



# 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

### D3.1 CLOUD INFRASTRUCTURE INCENTIVES MANAGEMENT AND DATA GOVERNANCE: DESIGN AND OPEN SPECIFICATION 1

Dissemination Level	PU
Due Date of Deliverable	31/08/2020, Month 8
Actual Submission Date	01/09/2020
Work Package	WP3 Cloud Infrastructures Utilization & Data Governance
Task	T3.1, T3.3, T3.4, T3.6
Type	Report
Approval Status	
Version	V1.1
Number of Pages	p.1 – p.45

**Abstract:** This report will provide the first version of the design of the cloud gateways and the cloud provisioning approaches to ensure utilization of cloud resources as required for the PolicyCLOUD environment. It will also provide the algorithms implementing the incentives management as well as the data governance model and the specification of the tools used to ensure compliance to the model across the complete data path.

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	15/06/2020	ToC	Giannis Ledakis
0.2	26/06/2020	ToC update and sharing of assignments and plan	Konstantinos Theodosiou
0.3	10/07/2020	Section 2 content	Giuseppe La Rocca
0.4	17/07/2020	Section 4 content	Maria Angeles Sanguino, Jorge Montero, Ana Luiza Pontual, Miquel Milà, Tomas Pariente and Ricard Munné
0.5.	20/07/2020	Section 2 updates	Giuseppe La Rocca
0.6	23/07/2020	Section 3 content	Ilias Maglogiannis, Thanos Kiourtis, Argyro Mavrogiorgou
0.7	24/08/2020	All sections in place	Giannis Ledakis, Konstantinos Theodosiou, Konstantinos Oikonomou
0.8	28/08/20	Review of document	Thanos Kiourtis, Javier Sancho
0.9	31/08/2020	Address peer review comments, updating section 2 and 4 based on comments	Giannis Ledakis, Konstantinos Oikonomou, Giuseppe La Rocca
1.0	31/08/2020	Final version	Giannis Ledakis
1.1	01/09/2020	Quality Check performed and comments addressed	Argyro Mavrogiorgou, Giannis Ledakis

## Author List

Organisation	Name
ATOS	Maria Angeles Sanguino, Jorge Montero, Ana Luiza Pontual, Miquel Milà, Tomas Pariente and Ricard Munné
EGI	Giuseppe La Rocca
UBITECH	Giannis Ledakis, Konstantinos Theodosiou, Konstantinos Oikonomou
UPRC	Ilias Maglogiannis, Thanos Kiourtis, Argyro Mavrogiorgou

## Abbreviations and Acronyms

Abbreviation/Acronym	Definition
ABAC	Attribute-based access control
API	Application Programming Interface
CMF	Cloud Management Framework
DB	Database
DSS	Decision Support System
EC	European Commission
EoIs	Expression of Interests
EOSC	European Open Science Cloud
GDPR	General Data Protection Regulation
GTD	Global Terrorism Database
IaaS	Infrastructure as a Service
JSON	JavaScript Object Notation
KPI	Key Performance Indicators
NLP	Natural Language Processing
NoSQL	Non Structured Query Language
PaaS	Platform as a Service
PDT	Policy Development Toolkit
PM	Policy Model
PP	Public Policy
REST	Representational state transfer
SQL	Structured Query Language
VM	Virtual Machine
XML	eXtensible Markup Language

# Contents

Versioning and Contribution History.....	2
Author List.....	2
Abbreviations and Acronyms.....	3
Executive Summary.....	7
1 Introduction.....	8
1.1 Structure of the document.....	8
2 Cloud Provisioning of the PolicyCLOUD Infrastructure.....	9
2.1 The EGI SLA/OLA framework.....	9
2.2 Collection of the technical requirements.....	10
2.2.1 Resource Centre.....	11
2.2.2 Resource summary.....	11
2.2.3 Allocation Types.....	11
2.2.4 Payment Models Offer.....	11
2.2.5 Service Cost.....	11
2.2.6 Other Technical Requirements.....	11
2.2.7 Duration (of the agreement).....	11
2.2.8 Supported Vos.....	11
2.2.9 VO Id card.....	12
2.2.10 Service Level Targets.....	12
2.2.11 Incidents Handling.....	12
2.2.12 Reporting and Violations.....	13
2.3 The Call for Cloud Providers.....	13
2.4 Next Steps.....	14
3 Cloud Gateways & APIs for Efficient Data Utilization.....	15
3.1 Cloud Gateway Capabilities.....	15
3.1.1 Fetching external Resources.....	15
3.1.2 Authentication and Security.....	15
3.1.3 Data filtering.....	15
3.1.4 Monitoring.....	15
3.2 Offering a well-structured and well documented API.....	16
3.3 Technologies & Methodologies.....	17
3.3.1 Services.....	17
3.3.2 Dynamic Service Discovery.....	17
3.3.3 Load Balancing.....	18

3.3.4	Fault Tolerance .....	18
3.3.5	Caching .....	18
3.3.6	Messaging and Message Brokers .....	18
3.3.7	API Gateway .....	19
3.3.8	Data Transformation and Database Adapters.....	19
3.4	Next Steps.....	19
3.4.1	Detailed system requirement analysis for the Gateway's services .....	19
3.4.2	Gateway's Architecture and Design.....	19
3.4.3	API Specification, Structure, and Documentation .....	19
3.4.4	Development and implementation .....	20
3.4.5	Testing and deployment .....	20
4	Incentives Management .....	21
4.1	Next Steps.....	22
5	Data Governance Model, Protection and Privacy Enforcement .....	23
5.1	Data Protection and Access Control.....	23
5.1.1	ABAC Implementations .....	26
5.2	PolicyCLOUD Data Governance Model and Privacy Enforcement mechanism.....	29
5.2.1	PolicyCLOUD Model .....	29
5.2.2	PolicyCLOUD Policy Enforcement .....	29
5.3	Next Steps.....	30
6	Conclusion and Next Steps.....	31
	References.....	32
7	Appendix I - Access the EGI User Community .....	34
7.1	Step 1. Sign up the EGI User Community .....	34
7.2	Step 2. Activate the account.....	35
8	Appendix II - Access the INDIGO PaaS Orchestrator .....	36
8.1	Step 1. Create a user's account.....	36
8.2	Step 2. Set the password for the account.....	39
8.3	Step 3. Access the INDIGO PaaS Orchestrator .....	40
8.3.1	Deploying of a Kubernetes cluster .....	41
8.3.2	Deploying a Spark (in K8s) cluster .....	44

## List of Tables

Table 1 – Incident priority vs. response time.....	13
--	----

Table 2 – Costs fo the cloud provisioning for period M05-M30.....	13
---	----

## List of Figures

Figure 1 – The resources allocation process in EGI.....	9
Figure 2 – Template to collect technical resources from partners.....	10
Figure 3 – The INDIGO datacloud PaaS orchestrator dashboard .....	14
Figure 4 – Supported transporters connect to a central message broker [15].....	19
Figure 5 – Incentives management component interactions.....	21
Figure 6 - ABAC Indicative information flow .....	25
Figure 7 - XACML Flow & Architectural components.....	27
Figure 8 – Usage of XML artefacts .....	28
Figure 9 - BALANA PDP (source: <a href="https://docs.wso2.com/display/IS510/XACML+Architecture">https://docs.wso2.com/display/IS510/XACML+Architecture</a> ) .....	30

## Executive Summary

This document is the first deliverable of WP3. It provides information about the work performed in tasks T3.1, T3.3, T3.4, and T3.6. As it covers various tasks, this document provides the information regarding a) cloud provisioning and utilization of cloud resources for PolicyCLOUD, b) the design of the cloud gateways and their APIs, c) the incentives management and identification in the scope of PolicyCLOUD, and d) the data privacy and governance mechanism. This document is the initial iteration of this report while two more iterations will be provided through the documents D3.4 (due on M20) and D3.7 (due on M32).

# 1 Introduction

This document is the first iteration of “Cloud Infrastructure Incentives Management and Data Governance: Design and Open Specification”, and is the first deliverable of WP3, covering tasks T3.1, T3.3, T3.4, and T3.6. In the scope of *T3.1 - Cloud Provisioning of the PolicyCLOUD Infrastructure*, focus was provided for the collection of the requirements and the planning of provisioning. In the scope of *T3.3 - Cloud Gateways & APIs for Efficient Data Utilization* and *T3.4 - Incentives Management*, the design of mechanisms of cloud gateways and Incentive management has been started respectively. Finally, for *T3.6 - Data Governance Model, Protection and Privacy Enforcement* the goal is the creation of both the model and the mechanism that is used for privacy enforcement and the protection of data. For all task this document provides the preliminary work performed until M8.

## 1.1 Structure of the document

The rest of the document is structured as follows. Section 2 cover the provisioning process for the cloud infrastructure of PolicyCLOUD, while Section 3 presents the Cloud Gateway components. Section 4 describes the identification and management of incentives. Section 5 presents the background and the technologies to be used for the creation of the data privacy mechanism. Finally, Section 6 provides the conclusion of the document, while Section 0 adds the document references. The document concludes with Appendix I - Access the EGI User Community and Appendix II - Access the INDIGO PaaS Orchestrator that cover the usage of the EGI platform and the INDIGO PaaS orchestrator correspondingly.

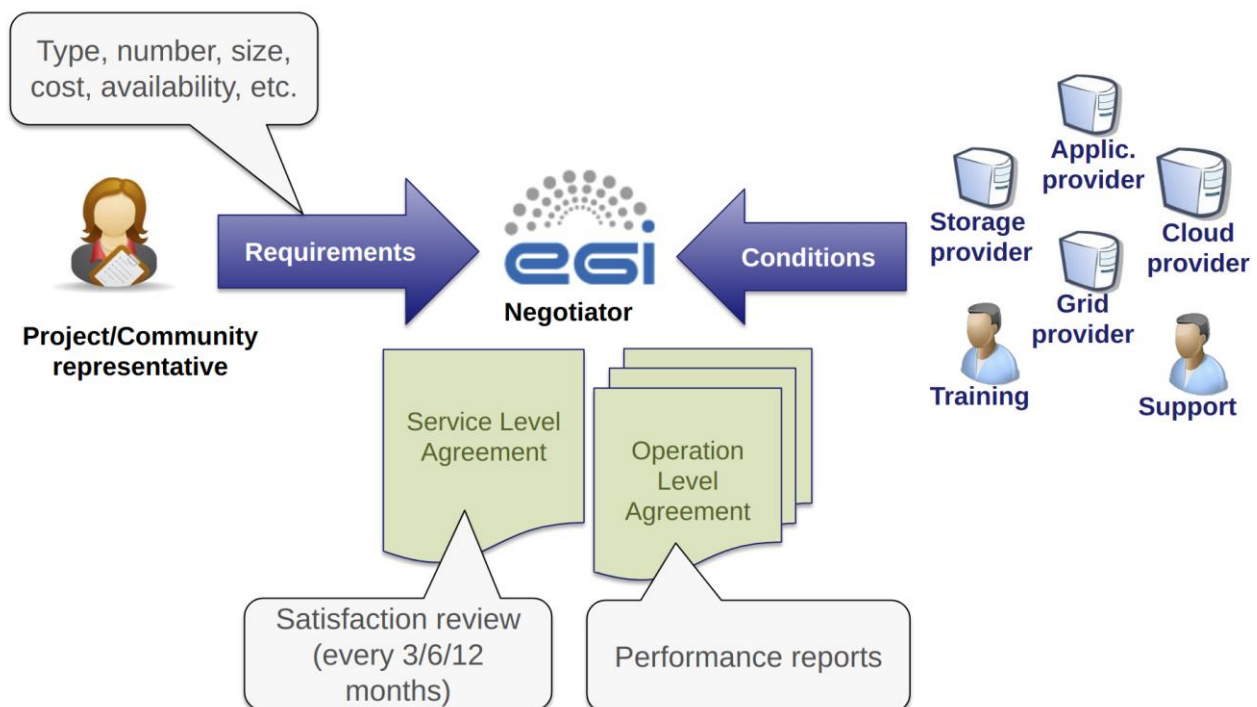
## 2 Cloud Provisioning of the PolicyCLOUD Infrastructure

In the context of the PolicyCLOUD project EGI is provisioning the needed computing and storage capacities to set-up the cloud-based infrastructure to serve the project needs. Overall, this cloud infrastructure will help policy makers, public authorities and different stakeholders, to analyse a wide plethora of datasets from different data sources, and facilitate policy making.

EGI's offering for the project includes a federated IaaS cloud to run compute- or data-intensive tasks and host online services in virtual machines or Docker containers on IT resources accessible via a uniform interface. To facilitate the cloud provisioning of resources for the project, the framework described in section 2.1 will be adopted.

### 2.1 The EGI SLA/OLA framework

To facilitate the allocation of resources to address the needs of scientific communities, Research Infrastructures and projects, EGI has established the framework depicted in Figure 1 .



**FIGURE 1 – THE RESOURCES ALLOCATION PROCESS IN EGI**

The process starts with collection of technical requirements expressed in terms of number of CPU cores, disk space, software packages, etc., needed by the Customer (e.g.: scientific community, Research Infrastructure, projects). EGI uses these requirements to collect Expressions of Interests (EoIs) from the different service providers of the Federation that are fit-for-purpose. The process requires one-on-one interactions between the Customer and the service providers of the federation. The interaction with the different service providers, is conducted by EGI which

acts as facilitator. As a result of this interaction EGI will establish Operational Level Agreements (OLAs) with the interested service providers, and single Service Level Agreement (SLA) with the Customer.

Overall, the main objective of this framework is to create a reliable, trust-based communication channel between the Customer and the Providers to agree on the services, their levels and types of support. From a technical perspective, these agreements state the type of service offered by the service providers (e.g. Cloud Compute, Cloud Container Compute, Online Storage, etc.), the conditions under which this service is offered along with the service targets (e.g. service availability and reliability).

To guarantee high quality of service(s) offered to the Customer, EGI coordinates and monitors the service delivery in order to measure the fulfilment of the agreed service level targets and manages the Customer complaints and disputes. From a technical perspective, this is done by monitoring the monthly service performance and checking whether the service targets are met. Every three, six and twelve months EGI runs a Customer Satisfaction Review process to review the whole agreement and identify possible improvements for the agreement and services. It is the Customer's responsibility to acknowledge EGI and the providers in the scientific publications benefiting from the service offered.

## 2.2 Collection of the technical requirements

To facilitate the provision of computing and storage resources in PolicyCLOUD a dedicated budget of 50K€ was allocated, as other direct costs, to EGI during the proposal preparation. This budget will be used by EGI to sign-up sub-contracts with all the cloud providers interested to contribute allocating resources for supporting pilots and partners' needs following the resources allocation process described in section 2.1.

To collect all the technical needs EGI has invited project members to report technical needs in a shared spreadsheet. This spreadsheet, which was introduced during one of the PolicyCLOUD Technical meetings organized by the Technical Coordinator, is shown in Figure 2.

A	B	C	D	E	F	G	H
Pilots			Pilot 1	Pilot 2	Pilot 3	Pilot 4	ATOS
Resource Centre							
Contact email							
	Total # VMs	0					
	Total # CPU cores	0					
Resource	Total # RAM (GB)	0					
Summary	Total # local storage	0					
Resource details							
	Online storage (guaranteed) (GB)	0					
	Online storage (opportunistic) (GB)	0					
	Public IP addresses	0					
Allocation type							
Payment mode offer							
Service Cost							
Other technical requirements							
	Start date						
Duration	End date						
Supported VOs							
VO ID card							

FIGURE 2 – TEMPLATE TO COLLECT TECHNICAL RESOURCES FROM PARTNERS

The template is composed of several parts. In green are highlighted the information that project members were asked to fill in, while in gray the ones that EGI will take care with the cloud providers. Some clarifications about the template are provided in the following sections.

### 2.2.1 Resource Centre

This section specifies the hostname of the cloud provider in the EGI Federation and its contact email address to be used in case of communications related to the service in the scope of this Agreement.

### 2.2.2 Resource summary

Amount of resources requested by the partners for supporting the pilots.

### 2.2.3 Allocation Types

There are three possible allocation types:

- **Pledged** – Resources are exclusively reserved to the Community and the job will be executed immediately after submission.
- **Opportunistic** – Resources are not exclusively allocated, but subject to local availability
- **Time allocation** – Resources are available in fair share-like mode for a fixed time period.

**For the PolicyCLOUD project, pledged resources will be made available.**

### 2.2.4 Payment Models Offer

There are two possible payment model offers:

- **Sponsored** – Resources are funded, or co-funded, by the European Commission or government grants.
- **Pay-for-use** – Resources for an agreed amount and fixed time period are paid for by the customer.

For the PolicyCLOUD project, resources will be made available with a pay-per-use payment model offer.

### 2.2.5 Service Cost

Costs to access the resources provided by the EGI Cloud providers described in section 2.2.1.

### 2.2.6 Other Technical Requirements

Additional technical requirements requested by the customer (e.g.: ports to be opened, OS release, etc.).

### 2.2.7 Duration (of the agreement)

The validity of the EGI VO SLA agreement with the customer. This duration is defined by the First and Last day of service delivery.

### 2.2.8 Supported Vos

Users of the EGI Federation are organised into Virtual Organisations (VO)<sup>1</sup>. A typical VO provides some base-level access to the resources for all VO members, with some members having elevated privileges. For the PolicyCLOUD project, the procedure described in this document<sup>2</sup> was used to create a dedicated VO for the project. The VO was enabled by the EGI cloud provider selected during the open call (see Section 2.3) to provide the requested resources.

**For the PolicyCLOUD project, the vo.policycloud.eu<sup>3</sup> VO was created in the EGI Operations Portal.**

## 2.2.9 VO Id card

Technical details about the VOs in the EGI Operations Portal are described in the VO Id cards.

## 2.2.10 Service Level Targets

Two possible Service Level Targets are available:

- **Monthly Availability:** Defined as the ability of a service or service component to fulfil its intended function at a specific time or over a calendar month.
  - Minimum (as a percentage per month): 85%
- **Monthly Reliability:** Defined as the ability of a service or service component to fulfil its intended function at a specific time or over a calendar month, excluding scheduled maintenance periods.
  - Minimum (as a percentage per month): 90%

**For the PolicyCLOUD project, resources will be offered with the following Service Level Targets:**

- Monthly Availability: 90%
- Monthly Reliability: 95%

## 2.2.11 Incidents Handling

Incidents will be handled according to the Quality of Support level that is estimated according to the impact of the outage or service quality degradation. There are three different QoS levels, each defining different response times for given Ticket Priority.

- Base
- Medium
- Advanced

**For the PolicyCLOUD project, the Quality of Support provided in the EGI Agreements has “Medium” level.** It means that different incident priorities will be processed taking into account the response time reported in Table 1.

Incident priority	Response time
Less urgent	5 working days

<sup>1</sup> A VO is a group of people that have similar focus in their work and have a common wish to share access to a subset of EGI resources.

<sup>2</sup> [https://wiki.egi.eu/wiki/PROC14\\_VO\\_Registration](https://wiki.egi.eu/wiki/PROC14_VO_Registration)

<sup>3</sup> <https://operations-portal.egi.eu/vo/view/voname/vo.policycloud.eu>

Incident priority	Response time
<b>Urgent</b>	5 working days
<b>Very urgent</b>	1 working day
<b>Top priority</b>	1 working day

**TABLE 1 – INCIDENT PRIORITY VS. RESPONSE TIME**

## 2.2.12 Reporting and Violations

Performance reports, describing how the involved cloud provider delivers the service, is sent to the PolicyCLOUD main contacts **every 6 months**. In case of any violations, EGI will contact the service provider to understand the problem and identify mitigation plans. In case of repeated violations of the service target for 3 consecutive months, the whole agreement with the provider will be reviewed.

## 2.3 The Call for Cloud Providers

After the 1st. General Assembly meeting [1], EGI sent the first call for cloud providers that, with a pay-per-use model, will be interested to support the project initiative and allocate the requested pledged resources. With this call, cloud providers of the EGI Federation were invited to provide the costs to set-up a shared and scalable Kubernetes cluster during the period (M06-M30), and further scale-up the capacity of the cluster with additional resources during the second part of the project (M31-M36).

Two Expression of Interests (EoIs) to provision up to 80 vCPU cores and 160GB of RAM each, and up to 2TB of block storage during the period (M05-M30) were collected by two providers of the EGI Federation. Both providers also expressed the interest to scale-up the initial amount of capacity during the second part of the project (M31-M36). The costs of the two cloud providers to provision the requested cloud capacities reported in Table 2.

#	Cloud Provider	Country	Costs	Additional comments
<b>1</b>	100%IT [2]	United Kingdom (UK)	- VM Specification, 4vCPU, 8GB RAM, 50GB disk: £30.75 per month - Block Storage, 2TB, £71.68 per month - Total cost £686.68  Total costs £17,167 (from M05-M30)  Received on: 12/05/2020	100%IT does not currently offer containers as a service but they have it on the roadmap to provide this as a service in the future.
<b>2</b>	RECAS-BARI [3]	Italy (IT)	Overall price (€ 25,000) for period (M06-M36) includes costs to allocate: 68 vCPU cores, 304GB of RAM, 2TB of block storage.  Received on: 15/05/2020 Updated on: 03/06/2020	The PaaS k8s will be available on the same resources, without additional costs. Technical support will also be offered with 5K euros of additional costs.

**TABLE 2 – COSTS FOR THE CLOUD PROVISIONING FOR PERIOD M05-M30**

EGI regularly reported the status of the cloud provisioning during the weekly Technical Coordination meetings. During these meetings project members have been invited to comment on the received EoIs. Among the two EoIs, the second one from RECAS-BARI was considered very interesting from a technical perspective. As a follow-up activity, the cloud provider was invited to introduce the high-level architecture of the INDIGO-DataCloud PaaS

Orchestrator during the PolicyCLOUD Technical Coordination meeting that took place on June 09 at 15:00 CEST. During the technical meeting Marica Antonacci and Giacinto Donvito from INFN have introduced the technical architecture of the INDIGO PaaS orchestrator. After the initial introduction, with a live demo, Marica showcased how the INDIGO PaaS Orchestrator can be used in a real scenario to deploy a distributed Kubernetes cluster in the Cloud infrastructure provided by EGI and target the needs of the project.

One of the main features of the INDIGO PaaS Orchestrator is that it supports federated authentication mechanisms and it is already connected with the AAI Check-In service [4] provided by EGI. Through the EGI check-In service, members of the PolicyCLOUD project are able to authenticate with the credentials provided by their Home Organisation (e.g. via eduGAIN [5] ), as well as using social identity providers, or other selected external identity providers. A snapshot of INDIGO PaaS Orchestrator is provided in Figure 3, while the procedure to access EGI User Community and activate an account in the INDIGO-DataCloud PaaS Orchestrator is documented in Appendix I - Access the EGI User Community and Appendix II - Access the INDIGO PaaS Orchestrator.

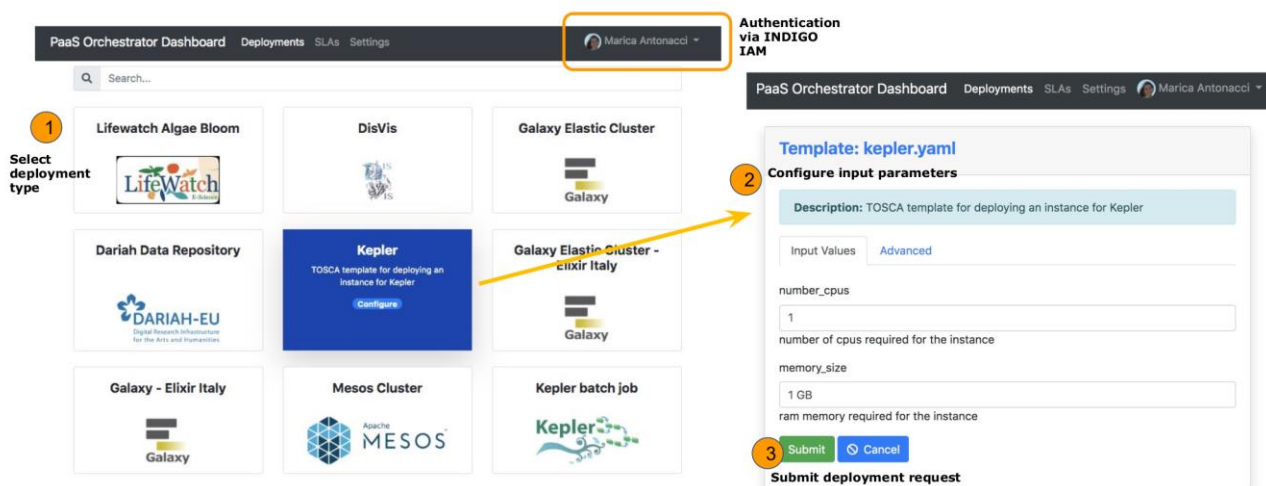


FIGURE 3 – THE INDIGO DATACLOUD PAAS ORCHESTRATOR DASHBOARD

## 2.4 Next Steps

The status of the PolicyCLOUD infrastructure will be further updated in the upcoming deliverables (D3.4 and D3.7). In both deliverables, the total amount of the resources provisioned by the EGI cloud providers will be also presented for supporting the PolicyCLOUD pilot use cases.

After the technical meeting the cloud provider and the project members agreed to have access to 10% of the total capacity (e.g.: 16 vCPU cores and 32GB of RAM) requested for a 1-month trial with no additional costs.

## 3 Cloud Gateways & APIs for Efficient Data Utilization

The Cloud Gateway component is the single point of connection of the cloud to the outside world. It works like an enhanced reverse proxy with more advanced capabilities including orchestration, security, monitoring etc. Internally it provides a simple and effective way to route incoming requests. Following the Gateway Pattern [6], it exposes a set of microservices to users through a unified API instead of providing an API for each service that includes. A few well-known API Gateway implementations includes Netflix Zuul [7], The Cross-Cloud API Management Platform (Apigee) [8], Amazon Cloud Gateway [9] and the Spring Cloud Gateway [10].

### 3.1 Cloud Gateway Capabilities

#### 3.1.1 Fetching external Resources

The Cloud Gateway will be a component that combines and integrates various microservices that will obtain data from heterogeneous data sources such as:

- External APIs (Twitter, Facebook, Reddit etc.)
- Files (Excel, CSV)
- Data streams from IoT devices and sensors
- Databases (SQL, NoSQL)

Each of the above services must be carefully configured in order to comply with the terms of usage for each source in order to ensure that the data stream will be not interrupted due to rules violation, such as overloading an external API despite the rate limit per request as it is declared in the documentation.

#### 3.1.2 Authentication and Security

The Gateway authentication and security mechanisms ensure that only authorized requests can get access to services and data.

#### 3.1.3 Data filtering

Cloud Gateway component will provide mechanisms that will be responsible for checking the reliability of the provided data by performing filtering of the obtained datasets before pushing them to the internal cloud's services. To this end, inaccurate records or corrupt data must be removed from the datasets and incomplete data must be identified before storing inside the cloud's infrastructure. Moreover, fetched data must be evaluated according to corresponding privacy levels. On top of this, sensitive and private data such as personal information must not be stored, in order to be compliant with GDPR regulations and laws for the privacy of data.

#### 3.1.4 Monitoring

Monitoring of the system by tracking meaningful statistics give an accurate view of the system's performance, availability, and overall health. To this end, metrics, events and metadata are being collected, in order to generate insights via graphical interfaces, reports, and alerts.

## 3.2 Offering a well-structured and well documented API

An essential part of the Cloud Gateway component will be the API functionality that will allow the interaction with the Gateway's functions and procedures. The need for having a programming interface, that will allow the effective interaction of the services inside PolicyCLOUD platform with the Gateway, will be achieved through the utilization of the API functionality. Using a REST API is considered a good practice and is a very common way for web service communication. Therefore, the design of the REST API has a great impact on the security, performance, and ease of use for API consumers. A well maintained, futureproof design for the REST API is surely required for having a reliable and fault tolerant Gateway component. Some of the best practices that will be utilized during the development of REST API include [11]:

**JSON for HTTP requests and responses:** Using a REST API means both request payload and responses should be in JSON format as it is the standard for transferring data. Although, XML format is an alternative selection, it is not widely supported by all frameworks. Furthermore, data in JSON format can easily be manipulated especially in NodeJS frameworks without the need to transform data before start processing. Moreover, a wide range of NoSQL databases like MongoDB save data in this format.

**Proper naming conventions:** It is very important for a REST API to have a strong and consistent naming convention strategy. To this end, the usage and understanding of a well and properly named API is easier. In the opposite case, a poor naming strategy can lead to confusion and mis usage of the API from its own users.

**Filtering and Pagination:** Datasets obtained by external sources are expended to be very large. In order not to slow down the overall system, data in REST API must be paginated. Filtering and pagination can increase performance and reduce resources usage for longer time periods.

**Caching:** Caching can be applied to data so to be retrieved from local memory instead of repeating same queries over and over. Caching in different levels of the Gateway (ex. Application level using Redis) can help significantly reducing resources usage and allow for the Gateway component to work faster and more efficiently. On top of this, proper configuration of caching mechanisms is important for PolicyCLOUD's Gateway component especially in occasions that there is a demand of the most updated data results like data streams etc. On the other hand, improper caching configuration can cause other services to get false data as input, leading to inaccurate results.

**Versioning:** Following a versioning system is important for both system's maintainability and for users. The most popular versioning system is Semantic Versioning [12] following the MAJOR.MINOR.PATCH pattern.

1. MAJOR version for backwards incompatible changes,
2. MINOR version for newly added backwards compatible functionality to the API,
3. PATCH version for backwards compatible patches and bug fixes,

Through the usage of semantic versioning Cloud Gateway component is able to handle what in programming world is referred as "dependency hell", meaning the inability to further extend the system because application uses many shared libraries and these libraries may depend on another library causing compatibility problems to the whole system, adding on complexity and causing the frustration of the users.

**Error Handling:** When a system is operational, errors occur. Therefore, in order to eliminate the possibility for an error to bring the whole system down, it is needed to gracefully handle errors and return well defined responses that indicated the kind of error that occurred. This allows maintainers to better understand and fix possible bugs. Common error HTTP status codes are:

- 400 Bad Request
- 401 Unauthorized Request – Only authorized users have access to this resource
- 403 Forbidden – Access to that resource is forbidden
- 404 Not Found – Resource not found
- 500 Internal server error – Generic error indicated some problem with the server
- 502 Bad Gateway – Invalid server response
- 503 Service Unavailable – Error occurred in server

**Security Practices:** Security should be an important part of the API's development. In monolithic applications security is being easily centralized with one interceptor that intercepts between calls. But, in a microservices environment this functionality can be very challenging. Moreover, since RESTful API is stateless, there is the need to implement a way of authentication/authorization that does not depend on techniques like sessions or cookies. JSON Web Token (JWT) [13] on the other hand is compact and self-contained way for securely exchange information between parties in JSON format. To this end, JWTs can be encrypted to provide a proper layer of secrecy in communication. Furthermore, JWT offers signed tokens verifying the integrity of contained data. A basic request implementing JWT consists of 3 basic parts:

- **Header:** Contains the token and the hashing algorithm,
- **Payload:** Consists of information about the authenticated entity along with metadata.
- **Signature:** Used to check the authenticity of the JWT and it is generated by hashing Header and Payload together along with a secret key.

During the design of the security practice, performance and scalability will also get into consideration. On top of this, JWT includes a cryptographic operation to validate the token. To this end, caching could help significantly reducing the time and resource impact of a repetitive token validation.

## 3.3 Technologies & Methodologies

The PolicyCLOUD Gateway will be designed and implemented using microservices architecture methodology. Every part of the Gateway will be wrapped in its own service, executing its own process. This architectural methodology allows building highly maintainable systems that are independent and loosely coupled, easily testable and deployable.

### 3.3.1 Services

PolicyCLOUD Gateway's microservices will utilize MoleculerJS [14], a framework built on top of NodeJS that is oriented in microservices development. NodeJS has been developed based on V8 runtime engine and can be very efficient for input-output (IO) heavy tasks, in contrast with CPU bound tasks like calculations, which have high demand on resources. To this end, NodeJS is an excellent choice for IO-bound tasks that consist the Gateways main functionality. On top of this, along with a very large community, many innovative enterprises add NodeJS to their main technology stack as it increases productivity and offers high performance at a much lower cost.

### 3.3.2 Dynamic Service Discovery

In a microservices environment the running instances of services dynamically change location inside networks. In order for client or services to make requests to a service it must use a service-discovery mechanism. MolecuJerJS has a built-in module to handle local service discovery but also supports integration with in-memory data stores like Redis for both local and remote service discovery. Using Redis-based discovery, a Redis server is used for keeping track of available nodes. MolecuJerJS nodes (running services) periodically publish their status (heartbeat) and fetch info from Redis updating their internal service registry. Redis key expiration mechanism removes nodes that do not publish heartbeat packets for a specified period. This keeps the service registry updated and allows MolecuJerJS nodes to detect disconnected nodes.

### 3.3.3 Load Balancing

To ensure that the Gateway will not fail in production, multiple nodes of the same services must be deployed when needed. When a service is called the request must be routed to the instance that has the lowest load at that moment. MolecuJerJS has built-in strategies for load balancing. Round-robin algorithm is very simple yet very effective algorithm that finds the most appropriate node to send the traffic by using an appropriate statistical model. More straightforward strategies can be used based on CPU-usage or latency of every node.

### 3.3.4 Fault Tolerance

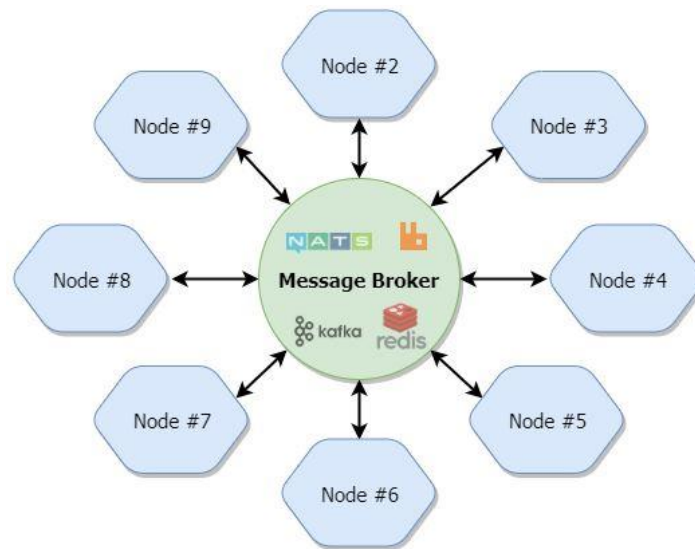
Microservices need to be designed so that they are fault tolerant meaning that if an error occurs it need to be handled gracefully and the service should return an error message. To this end, there are a few patterns that ensure service resiliency. Using the Circuit Breaker pattern, faults, that require a large amount of time to recover from, can be handled. Failing on fetching external APIs can be a time and resource consuming task. The Circuit Breaker pattern prevent repeatedly trying to execute operations that have high probability to fail. It also allows to detect whether the fault has been resolved and try to invoke the previously faulty operation. Moreover, it is a good practice to set timeout for service calling. On top of this, the utilization of MolecuJerJS for building services inside the Gateway and for the configuration of different fault tolerance strategies is very straightforward and easy to implement.

### 3.3.5 Caching

Through the usage of MolecuJerJS framework, for the implementation of various microservices, out-of-the-box caching solutions are offered. Redis caching is utilized using the ioredis NodeJS library. In cases that multiple nodes are needed for one service, Redis distributed cache module will allow sharing cached data across these instances. Moreover, this service coupled with options to use serializers, like MsgPack, allows store cached data faster and requiring less memory than a simple JSON serializer.

### 3.3.6 Messaging and Message Brokers

Services need to communicate with each other and often to collaborate to handle requests. Synchronous communication requests both services participating in the request to be available for the duration of the request usually over HTTP over REST. For inter-service communication, an asynchronous approach that allow exchanging messaging over specific channels. MolecuJerJS allows for the nodes to communicate with each other by using from a large variety of protocols and message brokers like NATS, Redis, MQTT, AMQP and Kafka. Kafka is suitable especially for occasions of large data streams and is remarkably fault tolerant by supporting “replay” scenarios.



**FIGURE 4 – SUPPORTED TRANSPORTERS CONNECT TO A CENTRAL MESSAGE BROKER [15]**

### 3.3.7 API Gateway

Services can be easily published as RESTful APIs using the *molecular-web*, the official API gateway for *MolecularJS* framework allowing routes whitelisting, throttling, mapping, authorization/authentication and many other features a modern API should be able to support.

### 3.3.8 Data Transformation and Database Adapters

*MolecularJS* allows connectivity with both SQL and NoSQL databases. More specifically, it includes official adapters for MongoDB, PostgreSQL, SQLite, MySQL, MSSQL. Data Manipulation is also possible by using action hooks before sending data to storage services inside *PolicyCLOUD* for removing sensitive or unnecessary info.

## 3.4 Next Steps

### 3.4.1 Detailed system requirement analysis for the Gateway's services

During this process, a more detailed description of system services and operational constraints such as how the system will be used. The external services that will be used for information gathering and the requested output from the Gateway's services.

### 3.4.2 Gateway's Architecture and Design

A detailed architecture for all services that will be implemented will be delivered. Moreover, communication between services including Dataflow diagrams that visualize data flow inside the Gateway will also be designed and described.

### 3.4.3 API Specification, Structure, and Documentation

Moreover, in the next steps will be delivered an OpenAPI specification for the Gateway's REST API, that describes the entire API functionality, including:

- Endpoints

- Authentication methods
- Parameters
- License, term of use of information

#### 3.4.4 Development and implementation

Development of all services and communication between services in a containerized environment.

#### 3.4.5 Testing and deployment

Evaluate system by checking requirements validation, system integration, security, performance, stress, usability, and user acceptance. This step also includes bug fixes that may occur.

## 4 Incentives Management

This section aims to introduce the foundations for the incentive management activity in PolicyCLOUD. Based on the idea that including citizens as participants in the policy development and management process, would enhance the acceptance and confidence on the created policies. PolicyCLOUD will provide a set of tools to policy makers to manage the incentives, so to raise the consciousness and allow citizens to be part of an evidence-based policy making process.

The policy maker will be at the centre of the approach.

- She/he will receive input from the incentives' analysis provided by PolicyCLOUD, which might support her/him in the decision-making process.
- She/he will declare and manage the incentives, through the mechanisms provided by PolicyCLOUD, which will motivate the participation of the citizens.
- She/he will interact with the involved participants and will guide and analyse how PolicyCLOUD will benefit by the relevant crowdsourcing data and knowledge created by the participants.

In order to provide the maximum support to the policy maker, the aim of the incentives management is twofold: support the policy maker in the incentives identification and help the policy maker in the incentives management. Figure 5 depicts those interactions.

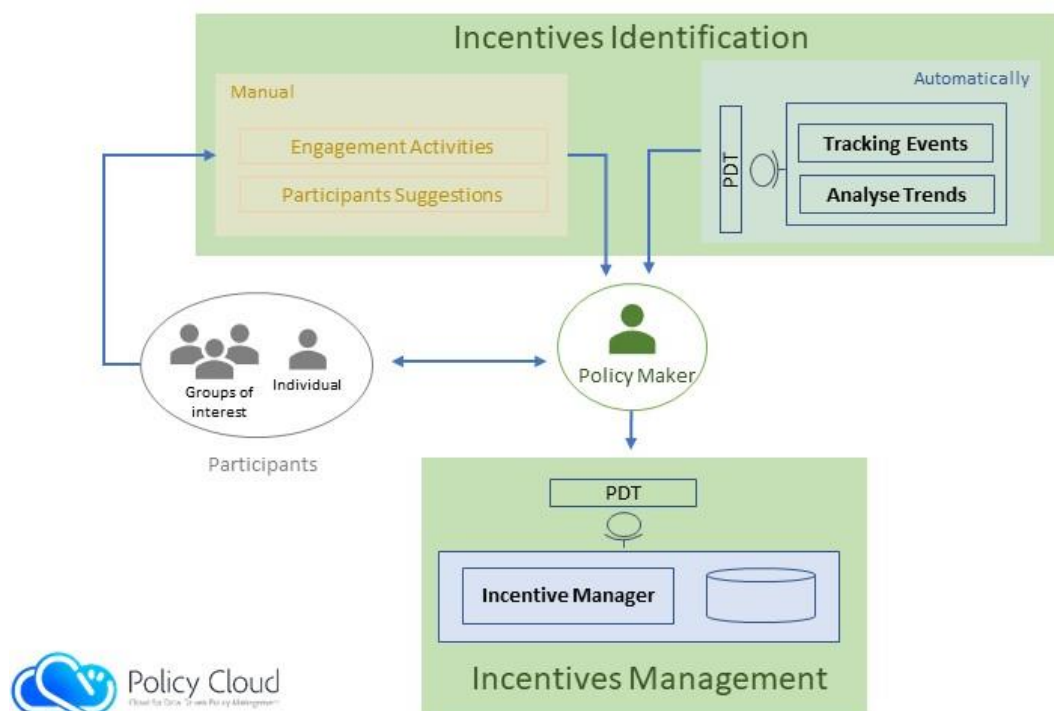


FIGURE 5 – INCENTIVES MANAGEMENT COMPONENT INTERACTIONS

On one hand, PolicyCLOUD will explore the *Incentives Identification* through two lines:

- **Manually:** Some initiatives might be in place for participants involvement, such as: living labs, or protocols to allow participants to declare their own interests.

- **Automatically:** Based on the data-driven facet of PolicyCLOUD, a set of model and analytical tools for the identification of trends, interests and events, which could motivate the participation of citizens, will be explored by PolicyCLOUD. Thanks to that, the policy maker will obtain knowledge, information and data that could be converted into incentives or even policy requirements.

On the other hand, PolicyCLOUD will offer to the policy maker the *Incentives Management* block, a set of mechanisms to declare and manage incentives and related actions for citizen involvement. At the same time, the incentives management block will support the policy maker in the ingestion of the results from this participation, acting that as a kind of log compiling content related to citizens contributions.

## 4.1 Next Steps

The specific ways in which citizens will participate will depend on the particularities of each use case. Due to the fact that at this stage the main effort of the requirements of the use cases is focused on the analytical tools, the details of the specification of the incentives management component are left open to be discussed in next versions.

## 5 Data Governance Model, Protection and Privacy Enforcement

The data governance model and the accompanying tools will offer protection and privacy enforcement for the data and will ensure that decisions across the complete path follow specific guidelines and legislations. Although the data governance model is not finalized yet, we provide its first version and we also cover relevant efforts on models, standards and frameworks.

### 5.1 Data Protection and Access Control

One of the core aspects of the protection and privacy mechanisms is to provide logical access control to generated data. It is important to clarify at this point the difference between authorization and authentication. Authentication deals with the problem of proving the identity of a natural person or a system entity that aims to interact with a resource while authorization deals with the problem of providing logical access control to these resources. The logical access control is always bound to specific actions that the natural-person/systemic entity (hereinafter subject) wishes to perform i.e. discovering, reading, creating, editing, deleting, and executing.

The “problem statement” of an authorization request entails some concepts that are common in all approaches. These include: the **subject** (i.e. requestor), the **object** (i.e. resource), the **action** (to the resource), the environmental context, the defined policy (or policies) and the policy-evaluation business logic (i.e. Policy Enforcement).

At the same time the **Responsibility assignment matrix (RACI)** model is an important tool for clarifying and defining roles and responsibilities in cross-functional or departmental projects or business processes, and it builds on four key responsibilities most typically used. The responsibilities used are the following;

- **Responsible** refers to the people who do the work to complete the task.
- **Accountable** is a single person specified answerable for the correct and thorough completion of the deliverable or task and the one that approves the work that responsible provides.
- **Consulted** refers to people whose opinions are taken into account and with whom there is a two-way communication
- **Informed** refers to people that are kept up-to-date on progress, often only on completion of the task or deliverable; and with whom there is just one-way communication.

RACI model defines roles and people: a role is a descriptor of an associated set of tasks. A role may be performed by many people and one person can perform many roles.

For the PolicyCLOUD Data Governance Model, we use RACI to model the access of specific stakeholders to specific data at specific points in the lifecycle.

Access Control Mechanisms are mechanisms realizing various logical access control models that provide the framework and a set of boundary conditions upon which the objects, subjects, operations, and rules may be combined to generate and enforce an access control decision.

There are several models and mechanisms, with each having its own advantages and limitations. Although already mentioned in D2.2 [16], for the sake of completeness the most dominant ACMs will be listed below:

- **Discretionary Access Control (DAC)** where the owner of the object specifies which subjects can access the object. Most operating systems such as Windows, Linux, Macintosh and most flavors of Unix are based on DAC models.
- **Mandatory Access Control (MAC)** where the system (and not the users) specifies which subjects can access specific data objects. The MAC model is based on security labels. Subjects are given a security clearance (secret, top-secret, confidential, etc.), and data objects are given a security classification (secret, top-secret, confidential, etc.).
- **Identity Based Access Control (IBAC)** uses mechanisms such as access control lists (ACLs) to capture the identities of those allowed to access an object. In the IBAC model, the authorization decisions are made statically prior to any specific access request and result in the subject being added to the ACL.
- **Role-based access control (RBAC)** employs pre-defined roles that carry a specific set of privileges associated with them and to which subjects are assigned.
- **Attribute-based access control (ABAC)** uses attributes, and policies that express boolean rule sets that can evaluate many different attributes before allowing access. ABAC, therefore, avoids the need for capabilities (operation/object pairs) to be directly assigned to subject requesters or to their roles or groups before the request is made. IBAC and RBAC can be seen as special cases of ABAC, with IBAC using the attribute of “identity” and RBAC using the attribute of “role”.

The adaptability and expressiveness of **ABAC makes it ideal for protecting the data in the lifecycle of PolicyCLOUD**. In more details, ABAC allows dynamicity in policy creation and also separation of concerns among policy definition and policy enforcement. Such policies shall be able to update dynamically based on specific constraints that are not a-priori related to requestors, in contrast to MAC scheme which is inappropriate since the policy definition point is unique and the resources have static properties based on which allowance and disallowance is provided, or DAC schemes where the properties are totally static.

Furthermore, IBAC and Access Control Lists suffer from a severe drawback, since despite the fact that they support extensible attributes for subjects & objects, the exhaustive list of authorization decisions should be defined prior to any request. Hence, we pay a huge penalty for the offered dynamicity.

On the other hand, RBAC and ABAC can satisfy our needs. As already discussed, RBAC is a specialization of ABAC (according to which one subject attribute is addressed as Role) and as such we will adopt the generalized ABAC.

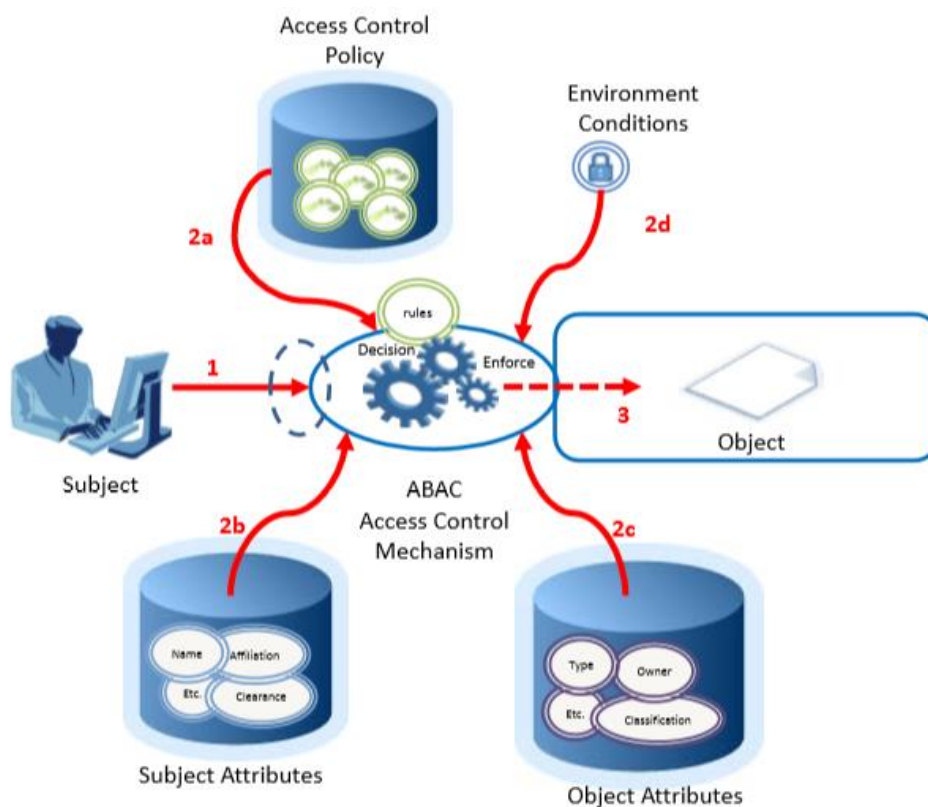
In general, using ABAC removes the need for capabilities (operation/object pairs) to be directly assigned to subject requesters or to their roles or groups before the request is made [17]. Instead, when a subject requests access, an ABAC-compliant engine makes an access control decision based on a) the assigned attributes of the requester, b) the assigned attributes of the object, c) the environment conditions, and d) a set of policies that are specified in terms of those attributes and conditions.

Following the strict definition of NIST [17] we will refer to ABAC as the “access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions”. Following this definition, the terms “attribute”, “subject”, “object”, “operation”, “policy” and “environmental condition” will be unambiguously used based on the following definitions:

- Attributes are characteristics of the subject, object, or environment conditions. Attributes contain information given by a name-value pair.

- A Subject is a human user or a systemic entity, such as a device that issues access requests to perform operations on objects. Subjects are assigned one or more attributes.
- An Object is a system resource for which access is managed by the ABAC system, such as devices, files, records, tables, processes, programs, networks, or domains containing or receiving information. It can be the resource or requested entity, as well as anything upon which an operation may be performed by a subject including data, applications, services, devices, and networks.
- An Operation is the execution of a function at the request of a subject upon an object. Operations include read, write, edit, delete, copy, execute, and modify.
- Policy is the representation of rules or relationships that makes it possible to determine if a requested access should be allowed, given the values of the attributes of the subject, object, and possibly environment conditions.
- Environment conditions represent operational or situational context in which access requests occur. Environment conditions are detectable environmental characteristics. Environment characteristics are independent of subject or object, and may include the current time, day of the week, location of a user, or the current threat level.

Any ABAC system should implement the conceptual flow that is depicted on Figure 6. According to this flow any subject can perform an access request for a specific operation regarding a specific medical record (step 1).



**FIGURE 6 - ABAC INDICATIVE INFORMATION FLOW**

The access request is handled by the ABAC reference implementation engine, which consults a policy repository (step 2a) to derive the set of attributes that have to be examined in order to reach a decision of “allow” or “deny”. The attribute examination phase checks subject attributes (step 2b), object attributes (step 2c) and environmental attributes (step 2d) in order to perform the actual assessment (step 3).

It should be clarified that ABAC is a theoretical framework and not a standard. In the next section, we will elaborate on which standard will be used for the Policy Cloud Protection and Privacy engine.

### 5.1.1 ABAC Implementations

The key standards that implement ABAC are OASIS standard of extensible Access Control Markup Language (XACML) and the Next Generation Access Control (a.k.a. NGAC) standard [18]. These two are considered to be the most notable ones.

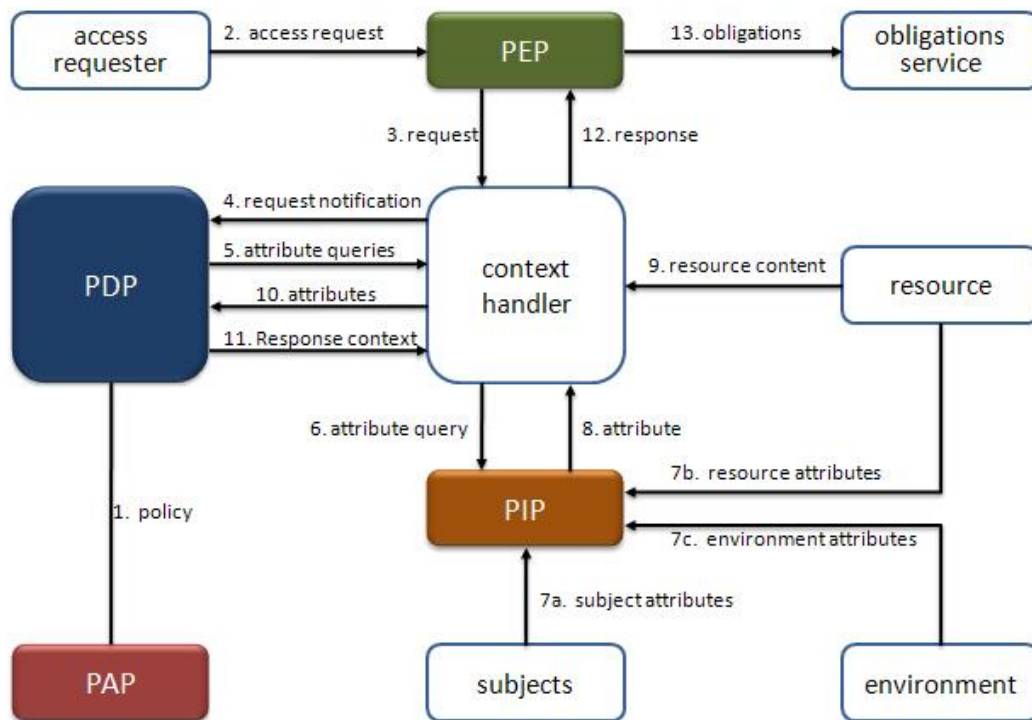
Finally, we also have to mention Abbreviated Language For Authorization (ALFA) [19]. ALFA is a pseudocode language that respects the XACML model (contains the same structural elements as XACML i.e. PolicySet, Policy, and Rule), but uses JSON instead of XML for the definition of access-control policies and maps directly into XACML.

#### 5.1.1.1 XACML

XACML is an OASIS [20] standard that describes both a policy language and an access control decision request/response language. Both languages use XSD [21] notations; hence policy definition and request/response elements are serialized as XML elements. The policy language details the general access control requirements and provides extension points for new functions, data types, combining logic, etc. In the other hand, the request/response language allows forming a query that asks whether a given action should be allowed. The response shall include an answer on the request allowance using one of the following four values:

- Permit
- Deny
- Indeterminate (an error occurred or some required value was missing, so a decision cannot be made)
- Not Applicable (the request can't be answered by this service).

The specification defines five main components (see Figure 7) that handle access decisions; namely Policy Enforcement Point (PEP), Policy Administration Point (PAP), Policy Decision Point (PDP), Policy Information Point (PIP), and a Context Handler.



**FIGURE 7 - XACML FLOW & ARCHITECTURAL COMPONENTS**

The functional purpose of the main components is:

- The Policy Administration Point (PAP) is the repository for the policies and provides the policies to the Policy Decision Point (PDP);
- The Policy Enforcement Point (PEP) is the interface of the whole environment to the outside world. It receives the access requests and evaluates them with the help of the other actors and permits or denies the access to the resource;
- Policy Decision Point (PDP) is the main decision point for the access requests. It collects all the necessary information from other actors and calculates a decision;
- Policy Information Point (PIP) is the point where the necessary attributes for the policy evaluation are retrieved from several external or internal actors. The attributes can be retrieved from the resource to be accessed, environment (e.g. time), subjects, and so forth.

As already mentioned, XACML uses XSD notation in order to model the three basic artefacts which are required i.e. the policy, the request and the response. Thus, as depicted on Figure 8, three types of XMLs are required by an XACML engine in order to judge upon a decision i.e. the Policy.xml which serializes an actual policy, the Request.xml which serializes an authorization request and the Response.xml that serializes the output of the engine.

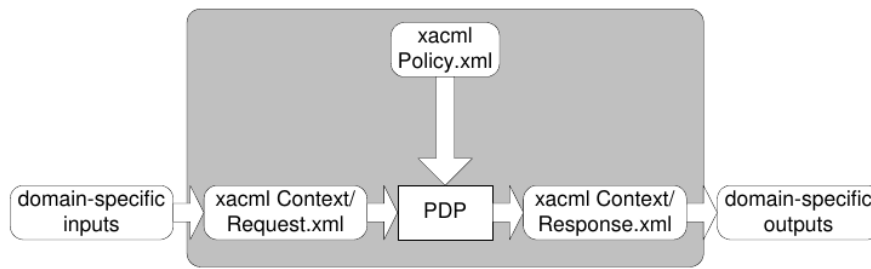


FIGURE 8 – USAGE OF XML ARTEFACTS

#### 5.1.1.2 NGAC

Next Generation Access Control [18] is a redefinition of access control in terms of a fundamental and reusable set of data abstractions and functions. NGAC provides a unifying framework capable of supporting, even without extension, not only many current access control approaches, but also novel types of policy that have been conceived but never implemented due to the lack of a suitable enforcement mechanism.

The set of NGAC standards specifies the architecture, functions, operations, and interfaces necessary to ensure interoperability between conforming NGAC implementations. It contains an abstract functional description of architecture to be implemented and also gathers the entities comprising the architecture on the basis of their function. Conforming implementations may employ any design technique that does not violate interoperability. NGAC's access control data is comprised of basic elements, containers, and configurable relations. NGAC uses the terms user, operation, and object with similar meanings. In addition to these, NGAC includes processes, administrative operations, and policy classes [22].

NGAC does not express policies through rules as XACML based ABAC, but instead used the following relation types:

- **assignments** (define membership in containers),
- **associations** (to derive privileges),
- **prohibitions** (to derive privilege exceptions),
- **obligations** (to dynamically alter access state).

XACML is similar to NGAC as they both provide flexible, mechanism-independent representations of policy rules that may vary in granularity, and they employ attributes in computing decisions. However, XACML and NGAC differ in the way they formulate policies and treat attributes, the way the requests are represented and the way that decisions are made.

Out of the two approaches (XACML and NGAC) we have selected XACML mainly because of its architectural clarity and our previous experience with it.

## 5.2 PolicyCLOUD Data Governance Model and Privacy Enforcement mechanism

Data Governance Model and Privacy Enforcement mechanism includes three different parts, a) the access policy editor, b) the model and model editor and c) the ABAC authorization engine. The access policy editor will provide the user with the ability to define and store policies based on the ABAC scheme according to the XACML standard. The data governance model of PolicyCLOUD will be used for the definition of these policies, and also for the actual enforcement of the policies by the authorization engine that will be able to evaluate the policies and the attributes, thus enforcing protection and privacy-preserving policies.

### 5.2.1 PolicyCLOUD Model

The Data Governance Model of PolicyCLOUD will ensure access of different entities to the corresponding datasets at specific phases of the data and the policy lifecycles.

For the model to be used for Data Governance in the scope of the PolicyCLOUD, we based our work on the existing context-aware security model of PaaSword EU project. PaaSword [23] is an XACML implementation that emphasizes in the **semantic evaluation** of attributes. It is very important for the scope of PolicyCLOUD that PaaSword provides the ability **to harmonize the attribute creation process through the usage of the extensible Context Model**. In traditional XACML the creation of attribute values and subject and resource assignments to those attributes is typically performed in disperse venues **without any notion of coordination or governance**. This is an inherent drawback of the XACML standard. The PaaSword engine alleviates this drawback through the usage of the developed and **extensible Context Model**.

PaaSword context aware security model is composed by five main classes. For the scope of PolicyCLOUD, we enriched the few classes of PaaSword model and more specifically its classes of Subject, Object and Connectivity. These classes which are described below, are subclasses of the class SecurityContextElement that refers to several contextual attributes that may be associated with the subject and/or the objects of a request, as well as with the request itself [24].

- Object: contains types of sensitive data;
- Subject: represents a requestor who intends to access the object content. The requestor can be a person, an organization, a group or a software;
- Location: tracks the exact location of a subject who requests access to data;
- DateTime: tracks the exact date and time of a subject who requests access to data;
- Connectivity: tracks the device type, the connection type, the connection security and the connection metrics.

### 5.2.2 PolicyCLOUD Policy Enforcement

For the implementation of the policy enforcement engine of PolicyCLOUD, Balana [25] is considered. Balana was the first open-source reference implementation of the XACML protocol and is a widely adopted solution. It supports the entire lifecycle of authorization processing and is tightly integrated into the WSO2 Identity Server [26]. Balana as XACML engine of the WSO2 Identity Server has two major components, the Policy Administration Point (PAP) and Policy Decision Point (PDP). Figure 9 presents the component architecture of the PDP that is our main interest.

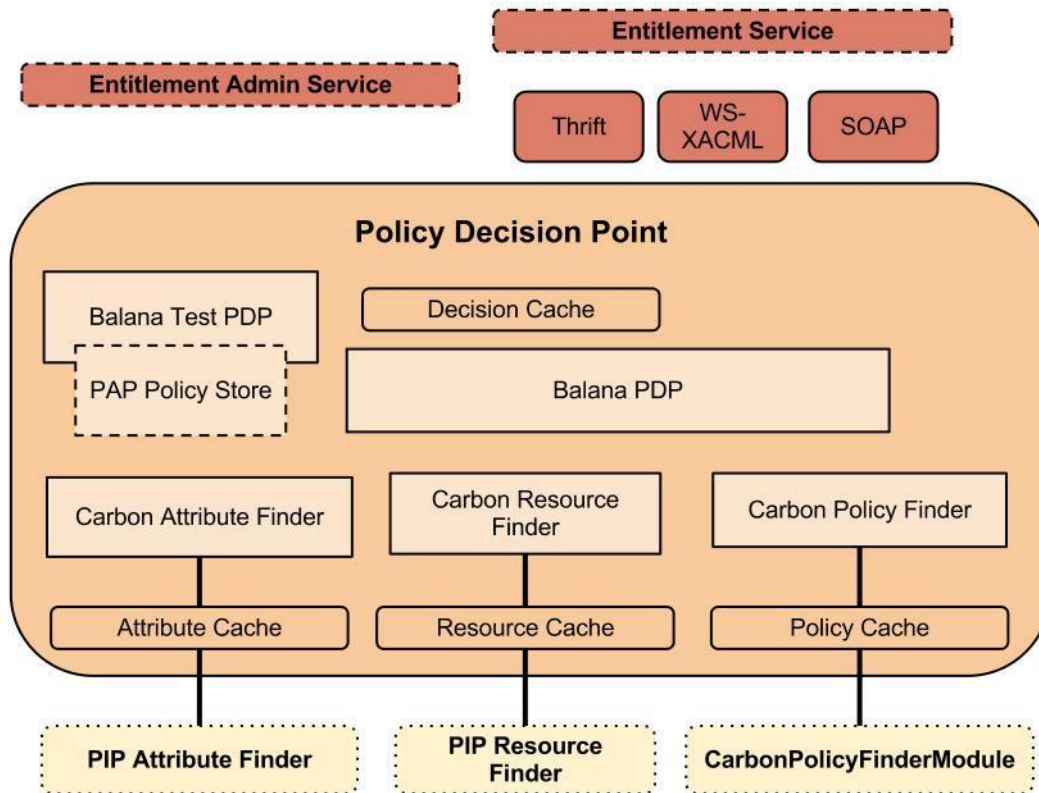


FIGURE 9 - BALANA PDP (SOURCE: [HTTPS://DOCS.WSO2.COM/DISPLAY/IS510/XACML+ARCHITECTURE](https://docs.wso2.com/display/IS510/XACML+ARCHITECTURE) )

In general, we consider Balana a highly extensive open source solution, suitable for the needs of PolicyCLOUD. It has to be mentioned that we consider using Balana PDP instead of PaaSword, due to the severe overhead in terms of time for semantic evaluating the XACML policies imposed by the ontological inferencing of rules in PaaSword.

## 5.3 Next Steps

In the next period the first version of the software prototype will be delivered and documented in D3.2. This means that the first version of the model and the policy enforcement mechanism will be implemented. For future releases of the platform we will also work on improving the mechanism with the inclusion of a user interface. Also, more detailed description of the mechanism and sample access control scenarios will be provided in the future iterations of this document (D3.4 and D3.7).

## 6 Conclusion and Next Steps

In this document we present the progress of the tasks T3.1, T3.3, T3.4, and T3.6 since M8 of the project. At first, we described the process for collecting the requirements and the planning of provisioning; EGI is managing the provisioning the needed computing and storage capacities to set-up the cloud-based infrastructure that will be used in PolicyCLOUD in order to analyse a plethora of datasets from different data sources, and facilitate policy making.

On this document we also reported the capabilities and mechanisms to be used by the components called cloud gateways and are responsible to obtain data from heterogenous data sources, and also provided the foundations for the incentives management activity in PolicyCLOUD. PolicyCLOUD will provide a set of tools to policy makers to identify and manage the incentives, so to raise the consciousness and allow citizens to be part of an evidence-based policy making process.

Finally, in section 5 we examined different models and technologies and presented how data protection and privacy enforcement can be offered by an ABAC based access control mechanism.

The first version of the software prototype of the services mentioned in this document will be provided by M10. This document is the initial iteration of this report while two more iterations will be provided through the documents D3.4 (due on M20) and D3.7 (due on M32). The status of the PolicyCLOUD infrastructure will be further updated in the upcoming deliverables (D3.4 and D3.7), with the total amount of the resources provisioned by the EGI cloud providers will be presented. Also, more detailed description of the cloud gateways and the data protection mechanism will be provided, including presenting the information about their implementation and usage. Finally, in next iterations of the document, the details of the specification of the incentives management component will be included.

## References

- [1] Agenda of the 1st PolicyCloud general assembly meeting, [https://policycloud.eu/sites/default/files/PolicyCLOUD\\_1st\\_GA\\_Agenda.pdf](https://policycloud.eu/sites/default/files/PolicyCLOUD_1st_GA_Agenda.pdf).
- [2] 100%it, <https://100percentit.com/>.
- [3] RecasBari, <https://www.recas-bari.it/index.php/en/>.
- [4] EGI Check-in Service, <https://www.egi.eu/services/check-in/>.
- [5] eduGAIN, <https://edugain.org/>.
- [6] API Gateway pattern: The API gateway pattern versus the Direct client-to-microservice communication, <https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/direct-client-to-microservice-com>, 2019.
- [7] Netflix Zuul: Documentation, <https://github.com/Netflix/zuul>.
- [8] Apigee, <https://cloud.google.com/apigee>.
- [9] Amazon API Gateway, <https://aws.amazon.com/api-gateway/>.
- [10] Spring Cloud Gateway, <https://spring.io/projects/spring-cloud-gateway>.
- [11] J. Au-Yeung, Best practices for REST API design, <https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/>, 2020.
- [12] Semantic Versioning, <https://semver.org/>.
- [13] JWT, <https://jwt.io/introduction/>.
- [14] MoleculerJS, <https://moleculer.services/docs/0.14/>.
- [15] MoleculerJS Networking, <https://moleculer.services/docs/0.14/networking>.
- [16] PolicyCLOUD, D2.2, Conceptual Model & Reference Architecture, 2020.
- [17] NIST, Guide to Attribute Based Access Control (ABAC) Definition and Considerations, <https://doi.org/10.6028/NIST.SP.800-162>, 2014.
- [18] NGAC standard (ANSI499), <https://webstore.ansi.org/standards/incits/incits4992018>.
- [19] Abbreviated Language for Authorization, <https://www.axiomatics.com/alfa/>.
- [20] OASIS Standards, <https://www.oasis-open.org/standards>.

- [21] W3C XML Schema Definition Language (XSD), <https://www.w3.org/TR/xmlschema11-1/>.
- [22] R. C. David Ferraiolo, A Comparison of Attribute Based Access Control.
- [23] PaaSword, <https://paasword.io/>.
- [24] Y. V. I. P. I. P. G. M. S. Veloudis, Context-aware Security Models for PaaS-enabled Access Control, Rome, Italy: 6th International Conference on Cloud Computing and Services Science (CLOSER 2016), April 23-25, 2016.
- [25] WS02 Balana, <https://github.com/wso2/balana>.
- [26] WS02 Identity Server, <https://wso2.com/identity-and-access-management/>.
- [27] EGI Identity select, [https://aai.egi.eu/registry/co\\_petitions/start/coef:100](https://aai.egi.eu/registry/co_petitions/start/coef:100).
- [28] INDIGO Cloud Login, <https://iam-test.indigo-datacloud.eu/login>.
- [29] INDIGO-DC PaaS Orchestrator, <https://indigo-paas.cloud.ba.infn.it>.
- [30] Apache Thrift, <https://thrift.apache.org/>.
- [31] ApacheDS, <https://directory.apache.org/apacheds/>.

## 7 Appendix I - Access the EGI User Community

The registration process to access the cloud infrastructure provided by EGI is based on two steps:

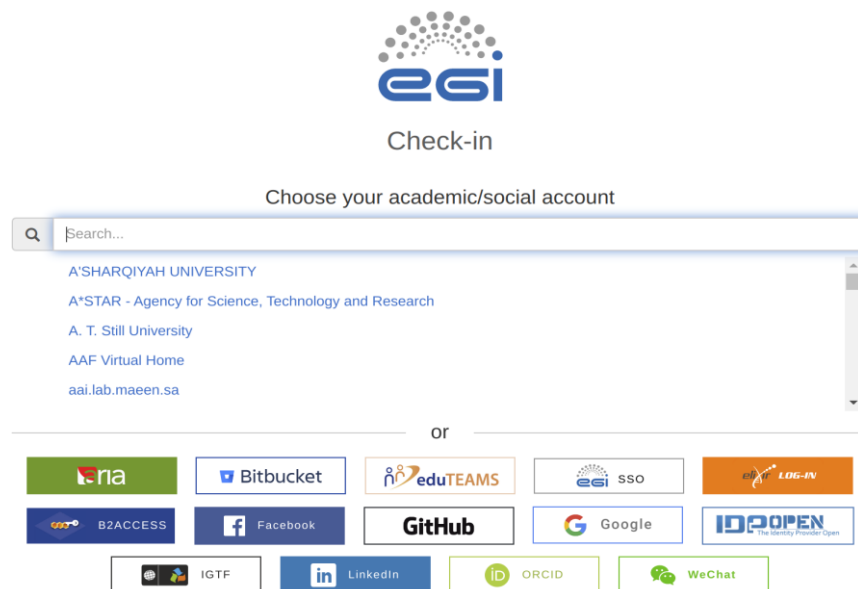
- Sign up the EGI User Community and join the vo.policycloud.eu organization.
- Validate and activate the registration to the EGI User Community.

### 7.1 Step 1. Sign up the EGI User Community

Before using any EGI services and resources, the user needs to sign up the EGI User Community. As part of this process, the user will be assigned a personal EGI ID which will then be used across all EGI services and resources.

To sign up the EGI User Community, and get your personal EGI ID account, please follow the instructions below:

- Enroll the vo.policycloud.eu organization [27] with the EGI Check-In [4] service.
- Users must select valid federated accounts from one of the available Identity Providers they belong to, or use one of the available social media providers such as: Facebook, Google, LinkedIn, ORCID, GitHub, etc. (Fig. 1).



**FIGURE 1 - SELECT THE PROPER IDENTITY PROVIDERS FROM THE DISCOVERY SERVICE**

The first time the user requires access to EGI services/resources, EGI Check-In will guide the user through a registration process and the submission of a web form.

## 7.2 Step 2. Activate the account

After the submission of the form, the user will be notified by the EGI Check-In service by email. This email contains a validation link to activate the account.

This is a typical email sent by the EGI Check-In service with the validation link to approve the account and join the EGI User Community (Fig. 2):

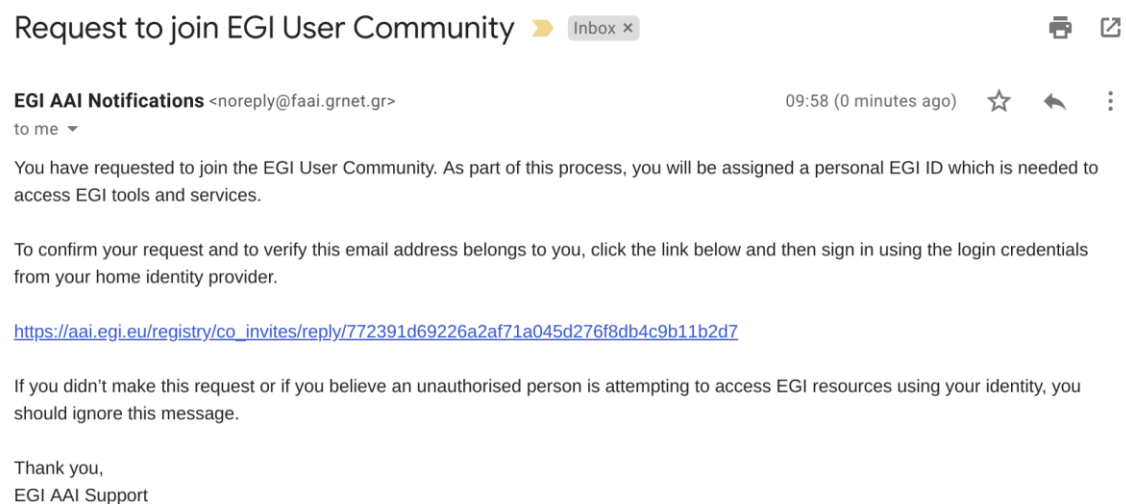


FIGURE 2 - NOTIFICATION EMAIL WITH THE LINK TO APPROVE THE REQUEST

As soon as the request is confirmed by the user and validated by the EGI User Community administrator, the user is officially a member of the EGI User Community and he/she can proceed requesting access to the INDIGO-DataCloud PaaS Orchestrator (Appendix II - Access the INDIGO PaaS Orchestrator).

## 8 Appendix II - Access the INDIGO PaaS Orchestrator

To activate an account in the INDIGO-DataCloud PaaS Orchestrator, please follow the steps described in this Appendix.

### 8.1 Step 1. Create a user's account

- Go to the INDIGO DataCloud IAM [28] dashboard.
- Users have to start the authentication process by clicking on the EGI Check-In service (Fig. 3) and select, from the discovery page, the Identity Provider they belong to, or one of the available social media providers.

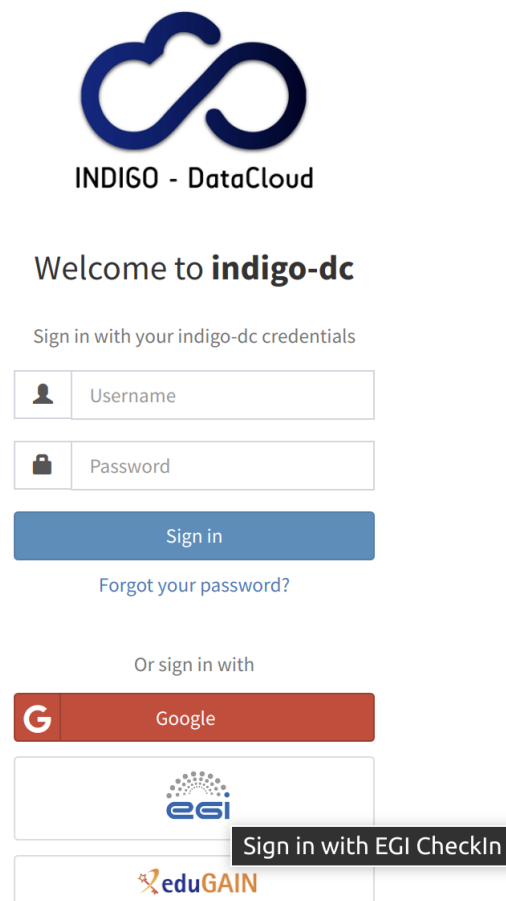


FIGURE 3 - LOGIN WITH FEDERATED CREDENTIALS

- Once the user has successfully authenticated through EGI Check-In, additional information is requested (e.g.: username and a brief text to motivate the need to access the service) to create an account in the INDIGO-DataCloud PaaS orchestrator.
- Click on “**Register**” to submit the form and create an account in the PaaS orchestrator (Fig. 4).



## Register at **indigo-dc**

You have been successfully authenticated with **an OIDC identity provider**, but your credentials are **not** yet linked to an **indigo-dc** account.

To proceed with the registration please fill in your personal information below.

To abort this registration click [here](#).

**Given name****Family name****Email****Username**

Please choose a username

**Notes**

Please provide a reason for your registration request

**External authentication type**

OIDC

**External authentication provider**

<https://aai.egi.eu/oidc/>

**Subject**

517e68d16df38e70433932193e955d6d66591271cec59d2656fb1d8abf5b14b3@egi.eu

FIGURE 4 - FILL IN THE REGISTRATION FORM WITH MISSING INFORMATION



## Request submitted successfully

Your registration request has been submitted successfully.

An email with a confirmation link is being sent to the email address provided in the registration form. Check your mail!

[Back to Login Page](#)

**FIGURE 5 - REGISTRATION REQUEST SUBMITTED SUCCESSFULLY**

In case the registration request has been successfully submitted the user will be notified by email with a validation link to confirm (Fig. 6).

iam@cloud-vm195.cloud.cnaf.infn.it  
Confirm your indigo-dc registration request  
A: antonacci@infn.it

Entrata - ba.infn.it 22:29



Dear Marica Antonacci,

you have requested to be a member of indigo-dc.

In order for the registration to proceed, please confirm this request by going to the following URL:

<https://iam-test.indigo-datacloud.eu/registration/verify/845e2241-872e-41da-a79c-6d122842b9c1>

The indigo-dc registration service

**FIGURE 6 - CONFIRM THE REQUEST TO GENERATE THE INDIGO-DC USER'S ACCOUNT**

Clicking on the validation link, the request to generate a new user's account will be confirmed as shown below:



## Request confirmed successfully

Your registration request has been confirmed successfully, and is now waiting for administrator approval. As soon as your request is approved you will receive a confirmation email.

[Back to Login Page](#)

FIGURE 7 - REGISTRATION REQUEST CONFIRMED SUCCESSFULLY

**The administrator of the INDIGO-DataCloud IAM is now notified about the new user's request.**

## 8.2 Step 2. Set the password for the account

Once the request for account is approved by the PaaS admin, the user will receive a second email with a link to set the password for his/her account:

iam@cloud-vm195.cloud.cnaf.infn.it

Your indigo-dc account is now active

A: antonacci@infn.it

Entrata - ba.infn.it 22:39



Dear Marica Antonacci,

your registration request has been approved.



You can set your password by following this link:

<https://iam-test.indigo-datacloud.eu/iam/password-reset/token/467ff1f7-7aca-4dea-a72f-a2028ff8a474>

The indigo-dc registration service

FIGURE 8 - THE INDIGO-DC USER'S ACCOUNT HAS BEEN APPROVED BY THE ADMINISTRATOR

- Click on the link provided in the email and set your password:

  
INDIGO - DataCloud  
  
Set your password  
  
  
  
  
  
  
  
INDIGO - DataCloud  
  
Your password has been reset successfully!  
  

**FIGURE 9 - PASSWORD SUCCESSFULLY SET FOR THE USER.**

## 8.3 Step 3. Access the INDIGO PaaS Orchestrator

As soon as the user's account has been created, through the INDIGO PaaS Orchestrator [29] dashboard it will be possible to select one of the available recipes for deploying new clusters in the Cloud Infrastructure (Fig. 10).

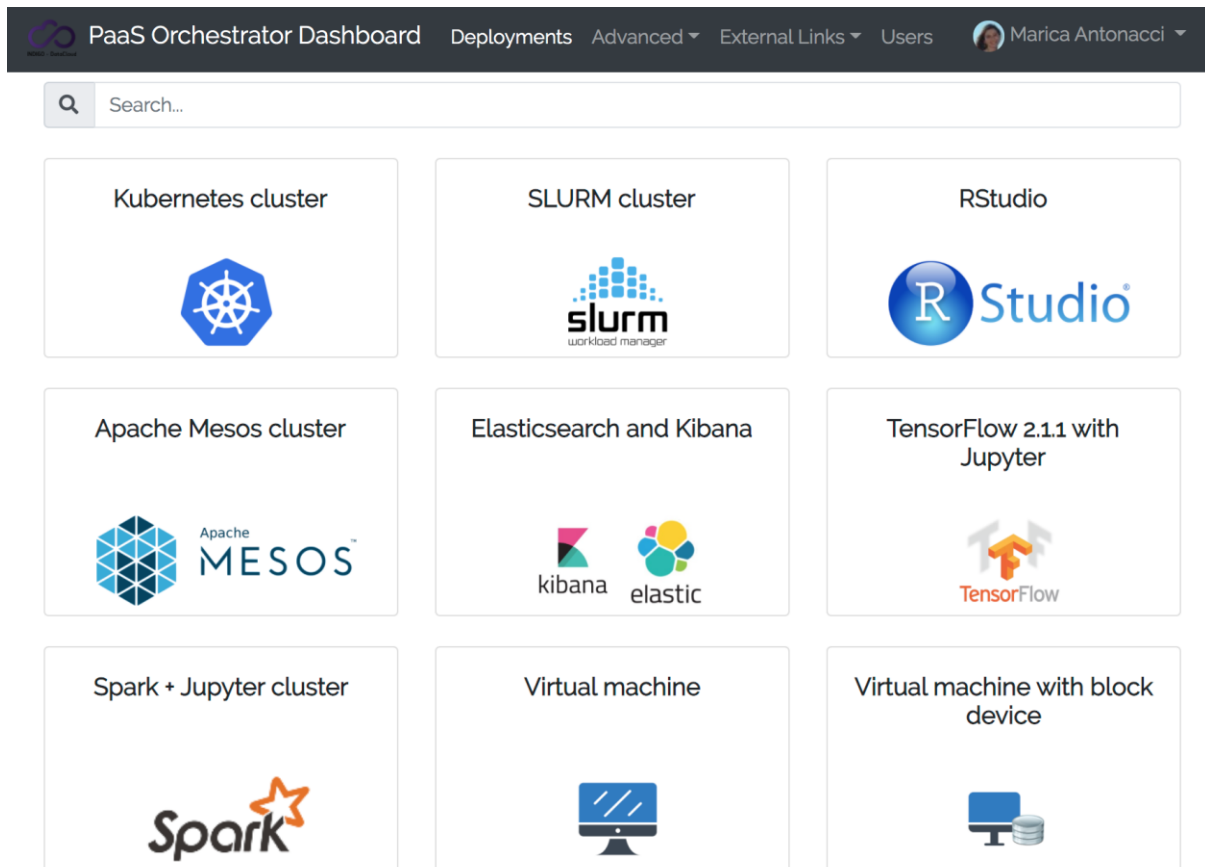
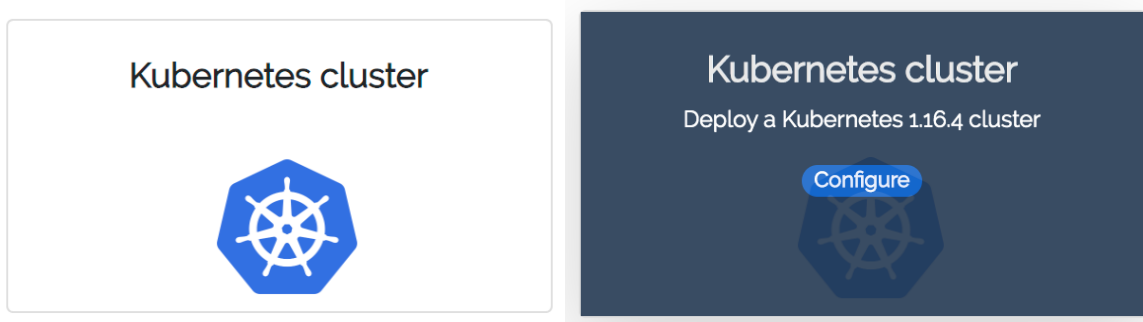


FIGURE 10 - THE INDIGO-DATACLOUD PAAS ORCHESTRATOR DASHBOARD

### 8.3.1 Deploying of a Kubernetes cluster

Once logged in the INDIGO PaaS Orchestrator dashboard, move the mouse over the “Kubernetes Cluster” tile and click the “Configure” button:



### Kubernetes cluster

Description: Deploy a Kubernetes 1.16.4 cluster

Deployment description

kubernetes cluster

Configuration

admin\_token

.....

token for accessing k8s dashboard

number\_of\_nodes

1

number of K8s node VMs

flavor\_master

large: 4 VCPUs, 8 GB RAM

Number of vCPUs and memory size for K8s master VM

flavor\_node

medium: 2 VCPUs, 4 GB RAM

Number of vCPUs and memory size for each K8s node VM

Submit Cancel

**FIGURE 11 - CONFIGURE THE SETTINGS OF THE KUBERNETES CLUSTER**

Clicking on the **“Submit”** button, the PaaS orchestrator will proceed with the deployment of the cluster in the EGI Cloud. Once the cluster is configured, the user will be able to access the front-node of the cluster with the public SSH key generated during the configuration.

PaaS Orchestrator Dashboard Deployments Advanced External Links Users Marica Antonacci

### My deployments

Refresh + New deployment

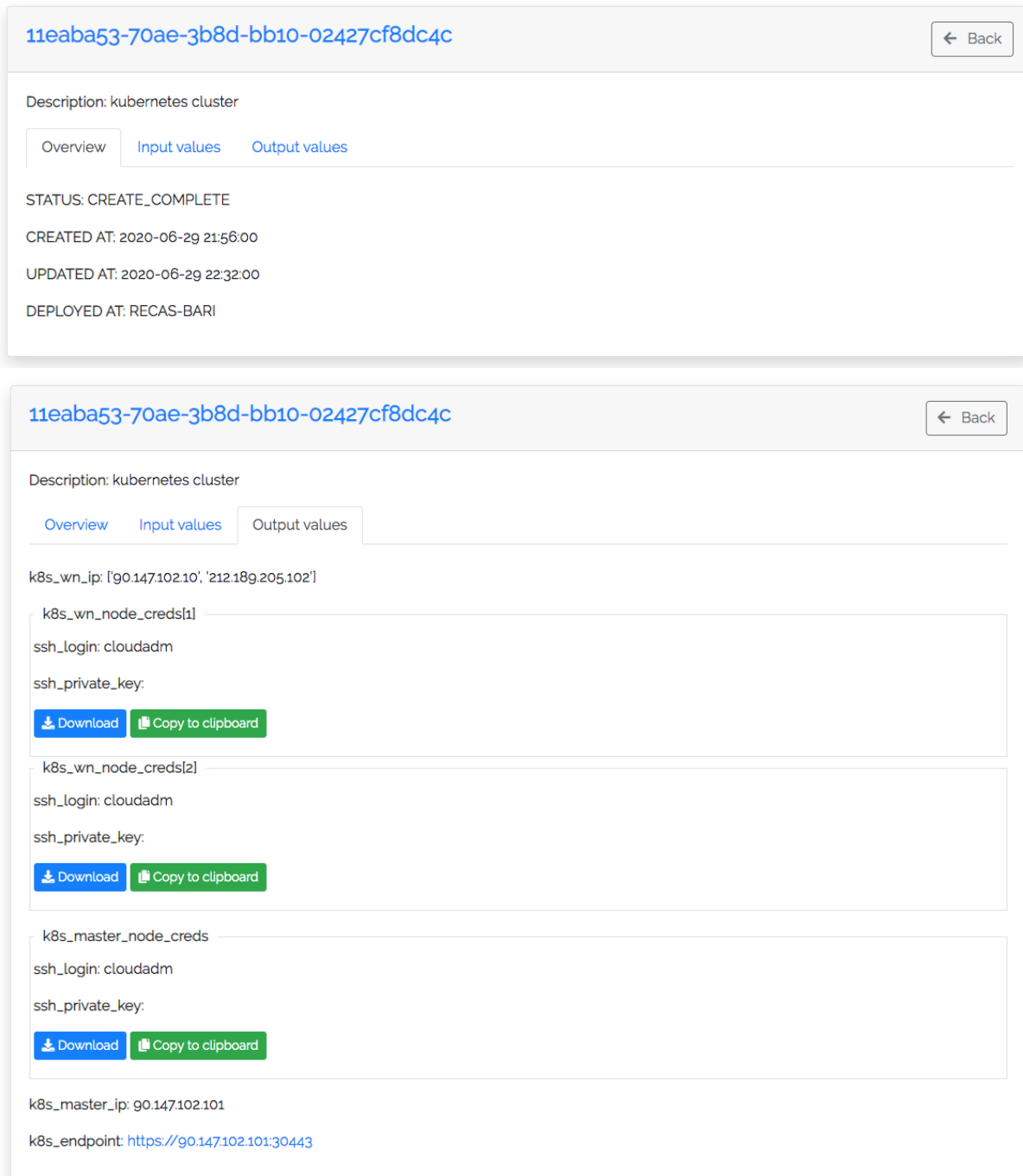
Show 10 entries Search:

Description	Deployment identifier	Status	Creation time	Deployed at	Actions
kubernetes cluster	11eaba53-70ae-3b8d-bb10-02427cf8dc4c	CREATE_COMPLETE	2020-06-29 21:58:00	RECAS-BARI	Details
cmdb	11eab7bd-b61d-58fd-bb10-02427cf8dc4c	CREATE_COMPLETE	2020-06-26 15:00:00	RECAS-BARI	Details
spark	11eab617-5d5b-9fd4-a07d-02427cf8dc4c	CREATE_COMPLETE	2020-06-24 12:36:00	RECAS-BARI	Details
k8s cluster	11eaa99f-4927-551b-a07d-02427cf8dc4c	CREATE_COMPLETE	2020-06-08 15:46:00	RECAS-BARI	Details

Showing 1 to 4 of 4 entries Previous 1 Next

**FIGURE 12 - LISTING AVAILABLE DEPLOYMENTS**

Clicking on the **Deployment Identifier** (or the Details button), the user can access the deployment details, in particular in the Output panel you can retrieve the ssh keys to access the cluster nodes and the kubernetes dashboard URL:



The screenshot shows the deployment details for a Kubernetes cluster. The top section displays the deployment identifier `11eaba53-70ae-3b8d-bb10-02427cf8dc4c` and a 'Back' button. Below this, the description is 'kubernetes cluster'. There are three tabs: 'Overview', 'Input values', and 'Output values'. The 'Overview' tab is active, showing the status 'CREATE\_COMPLETE', creation time '2020-06-29 21:56:00', update time '2020-06-29 22:32:00', and deployment location 'RECAS-BARI'.

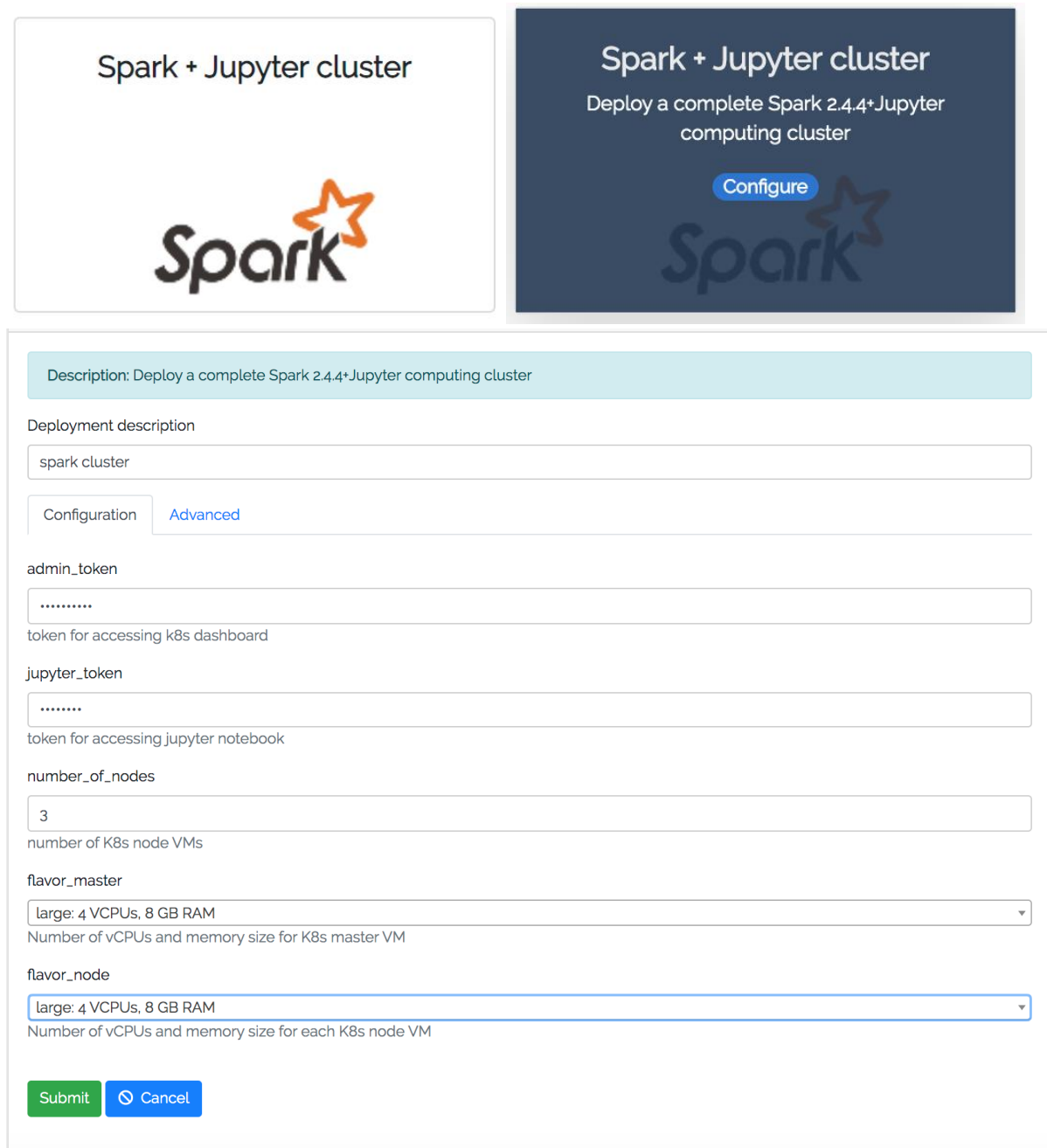
The 'Output values' tab is also shown, displaying the following information:

- `k8s_wn_ip`: ['90.147.102.10', '212.189.205.102']
- `k8s_wn_node_creds[1]`:
  - `ssh_login`: cloudadm
  - `ssh_private_key`: [Download] [Copy to clipboard]
- `k8s_wn_node_creds[2]`:
  - `ssh_login`: cloudadm
  - `ssh_private_key`: [Download] [Copy to clipboard]
- `k8s_master_node_creds`:
  - `ssh_login`: cloudadm
  - `ssh_private_key`: [Download] [Copy to clipboard]
- `k8s_master_ip`: 90.147.102.101
- `k8s_endpoint`: <https://90.147.102.101:30443>

FIGURE 13 - GET DETAILS ABOUT A SPECIFIC DEPLOYMENT

### 8.3.2 Deploying a Spark (in K8s) cluster

Once logged in the INDIGO PaaS Orchestrator dashboard, move the mouse over the “Spark + Jupyter Cluster” tile and click the “Configure” button:



**Spark + Jupyter cluster**

Deploy a complete Spark 2.4.4+Jupyter computing cluster

[Configure](#)

---

**Description:** Deploy a complete Spark 2.4.4+Jupyter computing cluster

Deployment description

spark cluster

Configuration [Advanced](#)

admin\_token

.....

token for accessing k8s dashboard

jupyter\_token

.....

token for accessing jupyter notebook

number\_of\_nodes

3

number of K8s node VMs

flavor\_master

large: 4 VCPUs, 8 GB RAM

Number of vCPUs and memory size for K8s master VM

flavor\_node

large: 4 VCPUs, 8 GB RAM

Number of vCPUs and memory size for each K8s node VM

[Submit](#) [Cancel](#)

**FIGURE 14 - CONFIGURE THE SETTINGS TO DEPLOY A SPARK + JUPYTER CLUSTER**

Clicking on the Deployment Identifier or the Details button, you can access the deployment details, in particular in the Output panel you can retrieve the ssh keys to access the cluster nodes and the endpoints of the graphical interfaces of kubernetes, spark and jupyter:

11eab617-5d5b-9fb4-a07d-02427cf8dc4c

← Back

Description: spark

Overview

Input values

Output values

k8s\_wn\_ip: ['212.189.205.99', '212.189.205.98']

jupyter\_endpoint: <http://212.189.205.97:30888>

k8s\_wn\_node\_creds[1]

ssh\_login: cloudadm

ssh\_private\_key:

Download

Copy to clipboard

k8s\_wn\_node\_creds[2]

ssh\_login: cloudadm

ssh\_private\_key:

Download

Copy to clipboard

sparkui\_endpoint: <http://212.189.205.97:30808>

k8s\_master\_node\_creds

ssh\_login: cloudadm

ssh\_private\_key:

Download

Copy to clipboard

k8s\_master\_ip: 212.189.205.97

k8s\_endpoint: <https://212.189.205.97:30443>

FIGURE 15 - GET DETAILS ABOUT A SPARK DEPLOYMENT