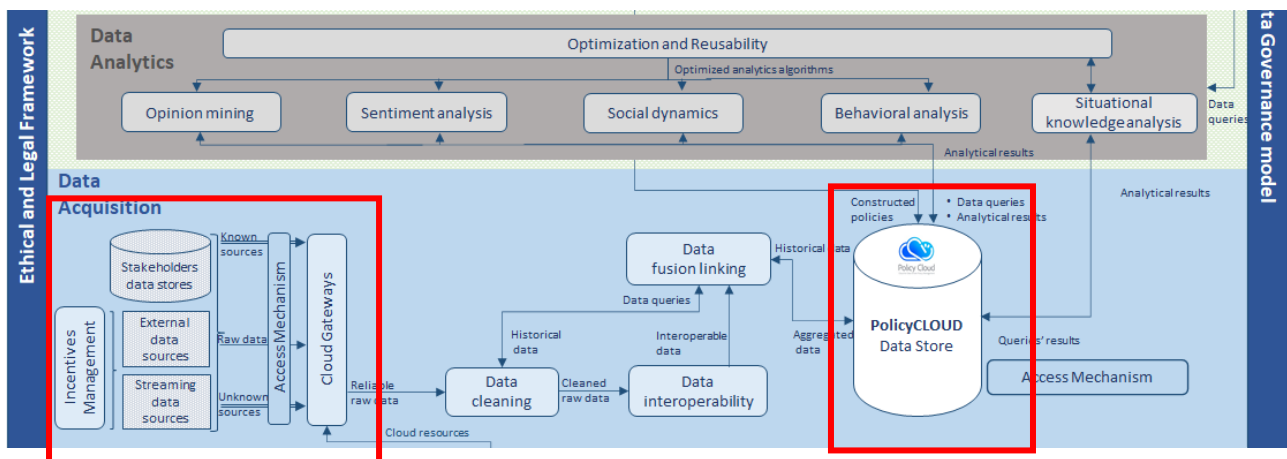


FIGURE 2 - DATA ACQUISITION LAYER

#### 2.1.1 Cloud Gateways & APIs for Efficient Data Utilization

The Cloud Gateway and API component enhances the abilities and services offered by a unified Gateway to move streaming and batch data from data owners into PolicyCLOUD data management layer, which supports access to both SQL and NoSQL data stores and public and private data. On top of this, the main goal of this component is to handle each request by invoking the appropriate microservices and aggregating the results. Hence, it enhances the design of resources and structure, add dynamic routing parameters, and develop custom authorizations logic. PolicyCLOUD's Cloud Gateway and API component are scalable, highly available and fault tolerant, also they share state without compromising performance.

Cloud Gateways & APIs component consists of two main sub-components/microservices. The initial design of these two specific sub-components, that can be used either for fetching data from an external file or from a social media platform like Twitter, includes complete workflows and pipelines for pushing data into the PolicyCLOUD platform. These sub-components have been integrated with each other and an authentication mechanism (both ABAC and keyCloak) controls access to them. In addition during the past year, additional parameters have been added to the registration APIs (both for datasets and analytic functions) to support the legal requirements. For details, please refer the DAA section of D4.3 Reusable Model & Analytical Tools: Design and Open Specification 2 [10].

### 2.1.1.1 TWITTER CONNECTOR COMPONENT

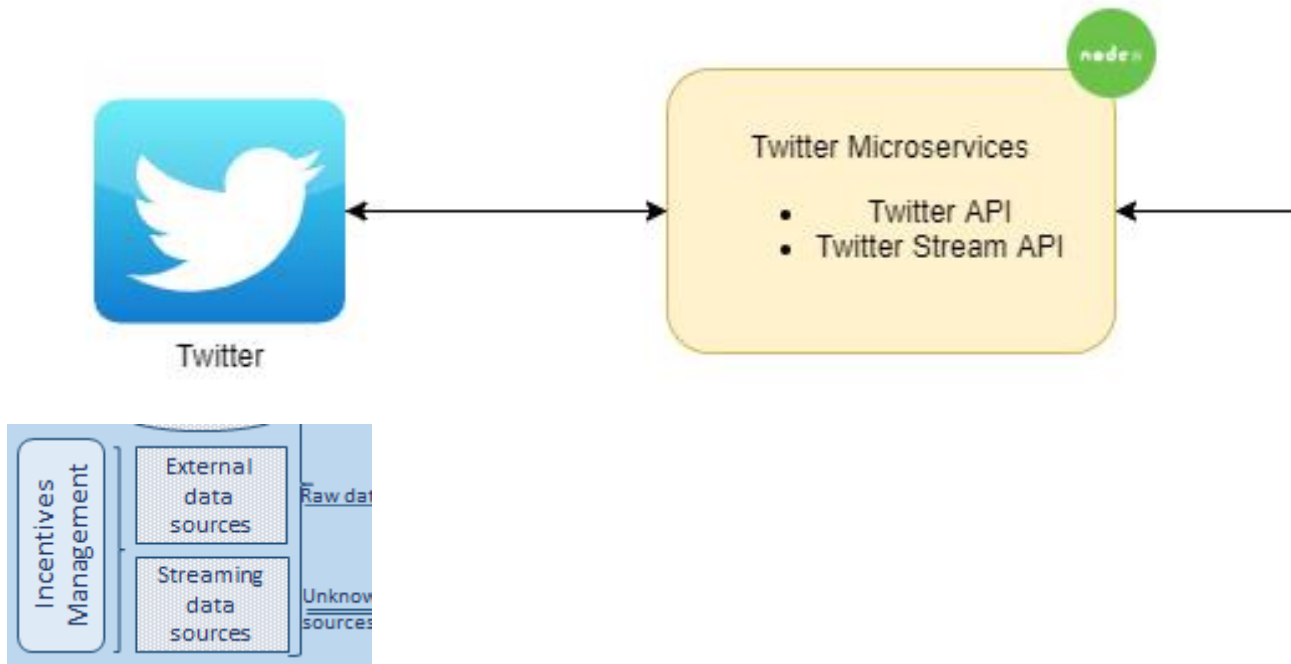


FIGURE 3 - STREAMING COMPONENT

The Twitter Microservice is a component of the Cloud Gateway, providing access to Twitter data to the Gateway's clients without the need to directly connect to Twitter API. It has been implemented in NodeJS utilizing the twitter-v2 npm package<sup>7</sup>. Furthermore, Twitter is launching the API v2, an improved version which promises to provide a better developer experience by giving access to a wide variety of data sources and tools. Twitter ensures the quality of its data by applying spam filters, access to all results of a query and not only to a partition of results, user-friendly and simplified JSON objects, shorter URLs and OpenAPI specification to test endpoints and watch for any change. One of the major limitations of data collection using the Twitter API is that the client machine will always try to keep up with the data stream otherwise the stream just disconnects. Hence, in order to overcome this limitation, this microservice integrates also with Kafka [11] event streaming platform, so raw Twitter data can be processed without worrying about the stream getting disconnected.

#### 2.1.1.2 SOFIA CONNECTOR COMPONENT

The Sofia sub-component, such as Camden and GTD sub-components, provides also a REST application interface following the OpenAPI specification in order to be easier for the end user to discover the capabilities of the component and to provide well-structured documentation for each of the component's services.

In contrary to the two previously introduced sub-components this particular component incorporates four sub-basic functionalities that are being offered and its one providing a corresponding microservice for its specific scenario of the Sofia use case.

- Roads
- Transport
- Waste
- Parking

#### 2.1.1.3 CAMDEN CONNECTOR COMPONENT

##### SOURCE CODE AND INTEGRATION DETAILS

For all the above-mentioned subcomponents the source code and an installation manual are available on the following repository <https://registry.grid.ece.ntua.gr/george/cloudgatewayv2>. The installation instructions and project requirements are firmly described inside the project's README.md file <https://registry.grid.ece.ntua.gr/george/cloudgatewayv2/-/blob/master/README.md>).

## 2.2 Data Analytics

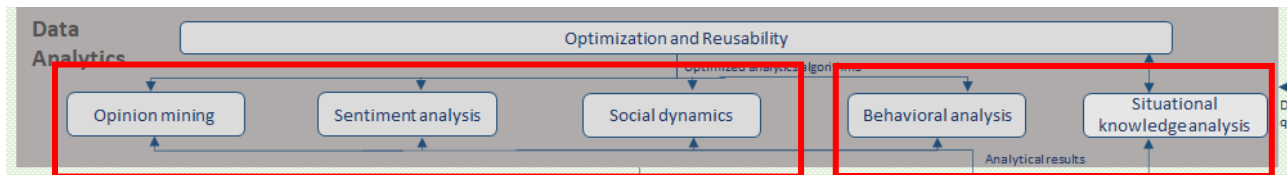


FIGURE 4 - DATA ANALYTICS LAYER

### 2.2.1 The Data Acquisition and Analytics (DAA)

This Layer is the central layer of PolicyCLOUD, between the cloud infrastructure and the Policy layers. This layer provides the functionality for ingesting data coming from various sources while applying filtering and initial analytics, preparing it for deeper analytics on longer term storage (DB, object storage).

### 2.2.2 DAA API Gateway

The DAA API Gateway is responsible for the orchestration and integration of all the components of the DAA layer (analytic functions, data sources, data repository) as well as providing the external interface for the layer's exploiters – analytic owners, data source owners, and the Policy layer (through which the end user apply desired analytics). Its roles include forming the proper setup of the registered analytic functions and data sources to ensure the correct data ingest process (applying the required ingest-time analytics/transformation – in both Ingest-now and Streaming modes) and applying the requested analytics on the requested data source. The DAA API Gateway is implemented as a set of serverless functions (in the OpenWhisk [3] cluster), where the state is managed both on the OpenWhisk itself (e.g. analytic functions' metadata) and the database.

The methods that are exposed are:

- ANALYTIC FUNCTION REGISTRATION
- DATA SOURCE REGISTRATION
- LIST REGISTERED FUNCTIONS
- LIST REGISTERED DATA SOURCES
- APPLY FUNCTION
- GET JOB STATUS

Software prototype code can be found on the PolicyCLOUD's github which is accessible to the members of the consortium and includes code to deploy in OpenWhisk.

#### SOURCE CODE AND INTEGRATION DETAILS

<https://registry.grid.ece.ntua.gr/oshrit/daa>

The setup details for the DAA prototype environment are detailed in "D4.4 Reusable Model & Analytical Tools: Software Prototype 2", section 3.2.1 DAA API Gateway.

### 2.2.3 Situational Knowledge Acquisition & Analysis

In the context of Situational Knowledge Acquisition and Analysis, the SKA-EDA v1 component has been provided for the exploration of datasets based on descriptive analysis conducted by data visualization. This is the second prototype developed in the task and in particular it includes the following three new functionalities:

- Frequency distribution of one or two variables across multiple categories. Representing the number of apparitions of one (or more) value/category corresponding to observable variable (s) in a specific dataset. The JSON results has (horizontal) bar chart as main visualization chart.
- Geographical Distribution. Representing the list of observations that happen in concretes values of a geographical-based variable. The JSON results has “heat map” as main visualization chart.
- Accumulation of one variable across multiple categories. This distribution provides the accumulated value (sum the value of a specific numeric variable) across several categories. The results are portrayed as a bar chart.

This component is in charge of providing exploratory analysis over different datasets, in particular is able to process the following datasets:

- 6 datasets coming from Sofia Municipality's Contact Centre CallSofia (mobile application, 24/7 phone contact centre, and web channel) which belong to the Use Case 3 (Facilitating urban policy making and monitoring through crowdsourcing data analysis),
- “JSA And UC Claimants In Camden” dataset, a *series counting the number of people claiming Jobseeker's Allowance plus those who claim Universal Credit* which belongs to Use case 4 (Predictive analysis towards unemployment risks identification and policy making),
- GTD data base which belongs to Use case 1 (Participatory policies against Radicalization).

This prototype is being integrated in the PolicyCLOUD framework and is in the process of being fully integrated with the rest of the PolicyCLOUD components: the DAA API Gateway developed in the context of WP4, available as an OpenWhisk function, the PolicyCLOUD storage for feeding data and storing the results and the PDT for its invocation.

#### SOURCE CODE AND INTEGRATION DETAILS

Similarly to the Opinion Mining & Sentiment Analysis Component, all the information related to this component is written in D4.4 [12] along with all the details about the baseline technologies and tools used, analytics performed, how to deploy a standalone version of this component, the description of the parameters that are used as input to execute it, and the outputs that will be generated in the execution.

All the source code is available at the project's provisional git: <https://registry.grid.ece.ntua.gr/maria.sanguino/exploratoryanalysis>.

## 2.2.4 Opinion Mining & Sentiment Analysis Component

This task is working on an enhanced version of the sentiment analysis provided in the first prototype. This second version is based on an entity-level sentiment analysis (ELSA) approach<sup>1</sup>. This improved version will support the project use cases by filtering and providing the corresponding sentiments for identified and extracted entities in Tweets. This component is under development and will be deployed via DAA API Gateway developed in WP4 as another analytical service.

All the information related to this component is written in “D4.4 Reusable Model & Analytical Tools: Software Prototype 2” [12] along with all the details about the baseline technologies and tools used.

### SOURCE CODE AND INTEGRATION DETAILS

Until the new version will be released, the installation guide, and all the resources needed of the first version of the sentiment analysis, are available at the project’s provisional git: <https://registry.grid.ece.ntua.gr/jorge.montero/sentimentanalysis>.

## 2.2.5 Operational Data Repository

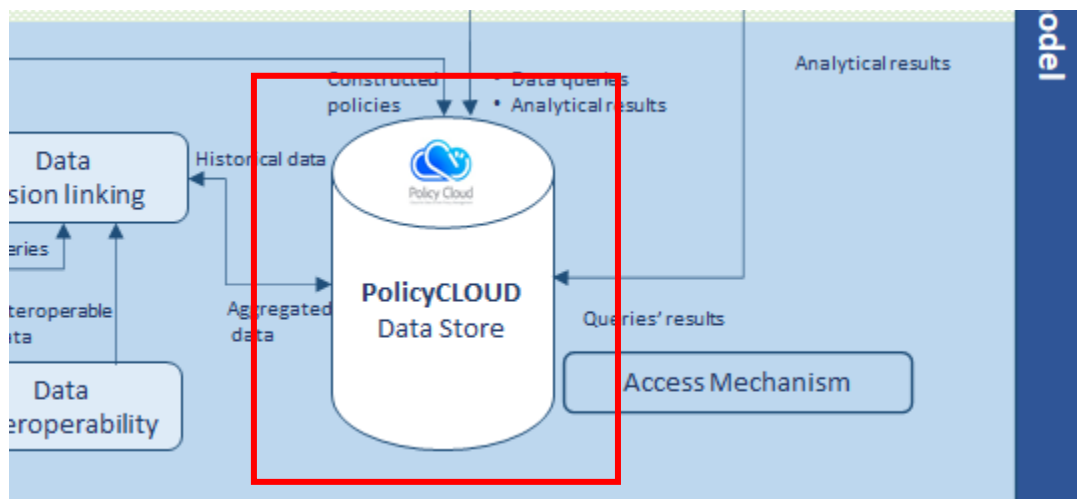


FIGURE 5 – MAIN REPOSITORY (ULTRASCALABLE OPERATIONAL DATABASE)

The central data repository of the PolicyCLOUD is the component where all raw or pre-processed data coming from the data providers is being persistently stored. It is a first-class citizen in the Data Acquisition layer, as all components interact with it, formulating various types of data pipelines. For instance, data being ingested from the gateway can be directly inserted to the data store, or can be ingested after being pre-processed by other components of this layer, like the data cleaning mechanism, the data fusion etc.

<sup>1</sup> Colm, Colm. (2017). Multi-entity sentiment analysis using entity-level feature extraction and word embeddings approach. 733-740. 10.26615/978-954-452-049-6\_094

In cases of data pipelines, the Apache Kafka data queue of the data acquisition layer is used in front of the datastore, and the latter takes data from the queue via its direct connector.

Moreover, the central data store is interacting with all the analytical tools coming from the Data Analytics layer. Those components need to retrieve the data that has been provided in order to perform their analytics, and might need to store intermediate results (i.e. the result of a training algorithm). Those results must be also retrievable by the Policy Development Toolkit, which will provide the results into the Data Visualization component, to visualize them via the use of charts, so that the end-user (i.e., the policy maker) can have a graphical view of the analysis.

#### SOURCE CODE AND INTEGRATION DETAILS

The data store of PolicyCLOUD provides several connectivity methods for all the aforementioned components to integrate. It provides a standard JDBC implementation, a direct API that allows the consumers of the data to connect directly to the storage engine, and various connectors to facilitate its integration with popular analytical frameworks that other components might use, like Spark, Kafka, Flink etc. More information on these interfaces can be found in “D4.4 Reusable Model & Analytical Tools: Software Prototype 2”. Moreover, it has been containerized and has been used in deployments that rely on the Kubernetes deployment orchestration framework. As a result, it is capable of being deployed on the Cloud Environment of the PolicyCLOUD. The component is under proprietary rights of LXS; however, it is accessible to the members of the consortium and has been uploaded in the private gitlab repository that can be found at:

[http://registry.grid.ece.ntua.gr/pavlos\\_LXS/lxs-store](http://registry.grid.ece.ntua.gr/pavlos_LXS/lxs-store)

To build the image locally, the user needs to execute the following command:

```
docker build -t policy-store
```

Then, having the docker image locally, the container can start issuing the following command:

```
docker run -d -p 1529:1529 --name datastore --env KVPEXTERNAL_IP='datastore!9800' policy-store
```

## 2.2.6 Data Cleaning

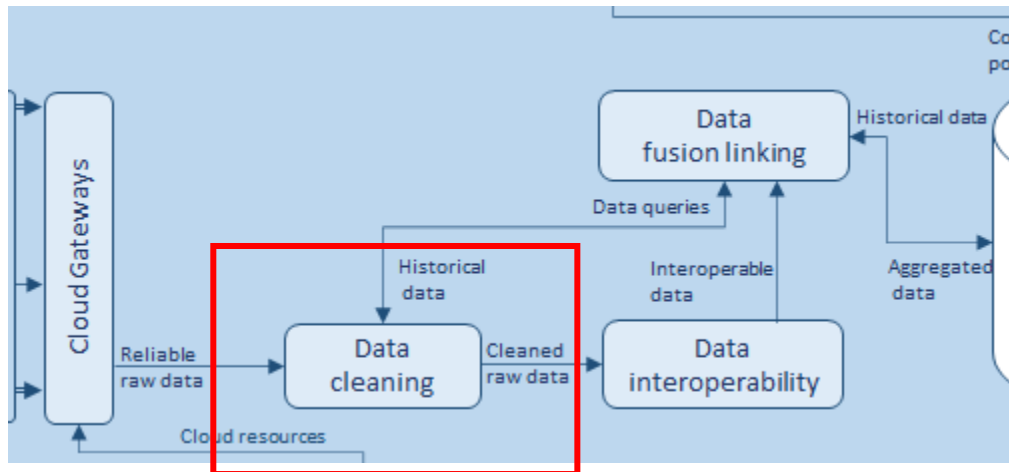


FIGURE 6 - DATA CLEANING

The Data Cleaning component is responsible for undertaking all the processes regarding the data validation and data cleaning of all the incoming heterogeneous data in the PolicyCLOUD platform. Thus, it provides a specific prototype as documented in deliverable D4.4 [12] of the project, which can be utilized by the rest of the PolicyCLOUD components ensuring data accuracy and consistency of the incoming datasets. As a result, the Data Cleaning component implements all the processes that identify inaccurate or corrupted datasets that may contain inaccurate, incorrect, incomplete or irrelevant data elements and consequently replace, modify or delete these data elements safeguarding the reliability and appropriateness of the incoming data information.

### SOURCE CODE AND INTEGRATION DETAILS

In order to successfully implement all the aforementioned actions, the second (updated) prototype of the Data Cleaning component consists of three (3) discrete services, namely the ValidationService, the CleaningService, and the VerificationService. Further information regarding these services and the technologies and techniques that are exploited can be found in D4.4. It should be noted that this prototype can work both as a standalone component, and as an integrated part with the rest of the PolicyCLOUD components.

The second software prototype of the Data Cleaning is provided in PolicyCLOUD's GitLab repository and can be found under the URL <https://registry.grid.ece.ntua.gr/george/cleaninginteroperability>

## 2.2.7 Enhanced Interoperability

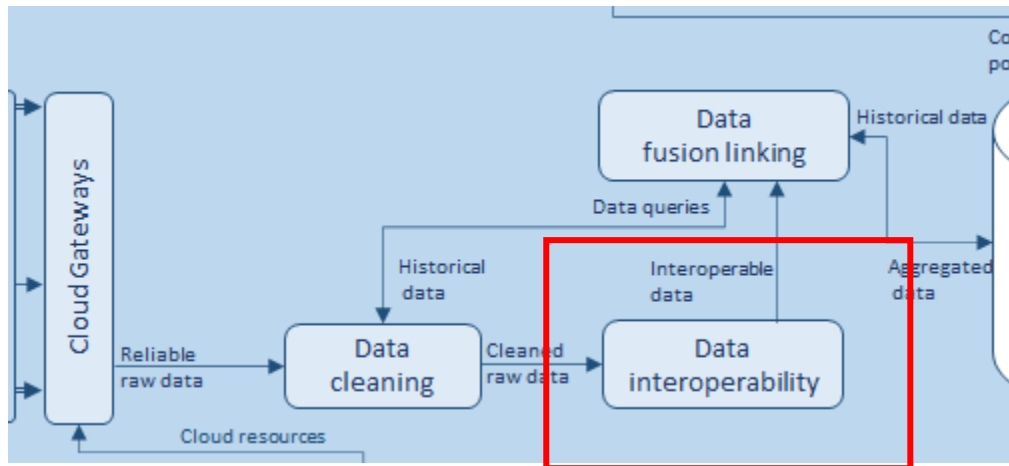


FIGURE 7 – ENHANCED INTEROPERABILITY

Mapping and creating interoperable data depend on providing semantic and syntactic interoperability across diverse systems, data sources and datasets. To this end, PolicyCLOUD's Interoperability Component aims to enhance interoperability into PolicyCLOUD project based on data-driven design by the utilization of linked data technologies, and standards-based ontologies and vocabularies, coupled with the use of powerful tasks from the domain of Natural Language Processing (NLP), in order to improve both semantic and syntactic interoperability of data and datasets. To this end, the Enhanced Interoperability Component consists of several methods, and sub-tasks, which are further separated into different layers of the overall proposed pipeline. The initial design of these layers, that integrate Semantic Web technologies with Natural Language Processing (NLP) technologies and tasks, includes complete workflows and pipelines for annotating cleaned data, which are being sent into the Enhanced Interoperability Component from the Data Cleaning Component.

Hence, several subtasks and functionalities have been introduced and utilized so far in this component and in the context of its second prototype. More specifically, as also analyzed thoroughly in D4.4 Software Prototype 2, the Enhanced Interoperability mechanism incorporates two main subcomponents, the Linguistic & Semantic Analysis with NLP (1st sub-component) and the Ontology Mapping (2nd sub-component). Furthermore, the abovementioned subcomponents utilize different subtasks from the fields of NLP and Semantic Web, such as Named Entity Recognition (NER), JSON URI annotation, Topic Modeling, and Ontology Mapping.

Moreover, under the scope of the second prototype, introduced in deliverable D4.4 Software Prototype 2 the above subtasks and the Enhanced Interoperability Component are provided as a docker installation with a corresponding Flask application coupled with the utilization of Swagger UI will be implemented in order to provide a REST application interface following the OpenAPI specification and in order any stakeholder or component to be able to utilize the above-mentioned methods.

### SOURCE CODE AND INTEGRATION DETAILS

Furthermore, software prototype code can be found on the PolicyCLOUD's Gitlab repository, which is accessible to the members of the consortium. As analyzed before, the overall code has been deployed and executed in as a docker image and is accessible under this specific URL <https://registry.grid.ece.ntua.gr/george/cleaninginteroperability>

The full list of the tools and libraries that need to be installed and full configuration and dependencies details about the setup are available to the corresponding README file (<https://registry.grid.ece.ntua.gr/george/cleaninginteroperability/-/blob/master/README.md>).

## 2.3 Policy Development Toolkit

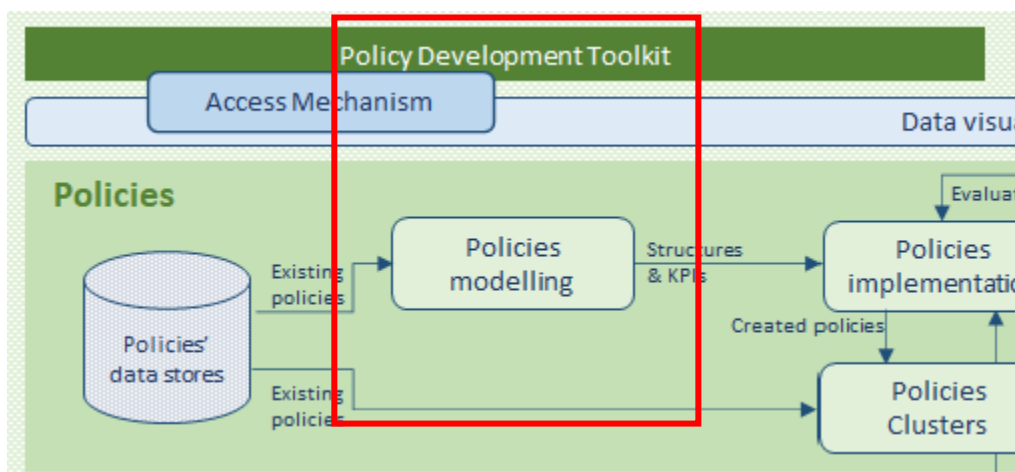


FIGURE 8 - POLICY DEVELOPMENT TOOLKIT (PDT)

As a web application, PDT front-end accepts http/https requests from the policymakers - end users to evaluate policies by the execution of analytics tools which calculate the KPIs related with the policies. Web sockets are employed from the PDT-backend for the User notification regarding the status of the submitted jobs.

PDT frontend consumes most of the REST API exposed by the PDT-backend and the Analytics Tools. In addition, it consumes the Authentication API for the User authentication process. The results from the analytics tools are visualized by the Data Visualization component.

### SOURCE CODE AND INTEGRATION DETAILS

The PDT-front end is being served by the NGINX web server, which is dockerized along with the PDT frontend code.

The installation process and configuration details about the setup is available to the README file (<https://registry.grid.ece.ntua.gr/kostas/pdt/-/blob/master/README.md>).

The PDT source code (<https://registry.grid.ece.ntua.gr/kostas/pdt/-/tree/master>) hosts has been separated from the PDT Frontend and the Visualization component, the PME (Policy Model Editor) and the IM (Incentive Managements components. In order to speed-up the merging process of the source contributions from the components' developers, the dockerized compilation and the deployment of the app with the web server into the RICAS-BARI VM, a new bash script has been put in place to automate the procedure.

Finally, regarding the PDT-backend, the source code is open and can be found at the following link:

[http://registry.grid.ece.ntua.gr/pavlos\\_LXS/pdt-backend](http://registry.grid.ece.ntua.gr/pavlos_LXS/pdt-backend)

## 2.4 Data Visualization

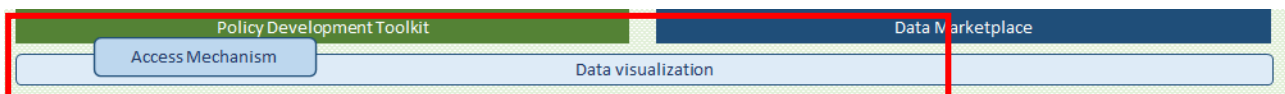


FIGURE 9 - DATA VISUALIZATION

During the second year of the project, the visualization component has been integrated completely inside the PDT, although not all charts can yet be accessed through the PDT real flow. The visualization component can be used to plot the results of the different analytics defined in the project, as the final step of the PDT defined flow.

For demo purposes, some visualization examples, with those charts not accessible through the PDT normal flow, have been created for each pilot in order to show how the results of the gathered data in the platform will be displayed at the end of the process. They can be accessed through the PDT URL, but with a specific URL termination. Most of these examples has been created with real data and JSON structure, agreed with WP4, in order to have them exactly as they will be shown, when the integration is completely finalized.

### SOURCE CODE AND INTEGRATION DETAILS

As the visualization component has been integrated inside the PDT the full source code has been upload into the PDT code repository available here: <https://registry.grid.ece.ntua.gr/kostas/pdt>

## 2.5 Authentication & Data Governance Model

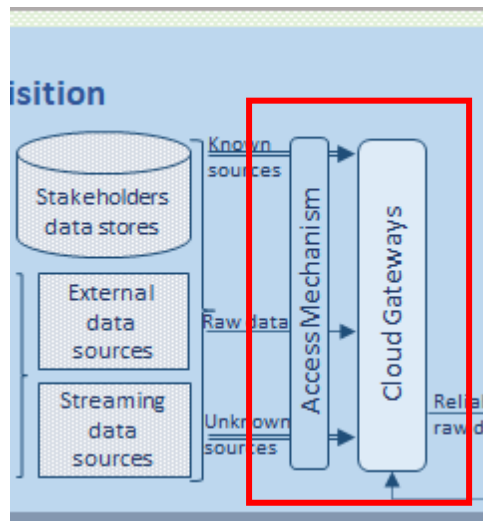


FIGURE 10 - DATA GOVERNANCE MODEL

The Attribute Based Access Control (ABAC) component is responsible for intercepting an access request and evaluate it based on the currently implemented policies and user attributes. The component will generate an ALLOW or PERMIT verdict, based on the aforementioned factors. The user attributes required for the evaluation are retrieved in a safe way via a Keycloak server that acts both as an Attribute Provider and a User Authenticator.

The first version of the component consisted of the ABAC Server, the ABAC Client Filter and the connected Keycloak server instance. The Keycloak Server can be configured to host multiple client applications with the same user base, in order to accommodate multiple pilots operating simultaneously. The continued development of the platform created the need to include a new component in this version, the ABAC Proxy, which helps integrate ABAC with non-Java applications.

The installation process for all three of the subcomponents, as well as instructions for their connection can be found in greater detail at deliverable “D3.2 Cloud Infrastructure Incentives Management and Data Governance: Software Prototype 1” [6].

In the first version, a web client app utilizing the component was accessible at [polycycloud.euprojects.net](http://polycycloud.euprojects.net) while the Keycloak server is available at [polycycloud-auth.euprojects.net](http://polycycloud-auth.euprojects.net). In the current version, the component has been integrated with the PDT and thus the user is prompted to first login to Keycloak in order to gain access to it, as shown on Figure 11.

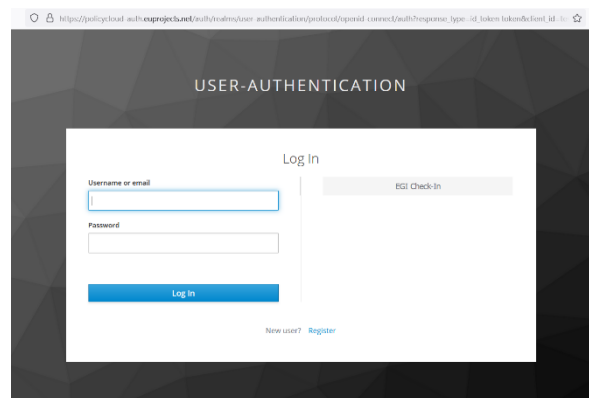


FIGURE 11 - KEYCLOAK LOGIN

After a successful login, the ABAC Engine evaluates the request and permits access to the PDT or denies it accordingly. Keycloak also acts as an attribute provider for the user attributes in the profile page of the PDT. Another useful feature implemented is EGI Check-In, as also shown on the right-hand side of Figure 11. This provides an alternate login option to the platform, via academic or social credentials, as shown on Figure 12.

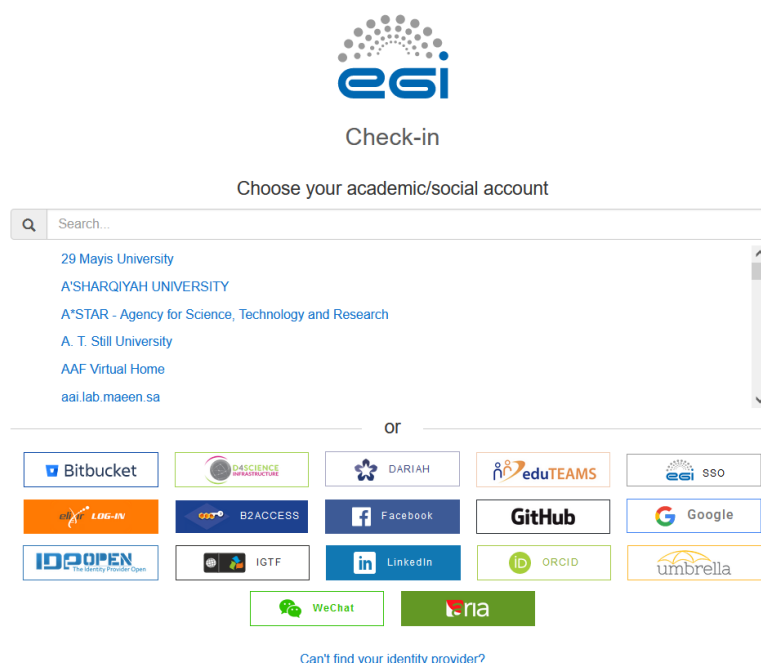


FIGURE 12 - EGI CHECK-IN

This feature allows users that are not already registered in the PolicyCLOUD ecosystem have access to its tools, thus increasing even further the potential user base of the project.

### SOURCE CODE AND INTEGRATION DETAILS

The source code of the components will be added to the main repository after the first integration round.

## 2.6 Data Marketplace

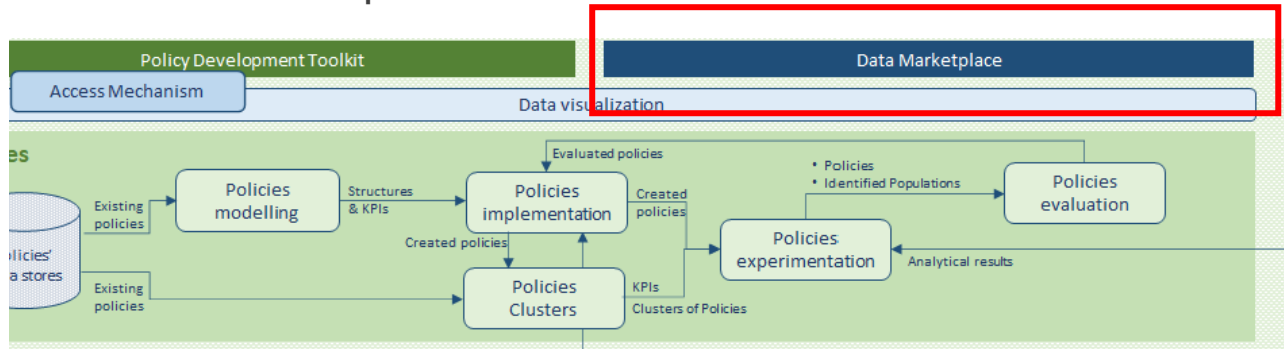


FIGURE 13 – DATA MARKETPLACE

The Data Marketplace is a standalone platform of the PolicyCLOUD that aims to create a community of users who will be able, through this platform, to provide and share various ready-to-use solutions/tools to various subjects and fields of use, related to the areas of interest of the PolicyCLOUD. The assets that may derive/result from the separate procedures and mechanisms that are implemented in the PolicyCLOUD platform are going to be exploited by offering them to the Data Marketplace's users who may be either project's partners or interested third parties.

According to these and the specifications that have been described in the Data Marketplace's deliverable D7.4, an integrated and public web-based environment was developed that introduces several APIs in order to manage the requirements and the functionalities of the Data Marketplace and its objectives. In more technical details, the Marketplace has been structured and developed having two core components, the back-end which contains in a structured way the information and stores the assets offered by the Marketplace, and the front-end which presents to the users the offered contents, allowing them to interact with the platform in an easier way, using its web-based user interface.

All this information, i.e., the objectives, the architecture, the technical details and the interfaces of the Data Marketplace, are already described in the Data Marketplace's relevant deliverables – D7.4 and D7.5 and they will be further extended in their updated versions.

### SOURCE CODE AND INTEGRATION DETAILS

The source code of the Data Marketplace's components will be available through the project's main repository after its first integration/deployment that is expected to take place in January 2022 (M25).

## 2.7 Incentives Management

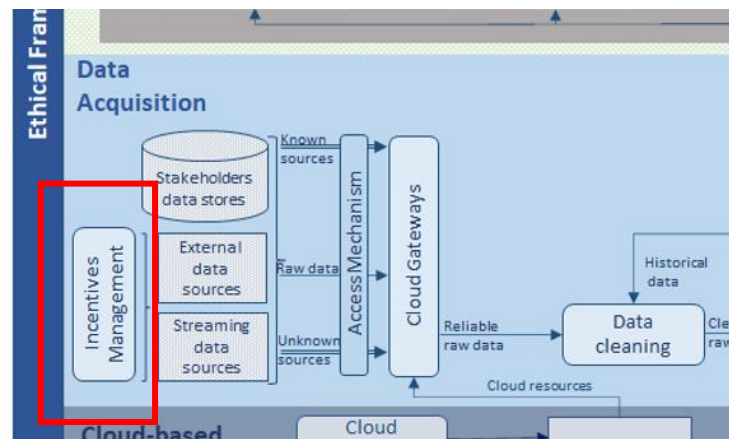


FIGURE 14 – INCENTIVES MANAGEMENT

During the second year of the project, the first version of the incentives management tool has been developed. In order to integrate this component into the project infrastructure, this component has been divided in two parts. The Frontend part, that provides the GUI, has been integrated inside the PDT, and the Backend part that exposes APIs to interact with the component database.

### SOURCE CODE AND INTEGRATION DETAILS

As the frontend part of incentives management component has been integrated inside the PDT, the full source code of this component has been uploaded into the PDT code repository available here: <https://registry.grid.ece.ntua.gr/kostas/pdt>

The backend part of the incentives management component has his own git repository here [https://registry.grid.ece.ntua.gr/ana.pontual/incentivesmanagement\\_be](https://registry.grid.ece.ntua.gr/ana.pontual/incentivesmanagement_be)

The backend is being served with a NGINX web server and a Flask server (Python). The component has been dockerized to facilitate the deployment.

### 3 Source code management

The development of such a large European Project can be a very complex task. To ensure code quality and to test the developed components (designed and implemented by the partners) integration guidelines and a centralized platform are vital. The integration guidelines are the necessary coding principles that should be followed as well as the steps that a developer should followed in order to be aligned with the ethical framework that is presented to the project.

#### UPDATES

During the second year an upgrade to the main platform took place giving us the opportunity to unified security policies to GitLab. Users can now require Dynamic Application Security Testing (DAST) and secret detection scans to run on a regular schedule. That can also be done as part of project CI pipelines.

Also, a more user-friendly wiki integrated module was installed for code documentation and description.

During the second round, we changed the way that the CI/CD template-based approach for cluster management was working. The cluster management is vital for testing and benchmarking as give us the ability to manage Kubernetes clusters and to improve PolicyCLOUD applications availability on the cluster. Now we can easily create a cluster management project for each use case from the project template and fully control the applications through there. The projects that will be created using the template can contain the code needed for cluster management jobs.

We switched to Trivy engine for container scanning. This change provides us with more vulnerability intelligence updates and with more accurate results in relation to the operating system. Trivy will scan Infrastructure as Code (IaC) files of Kubernetes in order to detect potential configuration issues that expose the deployment of PolicyCLOUD components to potential risks.

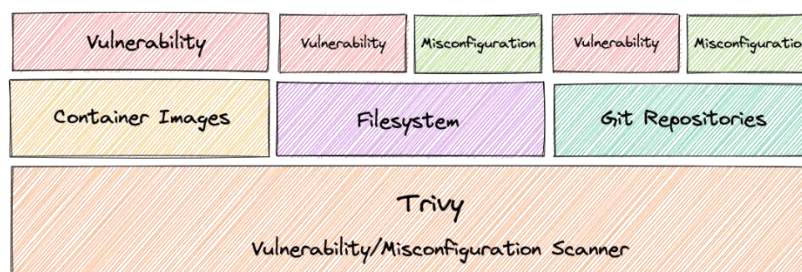


FIGURE 15 - TRIVY COMPREHENSIVE SCANNER

The new version supports DevOps Research and Assessment (DORA) metrics, where we can have insights of merge requests. It is also available through the integration portal where a representative chart is included. Now we can use the chart to identify how long it takes to merge requests that are deployed to the production environment. Moreover, give us the opportunity to have a better overview of the throughput across the components that we use to PolicyCLOUD.

A new pipeline graph that provided a visual representation of the jobs that are hosted to our environment was incorporated. That new visual tool helps us to track the progress of the PolicyCLOUD jobs and identify the integration goals that we can have.

### ANALYTICS

The project now has more than 25 subprojects (from 13 that we had last year)

We have 5 new members to the platform (23 instead of 18 that we had last year)

We made a group to better manage a complicated super-component (integration between repository, marketplace, and PDT).

Figure 16 - Integration Server Analytics (below) depicts the momentum of integration starting from the beginning of the project.

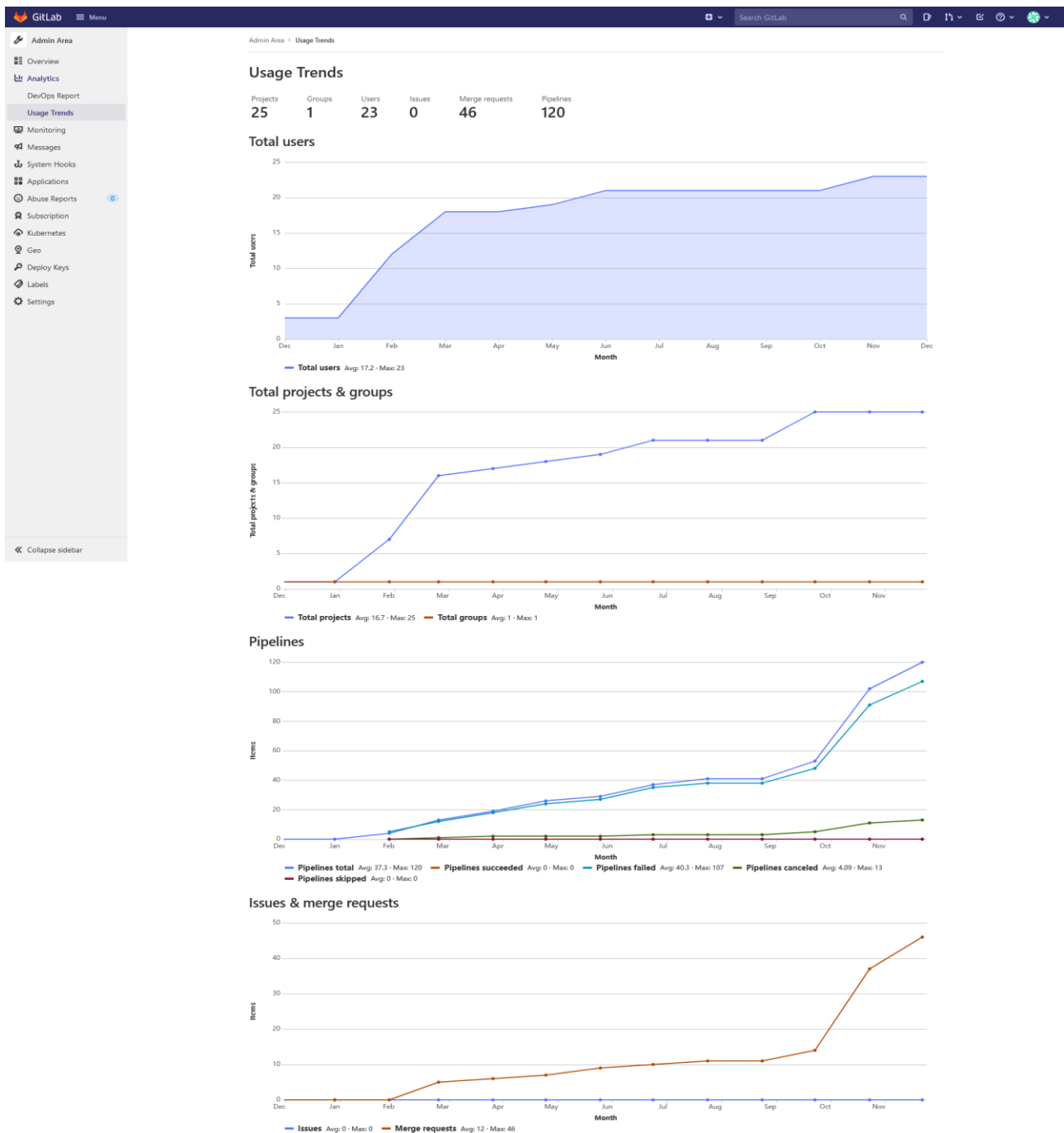


FIGURE 16 - INTEGRATION SERVER ANALYTICS

## CODE

During the second year we switched to a new integration server which is hosted to our main PolicyCLOUD cluster. During the first year the server was hosted to NTUA/ICCS as we had first to setup all the necessary components and environments for the cluster. Now we are fully hosted at RECAS-BARI where we can monitor every process and change to the ecosystem of PolicyCLOUD.

The link for the GitLAB repository server is the <https://registry.grid.ece.ntua.gr/>

## 4 Cloud Environment

The PolicyCLOUD cloud-based infrastructure is composed of IaaS-type cloud resources provided by EGI. From a technical perspective these resources are provided by RECAS-BARI [5], one of the EGI OpenStack providers selected via an open call. As part of the EGI Federated Cloud Infrastructure, the cloud provider provides users with an API-based service to enable the management of Virtual Machines and associated Block Storage, and the connectivity of the Virtual Machines among themselves and third-party resources. In the context of the PolicyCLOUD project the cloud provider is offering:

- 68 vCPU cores, 308 GB of RAM, 2TB of block storage and 50-100GB of object storage, and
- a PaaS orchestrator, with no additional costs, to facilitate the deployment and the operation of Kubernetes clusters where members of the project can use to host services and run pilots.

For more details, please refer to D3.2 - Cloud Infrastructure Incentives Management and Data Governance Software Prototype [6].

During 5th PolicyCLOUD General Assembly on 23rd-25th Nov 2021 we clarified that the project was using 40% of the allocated resources, and there are no future plans to process Big Data. The available capacity should be capable of accommodating the current and future requirements of PolicyCLOUD until the end of the project.

Moreover, also in the 5th GA, EGI has offered the spare capacity at RECAS-BARI to host the Data Marketplace.

EGI has been monitoring the usage of cloud resources and here is a summary up until November 2021.

- Average number of Virtual Machines running: 20 (see Figure below)
- Average number of vCPUs: 27; 40% of the 68 vCPUs agreed in the SLA [13]
- Average amount of RAM memory: 139GB; 45% of the 308GB agreed in the SLA [13]
- Average amount of storage: 587GB; 29% of the 2TB agreed in the SLA [13]

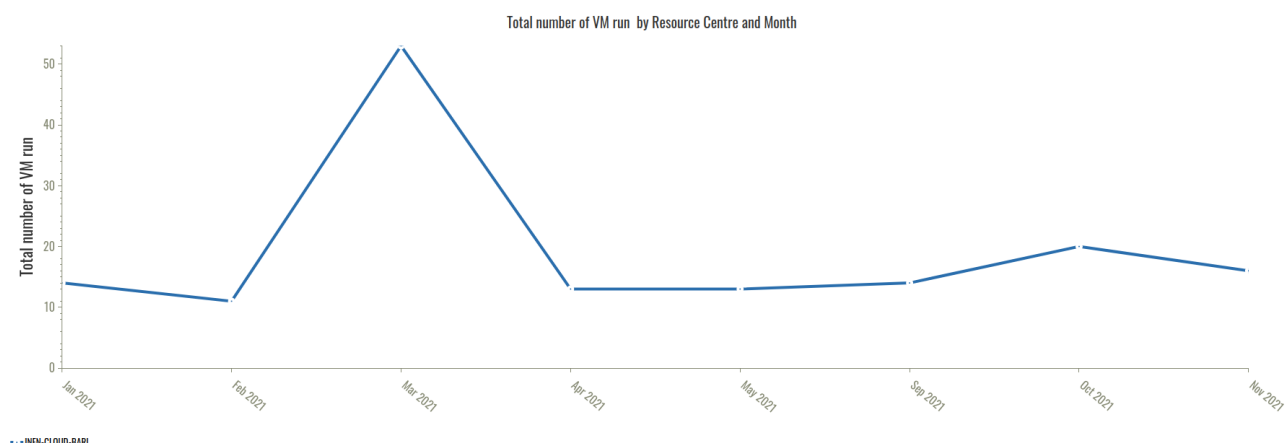


FIGURE 17 - UTILIZATION GRAPH

In summary, PolicyCLOUD has been using approximately around 40% of the allocated cloud capacity.

In the 5th General Assembly (23rd-25th November 2021) the project concluded the following:

- The available capacity is enough to accommodate future requirements until the end of the project.
- The Data Marketplace developed by T7.2 will be hosted using the spare capacity available in the cloud infrastructure. There are ongoing discussions at the time of this writing to define the computational requirements to host the Data Marketplace.

## 5 Conclusion

At this phase of the project, the outcomes of this deliverable provided valuable input to the integration orchestration. All these details are related to the design of the overall architecture of the platform and will be used to support the installation manual of the PolicyCLOUD environment. In parallel, security practises and interoperability issues will be identified through analysis in order to comply the environment with software design and architecture standards.

The main scope is to depict the exact state of each component via a brief description of it, a figure which positions the component into the architecture and technical details of the source code and repository.

The final updated version of this deliverable will include the optimized configurations that will be integrated after the testing phase.

## References

- [1] GitLab. [Online] <https://about.gitlab.com/>
- [2] NGINX. [Online] <https://www.nginx.com/>
- [3] OpenWhisk. [Online] <https://openwhisk.apache.org/>
- [4] PolicyCLOUD. D4.2 Reusable Model & Analytical Tools: Software Prototype 2. Biran Ofer. 2021
- [5] Recas Bari. [Online] <https://www.recas-bari.it/index.php/it/>
- [6] PolicyCLOUD. D3.2 Cloud Infrastructure Incentives Management and Data Governance: Software Prototype 1. Ledakis Giannis. November 2020.
- [7] PolicyCLOUD. D6.2 Integration of Results PolicyCLOUD Complete Environment. Michael Panayiotis, Moulos Vrettos. December 2020.
- [8] PolicyCLOUD. D2.6 Conceptual Model Reference Architecture\_v2.0, Tsanakas Panayiotis, Michael Panayiotis, Moulos Vrettos. June 2021
- [9] PolicyCLOUD. D3.5 Cloud Infrastructure Incentives Management and Data Governance: Software Prototype 2. Ledakis Giannis, Oikonomou Konstantinos. October 2021.
- [10] PolicyCLOUD. D4.3 Reusable Model & Analytical Tools: Design and Open Specification 2, Yosef Moatti Yosef. August 2021.
- [11] APACHE KAFKA, <https://kafka.apache.org/>
- [12] PolicyCLOUD. D4.4 Reusable Model and Analytical Tools Software Prototype 2. Moatti Yosef. October 2021
- [13] SLA between PolicyCLOUD and EGI: <https://documents.egi.eu/document/3667>