



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

D6.2 INTEGRATION OF RESULTS: POLICYCLOUD COMPLETE ENVIRONMENT

Dissemination Level	PU
Due Date of Deliverable	31/12/2020 (M12)
Actual Submission Date	29/12/2020
Work Package	WP6, Use Case Adaptation, Integration & Experimentation
Task	6.2 Integration
Type	Demonstrator
Approval Status	
Version	V0.9
Number of Pages	p.1 – p.19

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.



Versioning and Contribution History

Version	Date	Reason	Author
0.1	23/11/2020	Initial ToC	ICCS
0.2	11/12/2020	First integrated version (contribution from ATOS, LXS, EGI and UBI)	ICCS
0.3	12/12/2020	Contribution from IBM and LXS	ICCS
0.4	15/12/2020	Contribution from UPRC	ICCS
0.5	21/12/2020	Refinements	ICCS
0.6	23/12/2020	Version after the first review (LXS)	ICCS
0.7	27/12/2020	Version after the second review (IBM)	ICCS
0.8	28/12/2020	Quality check	UPRC
0.9	29/12/2020	Last refinements. Document ready for submission	ICCS

Author List

Organisation	Name
ICCS/NTUA	Panayotis Michael, Vrettos Moulos
LXS	Jose María Zaragoza
IBM	Ofer Biran
ATOS	María Ángeles Sanguino, Jorge Montero, Ana Luiza Pontual, Miquel Milà, Ricard Munné
EGI	Giuseppe La Rocca
UPRC	George Manias, Argyro Mavrogiorgou, Athanasios Kiourtis, Ilias Maglogiannis, Nikitas Sgouros
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
EBPM	Evidence Based Policy Making
EC	European Commission
EOSC	European Open Science Cloud
KPI	Key Performance Indicator

Contents

Versioning and Contribution History.....	2
Author List	2
Abbreviations and Acronyms	2
Executive Summary	5
1 Introduction	6
2 Components.....	7
2.1 Data Acquisition.....	7
2.1.1 Cloud Gateways & APIs for Efficient Data Utilization.....	7
2.2 Data Analytics	8
2.2.1 The Data Acquisition and Analytics (DAA).....	8
2.2.2 DAA API Gateway.....	8
2.2.3 Situational Knowledge Acquisition & Analysis.....	9
2.2.4 Opinion Mining & Sentiment Analysis Component	9
2.2.5 Operational Data Repository	9
2.2.1 Data Cleaning.....	10
2.2.2 Enhanced Interoperability.....	10
2.3 Policy Development Toolkit	11
2.4 Data Visualization	12
2.4.1 Visualization component.....	12
2.5 Data Governance Model	12
2.5.1 Attribute Based Access Control (ABAC) Component	12
3 Source code management	14
4 Cloud Environment	17
5 Conclusion.....	18
References.....	19

List of Figures

Figure 1 Overall core component architecture	6
Figure 2 Keycloak Login.....	13
Figure 3 Successful login and ABAC Permit.....	13
Figure 4 Keycloak User Attributes	13
Figure 5 PolicyCLOUD Projects	14
Figure 6 GitLAB Dashboard.....	15
Figure 7 GitLAB Statistics.....	16

Executive Summary

This deliverable has been released on M12 of the project, and its main objective is to specify the initial integration results between the PolicyCLOUD components.

This deliverable has two main pillars: 1.-define common practices for integration and validation of the outcomes of the project, and 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 will be applied, as it is considered the most suitable policy according to the project characteristics. Also, GitHub's issue reporting tool will be adopted, as it is fully integrated with GitHub's features. The test bed to support the demonstrators will be deployed over EGI's (EGI) infrastructure where flexibility is one of the main features.

This deliverable abstractly incorporates all the changes and implementations that WP2, WP3, WP4 and WP5 had made during the last year. More details about the components and the actual implementation can be found to the relative WP deliverables.

1 Introduction

In the following section the overall Architecture that was presented in D2.2 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 that are necessary for the full deployment.

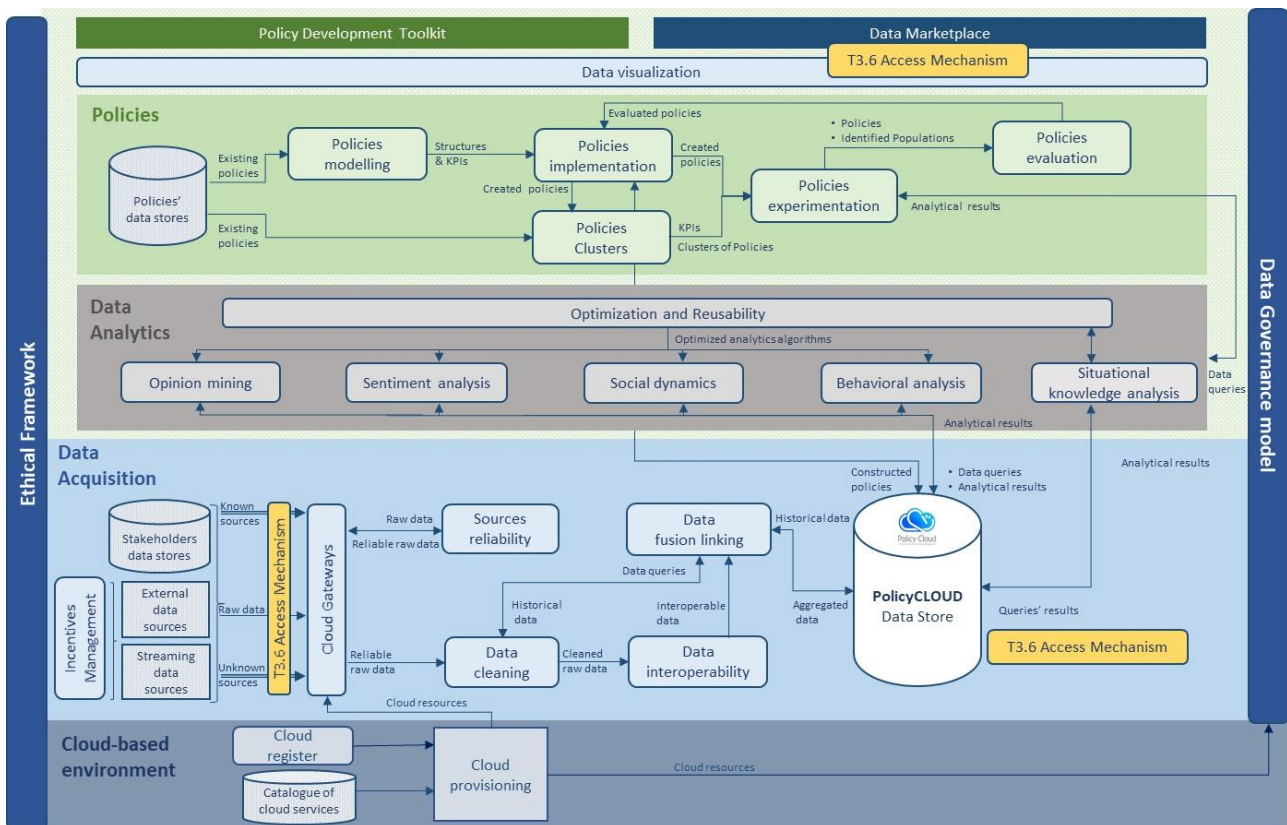


FIGURE 1 OVERALL CORE COMPONENT ARCHITECTURE

2 Components

In the following section a description of the component and a reference to the source code is provided. Each component has unique characteristics and where difference technologies are integrated.

2.1 Data Acquisition

2.1.1 Cloud Gateways & APIs for Efficient Data Utilization

The Cloud Gateway and API component will enhance 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 a request by invoking multiple microservices and aggregating the results. Hence, it will enhance the design of resources and structure, add dynamic routing parameters, and develop custom authorizations logic. PolicyCLOUD's Cloud Gateway and API component will support scalability, high availability, fault tolerance, and shared 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. Currently these sub-components are not integrated with each other or with an authentication mechanism to control access to them.

2.1.1.1 GLOBAL TERRORISM DATABASE COMPONENT (GTD COMPONENT)

The main goal of this sub-component is to obtain data from the CSV file of the GTD and to update the PolicyCLOUD's data stores. The GTD component is invoked either by asynchronous workers as background processes, or by user demand after making request to the appropriate API endpoint.

The main component's core functionalities are written in PHP v7.4. Moreover, in order to create a robust and flexible API service, Laravel framework is being utilized. The service is running on a NGINX server [2] that can scale very well as it can support thousands of connections per worker process and achieve great performance. For the component's temporary database, we used MySQL 5.7 because it is working out of the box with the Laravel's Eloquent ORM¹, but a migration to MongoDB² has already started to be implemented, considering the flexibility, scalability and performance the MongoDB provides. After the first integration round the component will incorporate features and will leverage the advantages that LXS will provide through the PolicyCLOUD data repository.

Source code and an installation manual for the GTD component is available on the following repository: <http://snf-877903.vm.oceanos.grnet.gr/george/policycloud-gtd-service.git>. The installation instructions and project requirements are firmly described inside the components README.md file (<http://snf-877903.vm.oceanos.grnet.gr/george/policycloud-gtd-service/-/blob/master/README.md>).

¹ <https://laravel.com/docs/5.0/eloquent>

² <https://www.mongodb.com/1>

2.1.1.2 TWITTER CONNECTOR COMPONENT

The main goal of the Twitter Connector sub-component is to fetch data from the Twitter platform based on specific hashtags. In this specific use case, the corresponding sub-component filters twitter data based on the high-interest hashtag “jihad”. On top of this, this sub-component relies its functionality on the Twitter API for data collection. One of the major limitations of data collection using the Twitter API is that the client machine always must keep up with the data stream and if it could not then the stream just disconnects. Hence, in order to overcome this limitation, the initial design of this sub-component integrates also with Kafka³ event streaming platform.

The implementation of the core functionalities of this component are written in Python 3.7. On top of this, Flask framework is being used in order to create a robust and flexible API service. Moreover, Kafka event streaming platform is utilized, so raw Twitter data can be processed without worrying about the stream getting disconnected.

Source code and an installation manual is available on the following repository <http://snf-877903.vm.oceanos.grnet.gr/george/policycloud-twitter-service>. The installation instructions and project requirements are firmly described inside the project’s README.md file (<http://snf-877903.vm.oceanos.grnet.gr/george/policycloud-twitter-service/-/blob/master/README.md>).

2.2 Data Analytics

2.2.1 The Data Acquisition and Analytics (DAA)

This Layer is the central layer of PolicyCLOUD, between the cloud infrastructure and 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

³ APACHE KAFKA, <https://kafka.apache.org/>

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.

<http://snf-877903.vm.oceanos.grnet.gr/oshrit/daa>

The setup details for the DAA prototype environment are detailed in “D4.2 REUSABLE MODEL & ANALYTICAL TOOLS: SOFTWARE PROTOTYPE 1”, section 3.2.1 DAA API Gateway.

2.2.3 Situational Knowledge Acquisition & Analysis

In the context of Situational Knowledge Acquisition and Analysis, a SKA-HeatMap v1 component has been provided as first prototype. This component is in charge of providing exploratory analysis over the GTD data base, more concretely the HeatMap prototype will connect with the PolicyCLOUD data storage, retrieve data according an input parameters and return aggregated data in a JSON file that will be visualized with the heat map visualization technique. The heat map will show the incidence of terrorisms attacks grouped by cities and periods of time.

This first prototype work in a standalone version and has been deployed in ATOS local premises. Is expected for future versions have it deployed in the testbed infrastructure and fully integrated with the rest of the PolicyCLOUD components. These are: DAA API Gateway developed in the context of WP4, available as an OpenWhisk function and the PolicyCLOUD storage to store the results.

Similar to what happen with the Opinion Mining & Sentiment Analysis Component, all the information related to this component is written in D4.2[4] 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: <http://snf-877903.vm.oceanos.grnet.gr/nines/heatmap>.

2.2.4 Opinion Mining & Sentiment Analysis Component

At this stage of the project, this component provides the functionality for the sentiment analysis task. It has been done as an isolated component outside the whole workflow due to the initial phases of the integration task. When the integration is done, this component will be accessible through DAA API Gateway developed in WP4 as another analytical service.

All the information related to this component is written in D4.2 (1) 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.

The installation guide, and all the resources needed, are available at the project's provisional git: <http://snf-877903.vm.oceanos.grnet.gr/jorge/sentimentanalysis>

2.2.5 Operational Data Repository

The central data repository of the PolicyCLOUD is the component where all raw data coming from the data providers is being persistently stored. It is a first-class citizen in the Data Acquisition layer, as all components are able to interact with it, formulating various types of data pipelines. For instance, data being ingested from the *gateway* (either via static ingestion or via a data stream) can be directly inserted to the data store, or can be after the interception of other components of this layer, like the *data cleaning* mechanism, the *data fusion* etc. In cases

of data pipelines, all an Apache Kafka data queue is being deployed 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 raw 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.

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 on D4.2 (“REUSABLE MODEL & ANALYTICAL TOOLS: SOFTWARE PROTOTYPE 1”). 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 to be 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 here: <http://snf-877903.vm.oceanos.grnet.gr/pavlos/lxs-store>

2.2.1 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 it was documented in deliverable D4.2 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.

In order to successfully implement all the aforementioned actions, the first prototype of the Data Cleaning component consists of four (4) discrete services, namely the ValidationService, the CleaningService, the VerificationService, and the LoggingService. Further information regarding these services and the technologies and techniques that are exploited can be found in D4.2. It should be noted that for the moment, this prototype works as a standalone component, whereas it is expected that in the next version it will be integrated with the rest PolicyCLOUD components.

The first software prototype of the Data Cleaning is provided in PolicyCLOUD’s GitLab repository and can be found under the URL <http://snf-877903.vm.oceanos.grnet.gr/argyro/cleaning-component>

2.2.2 Enhanced Interoperability

Mapping and creating interoperable data depend on a method of 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, the methods that are being utilized so far in this component are:

Method names
preprocess()
getSentences()
text_to_djson()
processSubjectObjectPairs()

On top of this, it should be noticed that more methods and sub-tasks will be utilized and provided in updated versions and prototypes of this component.

Moreover, under the scope of the 1st Prototype, introduced in Deliverable D4.2 Software Prototype 1 the above methods and the Enhanced Interoperability Component is provided so far through a IPython Notebook⁴ (.ipynb file extension) in order any stakeholder or user to be able to execute the code and identify the outcomes in every step and sub-task of this specific sub-component. In next steps 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.

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 IPython Notebook (.ipynb file extension), which is accessible under this specific URL <http://snf-877903.vm.oceanos.grnet.gr/george/interoperability-component>

The installation process and full configuration and dependencies details about the setup is available to the READ.ME file (<http://snf-877903.vm.oceanos.grnet.gr/george/interoperability-component/-/blob/master/README.md>).

2.3 Policy Development Toolkit

As a web application, PDT front-end accepts http/https requests by the policymakers - end users to evaluate policies by the execution of analytics tools which calculate the KPIs related with the policies. HTTP calls are made towards the PDT backend for all the data provenance actions. Web sockets are also 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.

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

⁴ <https://ipython.org/notebook.html>

The installation process and full configuration details about the setup is available to the READ.ME file (<http://snf-877903.vm.oceanos.grnet.gr/kostas/pdt/-/blob/master/README.md>).

2.4 Data Visualization

2.4.1 Visualization component

The Visualization component is thought as part of the PDT, integrated as a subcomponent of it. However, at this phase of the project this integration still cannot be done, and due to that, the component has been developed as a separated web application.

Visualization component will be responsible for the visualization of the data gathered by the PolicyCloud platform, through the integration with the Data Analytics component, from where data is recovered, according the needs of each pilot use case. In this first version, as this integration with the Data Analytics component has not yet been carried out, the component does not use online, but static data, although already with the agreed data format.

The visualization data web application is a SPA (Single-Page Application) web site where some first visualizations of two agreed use cases are displayed: the “Participatory Policies Against Radicalization” from Maggioli and the “Intelligent Policies for Denomination of Origin” from Sarga.

The installation process for this component is detailed into the READ.ME file available at the project’s provisional git: <http://snf-877903.vm.oceanos.grnet.gr/miquel.mila/visualization/-/blob/master/README.md>.

The full source code of this component is also available in the same git repository:<http://snf-877903.vm.oceanos.grnet.gr/miquel.mila/visualization>.

A more detailed information about the visualization component is available at deliverable “D5.3 CROSS-SECTOR POLICY LIFECYCLE MANAGEMENT: SOFTWARE PROTOTYPE 1”, where this component’s first charts are detailed, as well as the input data format agreed with WP4 for each of them.

2.5 Data Governance Model

2.5.1 Attribute Based Access Control (ABAC) Component

The Attribute Based Access Control (ABAC) component will be responsible for intercepting an access request and evaluating 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 consists 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 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”.

A web client app utilizing the component in its first version can be accessed at policycloud.euprojects.net while the Keycloak server is available at policycloud-auth.euprojects.net. The user is prompted to first login to Keycloak in order to gain access, as shown in Figure 2.

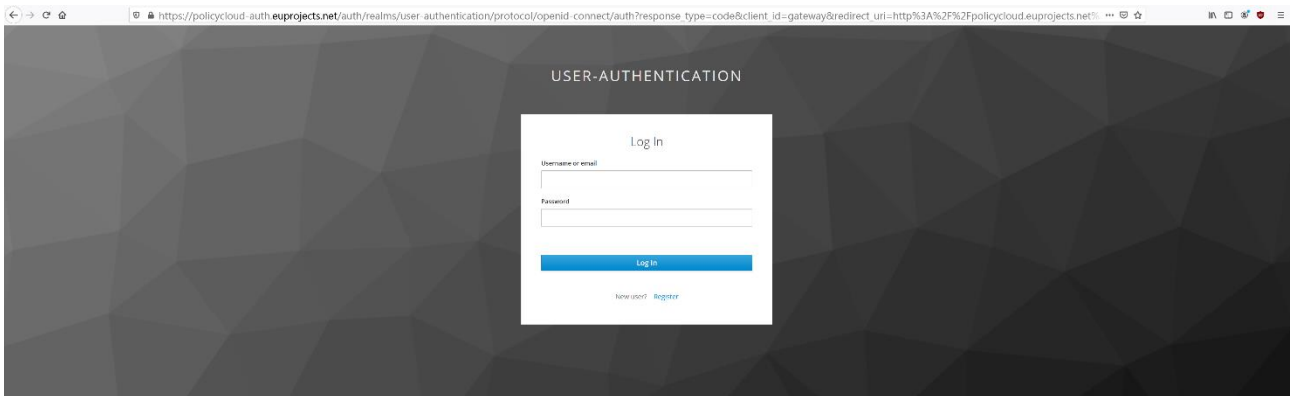


FIGURE 2 KEYCLOAK LOGIN

After a successful login, the ABAC Engine evaluates the request and permits or denies it accordingly. The online example contains a PERMIT-ALL policy for testing purposes, as shown in Figure 3.

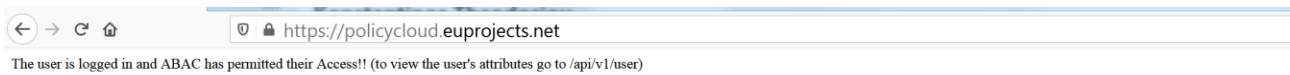


FIGURE 3 SUCCESFULL LOGIN AND ABAC PERMIT

Finally, the api `policycloud.euprojects.net/api/v1/user` can be used to retrieve the user attributes from Keycloak, as shown in Figure 4. These attributes will be evaluated by the ABAC Engine in order to be in compliance with the implemented Policies.

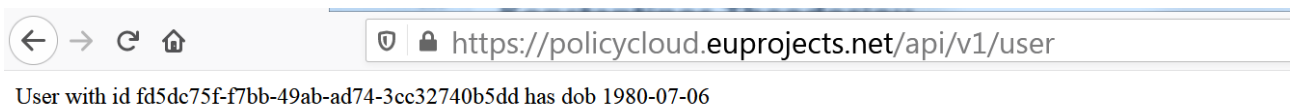
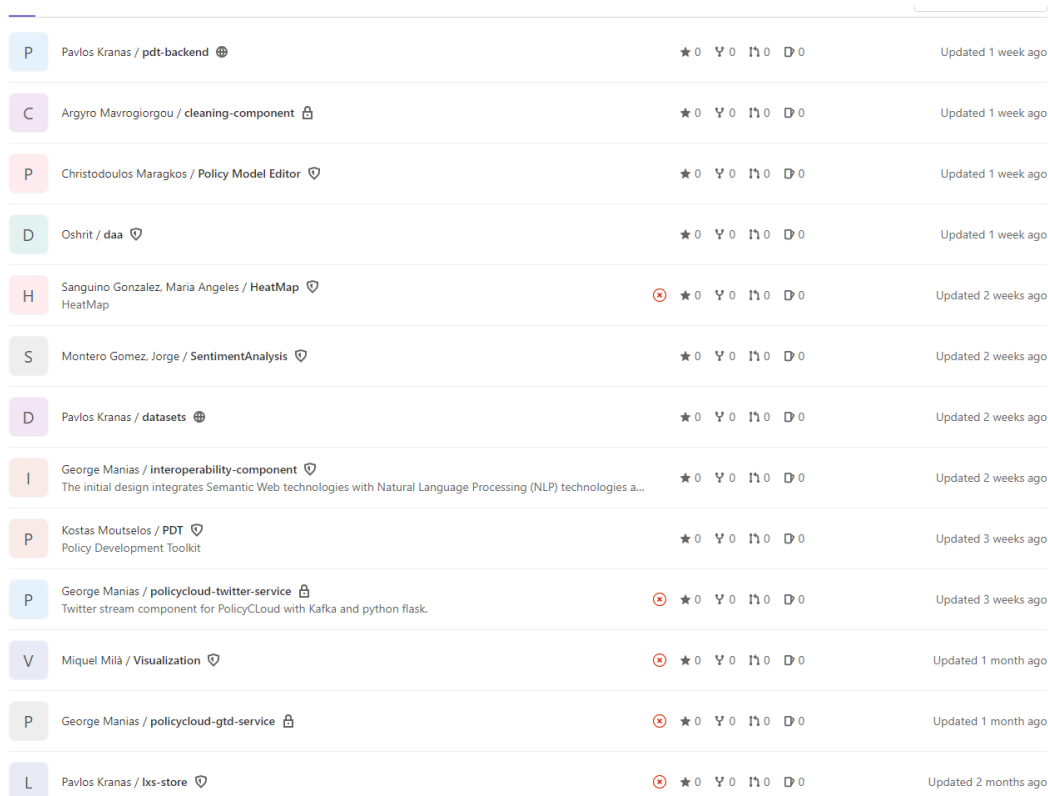


FIGURE 4 KEYCLOAK USER ATTRIBUTES

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

3 Source code management

The development of marketplaces like the one used in the scope of the PolicyCLOUD 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. To track that continues changing environment, we segment the code to repositories as shown in Figure 5



Repository	Owner	Stars	Forks	MRs	Issues	Updated
Pavlos Kranas / pdt-backend	Pavlos Kranas	0	0	1	0	Updated 1 week ago
Argyro Mavrogiorgou / cleaning-component	Argyro Mavrogiorgou	0	0	1	0	Updated 1 week ago
Christodoulos Maragkos / Policy Model Editor	Christodoulos Maragkos	0	0	1	0	Updated 1 week ago
Oshrit / daa	Oshrit	0	0	1	0	Updated 1 week ago
Sanguino Gonzalez, Maria Angeles / HeatMap	Sanguino Gonzalez, Maria Angeles	0	0	1	0	Updated 2 weeks ago
Montero Gomez, Jorge / SentimentAnalysis	Montero Gomez, Jorge	0	0	1	0	Updated 2 weeks ago
Pavlos Kranas / datasets	Pavlos Kranas	0	0	1	0	Updated 2 weeks ago
George Manias / interoperability-component	George Manias	0	0	1	0	Updated 2 weeks ago
Kostas Moutselos / PDT	Kostas Moutselos	0	0	1	0	Updated 3 weeks ago
George Manias / policycloud-twitter-service	George Manias	0	0	1	0	Updated 3 weeks ago
Miquel Mià / Visualization	Miquel Mià	0	0	1	0	Updated 1 month ago
George Manias / policycloud-gtd-service	George Manias	0	0	1	0	Updated 1 month ago
Pavlos Kranas / lxs-store	Pavlos Kranas	0	0	1	0	Updated 2 months ago

FIGURE 5 POLICYCLOUD PROJECTS

To ensure code quality, test the developed product and to track the changes made by different developers, integration guidelines are needed. The integration guidelines are defining the coding standards that should be followed, so that they can ensure that the code is readable and understandable by different developers involved in the process.

One core functionality that is used for code quality is the unit testing, which refers to testing the behaviour of the smallest building blocks. The scope is to identify code-related issues and ensuring the rationality of the produced outcomes. Next step that we will follow is the integration tests of several software components. In that phase we will test the integrated version of them as a whole.

For coordinating the development of the PolicyCLOUD and the co-creation of related policy services, PolicyCLOUD uses GitLab as central code repository and co-creation environment Figure 6 GitLAB Dashboard. Each partner has at least one admin account where he can create repositories for software components. Through that feature, he/she has the freedom to implement specific building blocks, libraries or services and by organizing his own team

or by adding to the repository users from other groups that he/she can fully control the programming process of the aforementioned building block.

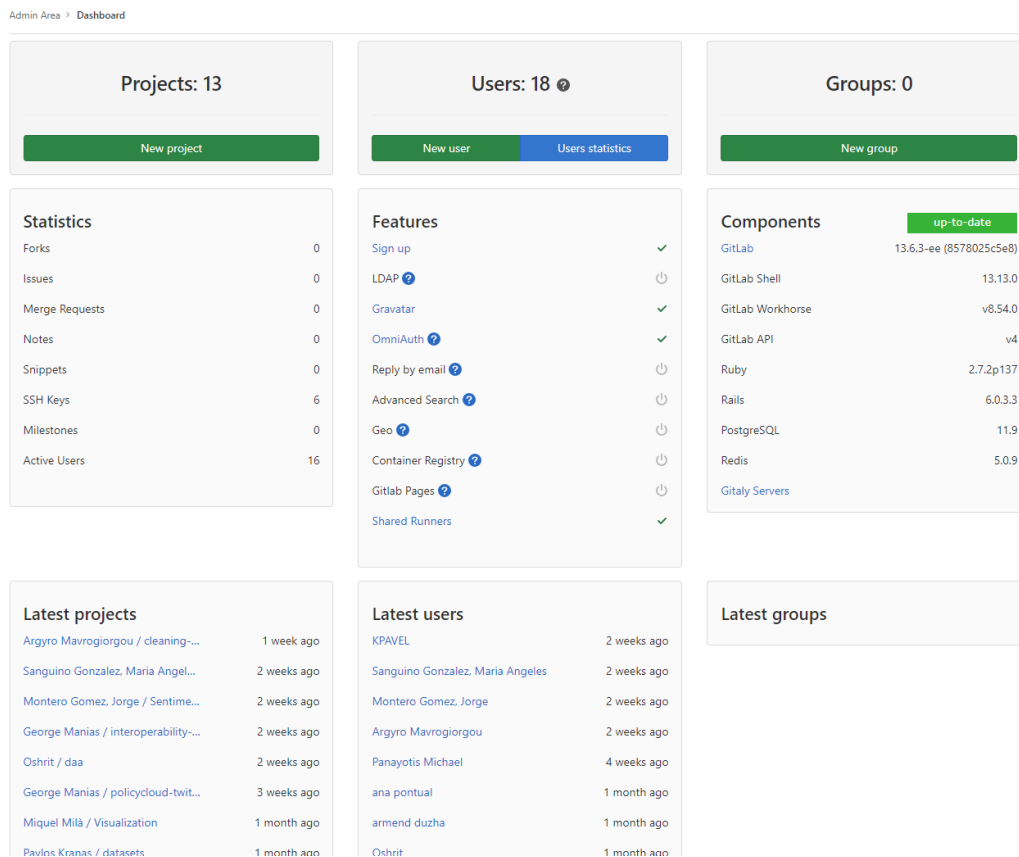


FIGURE 6 GITLAB DASHBOARD

The link for the GitLAB repository server is the <http://snf-877903.vm.oceanos.grnet.gr/>. After the first iteration we will setup a second server. The second one will be hosted to the same cluster as the PolicyCLOUD ecosystem.

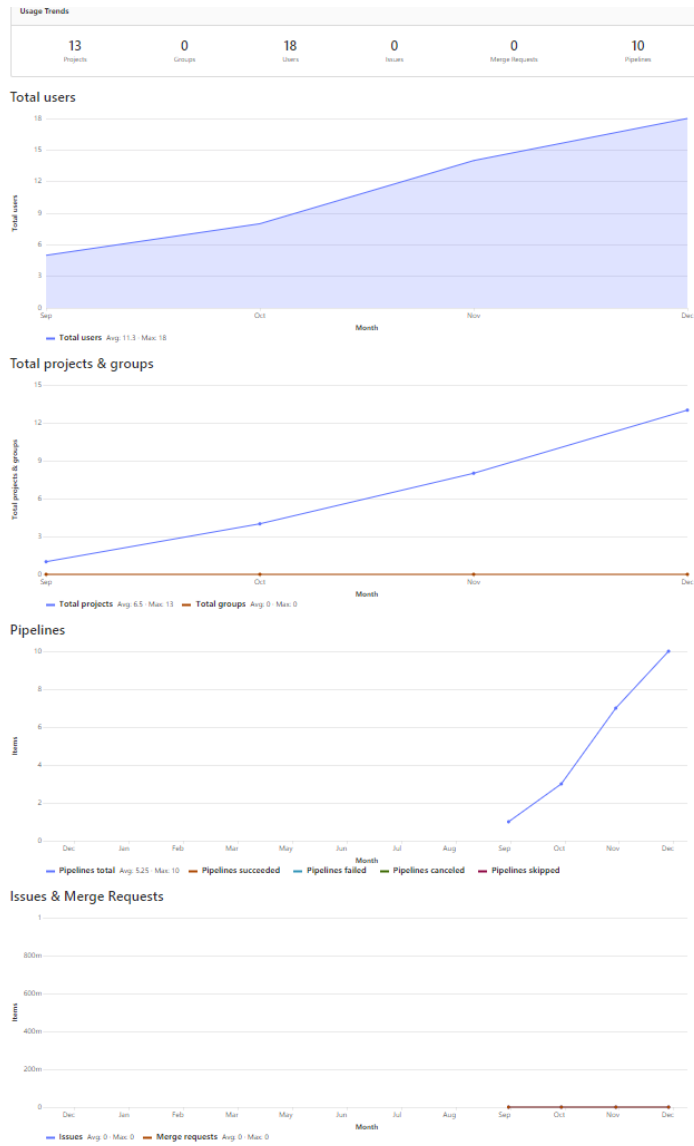


FIGURE 7 GITLAB STATISTICS

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 50-100MB 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].

5 Conclusion

At this initial 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 plan is to update this complementary material in each deployment iteration. So, in each round we will complement the source code with instructions about the new services and the new integration practises that we followed.

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 1. Biran Ofer. 2020
- [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. 2020.