DELIVERABLE D4.3

Grant Agreement number : CIP-297300
Project acronym : InGeoCLOUDS
Project title : INspiredGEOdata CLOUD Services

Funding Scheme : Pilot B

Design and Implementation Report

D4.3

Version 1

Reference D4.3-INGC

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

ContractNumber : CIP-297300

Document Title : Design and Implementation Report

Document version : 1

Document status : Approved

Date : 2014-02-19

WP contributing to the deliverable : *WP4*

Availability : *Public*

Authors : AKKA and all technical teams

Approved by : InGeoCLOUDS Steering Committee

Abstract

The InGeoCloudS platform entails various technical and functional components that are combined in a distributed architecture. This document presents main design choices and some strategic decisions for the implementation of the different functionalities in the cloud. Frameworks and open source components used and integrated in each and every InGeoCloudS main modules are reminded. The document also gives some insights in the development tools used by the teams and in main test results.

Keywords List

Middleware, software code, configuration, security, services, workspaces, accounts, authentication, user management, analytics, metadata, INSPIRE implementation, download, portal

DOCUMENT CHANGE LOG

Document Issue.	Date	Reasons for change
Version 1-Draft 1	2014-01-13	Creation of the document
Version 1-Draft2	2014-01-27	Integration of partners contributions, revision of abstract and objectives sections.
Version1-FinalDraft1	2014-02-17	All contributions but some pending issues on BRGM and CNR parts.
Version1-Approved	2014-02-19	Final integration and proof-reading for delivery

APPLICABLE AND REFERENCE DOCUMENTS (A/R)

A/R and Document Reference	Title
[A1] ICT PSP Grant Agreement N° CIP 297300	InGeoCLOUDS Grant Agreement and its annex (including the description of work)
[R1] D3.1.1-INGC	Analysis and Monitoring of clouds for geo-data services
[R2] D3.1.2-INGC	Analysis and Monitoring of clouds for geo-data services
[R3] D2.1-INGC	Use Cases for InGeoCLOUDS data and services
[R4] D2.2-INGC	Interface of Web Services and models of data
[R5] D3.2-INGC	Cloud architecture, configuration and data access implementation.
[R6] D3.3-INGC	Maintenance plan and service profiling
[R7] D4.2-INGC	Release note accompanying D4.2 (Pilot2 delivery)
[R8] D5.1-INGC	InGeoCloudS User Documentation
[R9] CS-INGC-02 and CS-INGC-04	Replication mechanisms and High-availability architecture with Postgresql See http://www.InGeoCLOUDS.eu/?q=task-32-infrastructure-and-data-access-functionnalities/replication-postgresql
[R10] CS-INGC-03	Supervision Tools (comparative study of a couple of solutions)
[R11] D6.5.1-INGC	InGeoCloudS Pilot Exploitation Plan
[R12] CS-INGC-16	ElasticDB-ElasticityTests-V1.0.doc

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

1.1. ACRONYMS AND DEFINITIONS

Term	Definition
Amazon AWS	Amazon Web Services, i.e. the amazon cloud computing infrastructure
API	Application Programming Interface
CDR	*Customer Data Record*: atomic piece of information conveying data about usage of a service (originally in telecom domain). CDRs are used for accounting and billing.
Data Provider	A user willing to contribute to InGeoCLOUDS with his own data or with a novel service
Data Provider Service	Web-accessible service based on InGeoCLOUDS system
DOI	Digital Object Identifiers
GDAL	Geospatial Data Abstraction Library
GFS	Gluster File System
IGC	InGeoCLOUDS
LDAP	Lightweight Directory Access Protocol
LOD	Linked Open Data
N/A	Not Applicable
NFS	Network File System
ODBC	Open Database Connectivity
OGC	Open Geospatial Consortium
OWL	Web Ontology Language
RDF	Resource Description Framework
Registered User	Visitor of the platform that performed the registration process
REST	REpresentational State Transfer
SPARQL	SPARQL Protocol and RDF Query Language
SPARUL	A.k.a. SPARQL/Update, is a declarative data manipulation language extending SPARQL
URIs	Universal Resource Identifiers
WFS	Web Feature Service
WMS	Web Map Service
Workspace	Provides to data providers the possibility to store (and access) own data in a private database of the InGeoCLOUDS Elastic Database, and/or in a private folder of the InGeoCLOUDS Elastic File Server

Table 1. Useful acronyms and Definitions

1.2. OBJECTIVES OF THE DOCUMENT

This document recapitulates the design of main functions that are provided by the InGeoCloudS platform. WP2 documents [R3], [R4] allowed for the definition of main requirements through a comprehensive description of datasets, technical assets used in existing providers infrastructures and specification of needs for the support of more generic features. Document [R5] presented the resulting architecture designed for the cloud. Pilot2 is the resulting infrastructure and platform that hosts the different use case applications. In this document, we will go more in details through the design principles that have been recently followed for each component of the infrastructure of the production of Pilot2. The document also describes how data providers' datasets and applications are technically integrated in the platform. It also provides some insights in the software development and integration rules and tools that are followed by technical teams.

1.3. OVERVIEW OF THE DOCUMENT

Chapter 2 recaps the architecture principles that have been followed for integrating the different logical components of the platform: RESTful APIs, segmentation and high-availability solutions, security……

Chapter 3 presents the concepts and integration assets exposed to data providers for the integration of their datasets and services in the platform: notion of workspaces, user accounts and privileges.

Chapter 4 focuses on higher-lever services / utilities featured by Pilot2 and targeted to data providers, data consumers and exploitation teams: technical insights and integration are described.

Chapter 5 lists development environment, tools and procedures that have been settled since beginning of the development/integration tasks. It also gives some results of how main tests on middleware components have been performed.

Section 6 draws some final conclusions about the efficiency of the technical developments and technical environment of the platform. Italso lists some consideration on possible technical scenarios for the future.

2. INGEOCLOUDS ARCHITECTURE PRINCIPLES

2.1. ARCHITECTURE PRINCIPLES

Main messages to convey:

- Restful API / Stateless services: any service instance can serve any request in a timely fashion and so, a server failure does block the whole system. In case of failure, requests can be routed to another service instance and we can automatically initiate a new node to replace it.
- Design for the Cloud: it is not just lifting existing applications from a data center to Amazon's EC2: we designed for resiliency.
- High-Availability / Design for failure (at least on key services like the middleware): what to do when a component fails? It has an answer for ElasticDB, FS, Web Mapserver components with an active/passive schema that allows recovery of operation in short time.
- Dependencies with AWS are limited to fine-grained technical solutions where we were not capable to compete with our resources. An example of it is S3 storage. Literature about S3 (used in mapcaching) has proven to be highly reliable (degrading rather than failing). . See section 2.4 for a synthesis about InGeoCloudS architecturedependencies

2.2. LOGICAL VIEW AND DESIGN

The Figure 1 below illustrates a logical view of the InGeoCLOUDS architecture. Note that it is an updated version of the architecture design described in previous D3.2 and D3.3 deliverables [R5] and [R6]. For each software component, the exposed services and their interactions are shown. Section 2.3 below lists the services and the comprehensive technical documentation of the REST APIs can be attained by following the URLs specified.

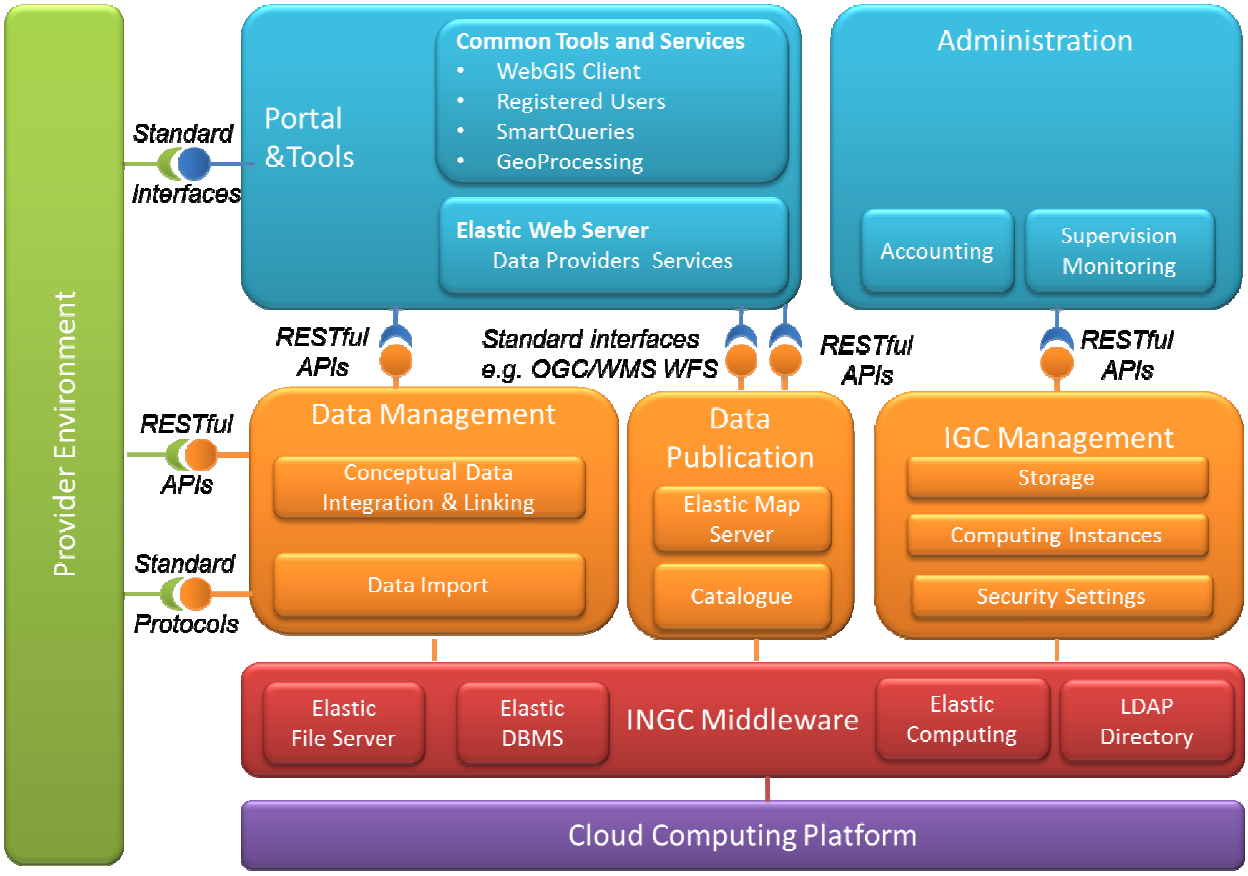


Figure 1: Logical view of InGeoCLOUDS architecture: components diagram

2.2.1. INGEOCLOUDS MIDDLEWARE

2.2.1.1. Design and Implementation

2.2.1.1.1. Elastic File Server

The INGeoCloudS Elastic File Server relies on GlusterFS, an open source, distributed file system capable of scaling to large volumes of data and clients. It clusters together storage building blocks over the network, aggregating disk and memory resources and managing data in a single global namespace. GlusterFS was developed with three objectives in mind: elasticity, linear scaling and scale-out.
In InGeoCloudS, the choice of this component allows to provide data providers with familiar NFS environment for their workspaces and to be as much independent as possible from similar off-the-shelf solutions offered by AWS, releasing yet another point of lock-in risk.

Load balancing of Elastic File Server directly relies on GlusterFS Elastic Hashing Algorithm which allows locating a file on the system without the use of file metadata server.

High availability is one of the most cited requirements from data providers and especially for data storage part like the Elastic File Server. High-availaibility directly relies on GlusterFS ability to scale-out on the Amazon cloud platform. See [R5] for more details.

Data disaster recovery relies on scripts that periodically backup all the files located on GlusterFS volumes to the Amazon S3. In this case, the Amazon service offers both efficiency, reliability and low-cost for this utility function.

The INGeoCloudS Elastic File Server is managed by the following RESTful services:

- administration of the Elastic File Server component: start and stop the component, set storage capacity, set number of server and set distribution schema,
- monitoring the Elastic File Server component:: general status and detail status (number of servers, storage capacity),
- backup and recovery of the files located on GlusterFS,
- management of the providers' workspaces: creation and deletion.

The Elastic File System Service documentation is online on the Pilot 2 portal . The API implementation is described in Section 5.1.

2.2.1.1.2. Elastic Database Server

The INGeoCloudS Elastic Database Server relies on PostgreSQL/PostGIS, the most used solution in GIS domain. This also has been the privileged choice for the consortium (see [R1]). Both PostGIS 1.5 and PostGIS 2.0 are supported so data providers can use one or the other.

Load balancing of Elastic Database Server relies on Pgpool, a dedicated load balancer for PostgreSQL servers. This choice (versus regular Load-Balancing services offered by Amazon) is again to avoid too much dependency with the CSP and to lower costs.

High availability of Elastic Database Server relies on replication mechanisms provided by PostgreSQL, and a failover solution to ensure robustness. See [R5] for more details.

Data disaster recovery relies on scripts that periodically backup all the databases to the Amazon S3.

The InGeoCloudS Elastic Database Server is managed through the following RESTful services:

- administration of the Elastic Database Server component: start and stop the component, set number of slave server, add pgpool server,
- monitoring the Elastic Database Server component:: general status and detail status (number of servers, storage capacity),
- maintenance of the database server: vacuum, rebuild indexes,
- backup and recovery of all the databases or of a provider database,
- management of the providers' databases: creation and deletion, status, connection information.

The Elastic Database Service documentation is online on the Pilot 2 portal . The API implementation is described in Section 5.1.

2.2.1.1.3. Elastic Computing

The InGeoCloudS Elastic Computing is a set of API that relies on Amazon EC2 services. Amazon API is encapsulated to ensure a loose coupling with the underlying cloud platform. The Elastic Computing API is internally used by most of the RESTful services of the platform.

Load balancing of Elastic Computing relies on Amazon Elastic Load Balancer (ELB) service. However, because of a loose coupling, other load balancers can be used in place of Amazon ELB (e.g. Pgpool is used for ElasticDB)

As for all RESTful services of the platform, high availability of Elastic Computing relies on the possibility to easily replace the unhealthy API instance. See [R5] for more details.

Data disaster recovery is not needed in Elastic computing because API is stateless.

On the top of Elastic Computing API, the INGeoCloudS Computing provides the Geo Processing RESTful service to start compute instances on demand:

- start or restart a compute instance,
- stop or terminate a compute instance,
- getting status of a compute instance.

Geo Processing API is useful to run batch processes on dedicated and ephemeral instances.

The Geo Processing Service documentation is online on the Pilot 2 portal. The API implementation is described in Section 5.1.

2.2.1.1.4. User Storage

The user storage relies on OpenDJ, an open source LDAP directory server that provides LDAPv3 support as well as RESTful access to directory data.

Using an LDAP directory allows centralizing the user management. The LDAP directory is internally used, directly or indirectly by all the components of the platform that deal with user management and user authentication.

OpenDJ offers built-in data replication to ensure high availability.

2.2.1.2. Off-the-shelf Tools and Frameworks

This shall be an update or reminder of what we had in D3.3

Technical Asset	Version used	Inside InGeoCloudS
PostgreSQL	V9.1.7	ElasticDB
PostGIS	V1.5.3 and V2.0.1	ElasticDB
Pgpool	V3.2.3	ElasticDB
GlusterFS	V3.4	ElasticFS
OpenDJ	V2.5.0-Xpress1	LDAP Directory
Amazon AWS: • **EC2** • **EBS** • **S3** • **Elastic IP** • **Load Balancer** • **Elastic Group**	Not Applicable	ElasticComp

2.2.1.3. Key Design and Implementation Issues

Security

The whole middleware API (ElasticDB, ElasticFS, Elastic Map Server, Elastic Compute) has been upgraded in Pilot-2 for integrating security mechanisms as described in Section 3.3

Scalability

Some improvements have been achieved with respect to Pilot-1 deployment. In Pilot-1, not all layers of the architecture were exploiting the elasticity facilities provided by the Elastic Compute component. Now elasticity is exploited not only by the Elastic Web Server, but also by the Elastic Database Server, Elastic File System, Elastic Map Server and Elastic Linked Data Storage. Also, the Geo-Processing layer is supporting on-demand computation for use cases such as the shakemap one, by allowing running and stopping instances on demand. The next maintenance period will be very useful for gathering statistics about the behaviour of the platform and to tune the elasticity of the different InGeoCloudS services.

Refactoring of source code projects and configuration files

An important refactoring of the project's code and more efficient configuration management rules have been put in place for Pilot2 developments and beyond. The objective has been to improve independence of developer teams for the parts they are accountable for and to improve maintainability of the code by removing dependencies between modules and configurations. More details are available in section 3.3.1 of deliverable D3.3

Data storage

In Pilot 2 we deployed an OpenDJ installation instead of an OpenLDAP installation. Unfortunately, at time being, OpenAM does not officially support OpenLDAP as a LDAP backend: OpenAM uses specific LDAP schemas which have not been implemented yet on OpenLDAP. A lot of work would be necessary to adapt the LDAP schemas to OpenLDAP. Because OpenDJ is fully supported by OpenAM, We decide to use it instead of OpenLDAP. However, the integration of OpenLDAP with OpenAM would enhance the sustainability since OpenLDAP is a de facto standard for an open source implementation of LDAP directories,. We will of course survey evolutions of OpenAM to make this change whenever possible.

2.2.2. INGEOCLOUDS MANAGEMENT

2.2.2.1. Design and Implementation

2.2.2.1.1. User Management

User management relies on OpenAM, an open source project providing services for authentication, authorization and identity services.

OpenAM provides RESTful services and a console that makes user management more flexible.

Some users of the platform must have a POSIX account. OpenAM does not provide an easy way to manage POSIX account. So an extension was implemented and deployed with OpenAM to simplify POSIX account management: when using the RESTful services or the console, The administrator just indicates if the user to be created or to be modified is a POSIX account. POSIX information like user id, group id and home directory are automatically computed or can be specified as user attributes.

The platform also provides RESTful services to manage data providers (through the Master Service API) and registered users (through the Registered User Service API). These APIs rely on the SDK provided by OpenAM, which ensures that CRUD operations are delegated to OpenAM.

Creating a new data provider implies the creation of its workspace (see also section 3.2 below). This is automatically done by the Master Service API.

The platform API also registers to OpenAM for notification of user creation or modification. This ensures the workspace is created when an administrator creates a new data provider using the OpenAM RESTful services or OpenAM console instead of the Master Service API,

2.2.2.1.2. User Authentication and Authorization

For the HTTP access, the platform authentication relies on single sign-on (SSO) which is more secure and simpler for the end-users than the basic HTTP authentication. The end-user authenticates once and accesses the different components of the platform thanks to a unique token that has a limited period of validity.

SSO implementation relies on OpenAM, an open source project providing services for authentication, authorization and identity services.

Most of the components of the platform that provide HTTP services use SSO to identify the end-user:
portal of the platform (Sitools) , data providers' applications,data publication applications (Geonetwork),, API of the platform (RESTful services).

OpenAM provides RESTful services and SDK to allow components checking the validity of a token.

Once a user is authenticated, the component is responsible for managing the authorizations. The SSO token allows components to retrieve user information (e.g. group) that are used to filter access to the resources.

Other components of the platform require user authentication for services that are not HTTP oriented:
System accounts, file permissions and system access (SSH, FTP, etc.). These components rely on POSIX accounts instead of SSO: they are configured to delegate the authentication to the underlying operating system.

All instances are configured to delegate system account management and authentication to the LDAP directory. This allow to transparently manage users, authentication and file permissions.

Obviously, only users with POSIX account are supported by the underlying operating system.

2.2.2.1.3. Analytics

Basic analytics functions have been integrated in the platform consisting in logging and presenting usage information for the different projects that are integrated in the portal. These facilities are provided through the integration of Piwik tool (www.piwik.org), a powerful and comprehensive open source solution of analytics of web servers. This feature is now part of the platform.

2.2.2.1.4. Monitoring, Supervision and Accounting

The technical monitoring relies on a quite pragmatic ad hoc implementation defined in terms of indicators and alarms. Each indicator measures the status of the platform according to a specific point of view, while an alarm signifies an event that requires some intervention or further investigation (e.g., excessively large response time).

The indicators can be queried at any time, and they are also organized so as to provide the historical monitoring of the platform. A detailed description of each indicator is given in Deliverable D.3.3 "Maintenance Plan and Service Profiling" [R6]

An ***Alarm*** is defined by an *event* and a list of *correcting actions*:

- **Event**: the occurred event that requires some action to be undertaken (e.g., "*the system is too slow*"). The event is defined on a basis of some indicators (e.g., by doing simple correlation of concurrent measurements) and a proper triggering threshold (e.g., *avg. response time* is *larger than 5 seconds*);
- **Correcting Actions**: a list of correcting actions that should be undertaken when the event is detected. We exclude from the scope of this document those automatic actions, e.g. elasticity, that take place without harming the quality of service of the system and that do not require any kind of human intervention. We focus on those actions that require some manual investigation (e.g., *modify the deployment strategy*). We believe that most of the correcting actions are reconfiguration or fine-tuning that can be implemented quickly.

The set of indicators allow a continuous monitoring of the InGeoCloudS platform and of its software components in order to ensure a satisfactory quality of service. The set of alarms defines the proposed maintenance plan, i.e., the correcting actions to be undertaken over time.

Another important component developed inside the InGeoCloudS platform is the **accounting service**. Some costs can be split among users in a very simple way, e.g., storage in proportion to the volume of data owned by users. Some other costs, and in particular regarding shared services (e.g., CPU cost of the Elastic Database), are handled differently. Through the analysis of the logs of the platform applications (e.g., number of Web requests), we estimate the amount of use of shared services, and this is used to split the cost of each service among the different data providers. The main objective of this service was to provide the tools (low-level data analysis and user friendly web interface) required to monitor the behaviour of the platform and the cost of its services, as well as to support billing according to the share of the clouds infrastructure used.

This is currently used by the consortium to better understand and evaluate the costs of the InGeoCloudS platform, and to design a proper business plan for future users. After one or more of such business plans are defined, the cost billing service will be changed accordingly. The costs of the services of the platform are monitored within the monitoring framework of the platform.

2.2.2.2. Off-the-shelf Tools and Frameworks

Technical Asset	Version used	Inside InGeoCloudS
OpenAM	10.1.0-Xpress	IGC Management
Piwik (analytics tool)	1.9.2	IGC Management
Amazon AWS: • **RDS** • **CloudWatch** • **S3**	Not Applicable	Not Applicable

2.2.2.3. Key Design and Implementation Issues

Monitoring

At the beginning of the Pilot 2 development, we implemented the monitoring system by using the Nagios[1] framework. This is illustrated in the deliverable D3.3. After some experience, we decided to drop the previously proposed solution, and we preferred to store the data in a permanent database provided by the cloud platform. The approach is similar to that of Nagios, where a number of scripts are deployed onto the monitored machines and they send some indicator information to a monitoring server. But we found that storing the data directly in a permanent database without passing through a Nagios server would allow to remove one component to be maintained (i.e. Nagios) to have availability of the recorded behaviour of the platform at any time, independently from the platform itself and its Nagios server. This required changing the monitoring scripts implemented before.

An important upgrade of the monitoring service have been put in place in Pilot2 in order to support the accounting service. The objective has been to modify the Elastic Compute Component so that the identification information required for identifying the IGC sources of costs are attached to the components of the platform when they are launched. By doing that we are now able to provide an estimation cost on a per-service basis.

Security

In Pilot 2 we deployed an OpenAM installation to enhance the user management and security management. Unfortunately, the version of OpenAM which is deployed is not fully compliant with the emerging security standards like OAuth 2.0 and OpenID Connect 1.0. Upgrade to last release would enhance sustainability.

2.2.3. DATA PUBLICATION

2.2.3.1. Off-the-shelf Tools and Frameworks

Technical Asset	Version used	Inside InGeoCloudS
GeoNetwork	V2.10	GeoCatalog
MapServer	V6.2	Map Server

[1] http://www.nagios.org

2.2.3.2. Key Design and Implementation Issues

In Pilot-2, the GeoCatalog was deployed in the data publication component. SSO has been introduced in the code of Geonetwork by AKKA (contribution to the OpenSource project pending). The map publication API was improved with the integration of the other improvements of the middleware (e.g., elasticity) and new functionalities required by use cases.

Mapserver :

Component allowing distribution and query of map layers with respect to OGC standards (WMS, WFS,…)

This service is used by most of the portal components in a significant way. For performance reasons, mapserver has been installed on a EC2 dedicated instance, deployed by IGC API, and loadbalanced for an optimal service.

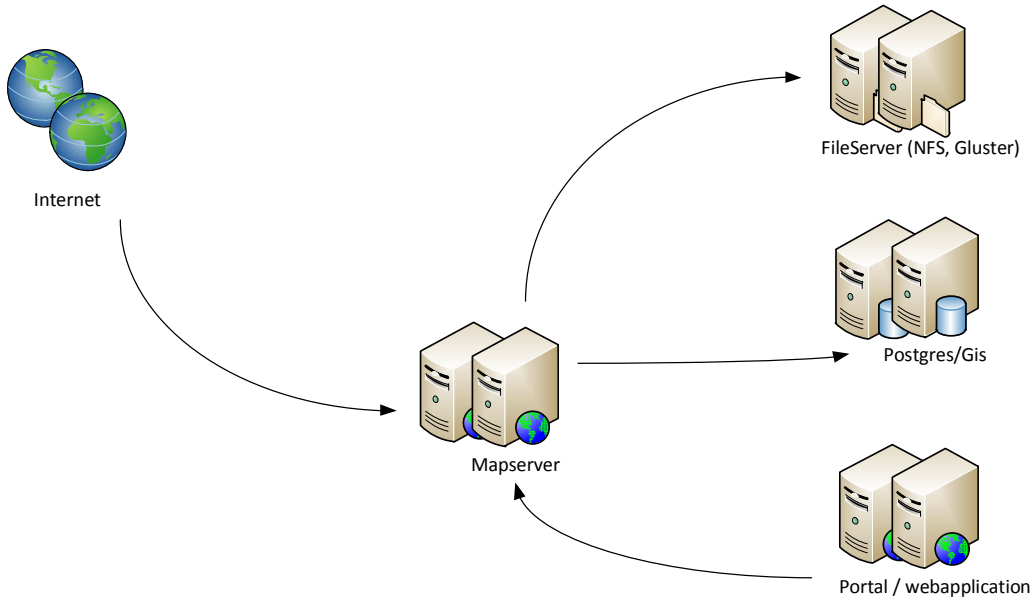


Figure 2: Mapserveur implementation

Mapcache :

The mapcache component is installed on each mapserver. It allows caching each map service.

By building tiles cache, mapcache improves the speed access to all existing mal services including fairly big dat (such as orthophotos, scans…).

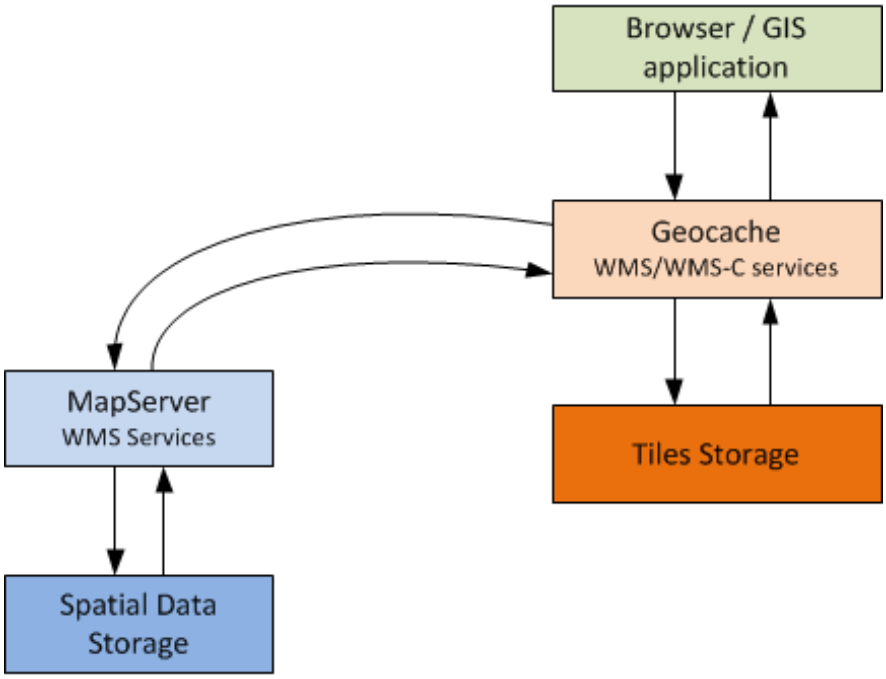


Figure 3: mapcache component

Somehow, Mapcache plays the function of a mapserver proxy. It is build on a map server in order to deal thematic services and manages a disk space to store tiles.

The mapcache architecture deployed is as such:

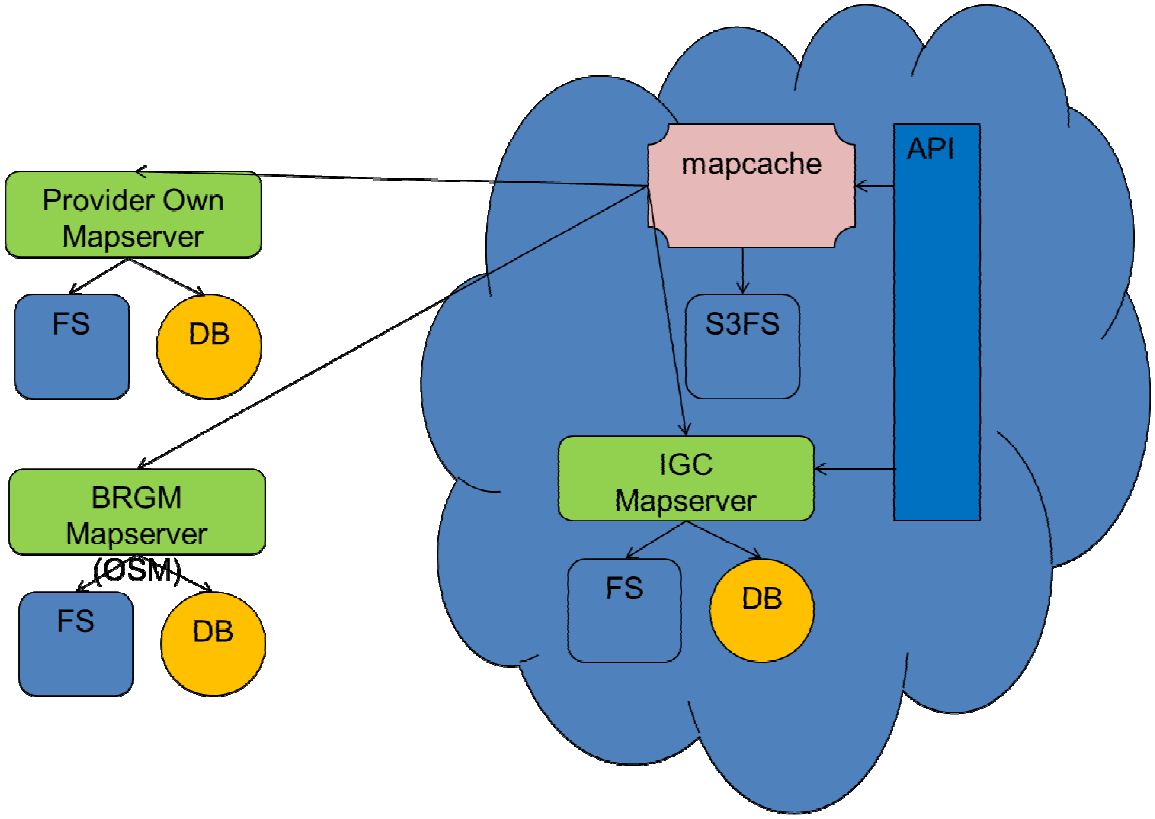


Figure 4: mapcache architecture deployed

Thanks to this architecture, it is possible to cache map services hosted within InGeoClouds, and also map services distributed directly by providers.

Geopublication :

The geopublication service is hosted by a dedicated EC2 instance made of an apache web serveur, the PHP framework and the specific IGC PHP application.

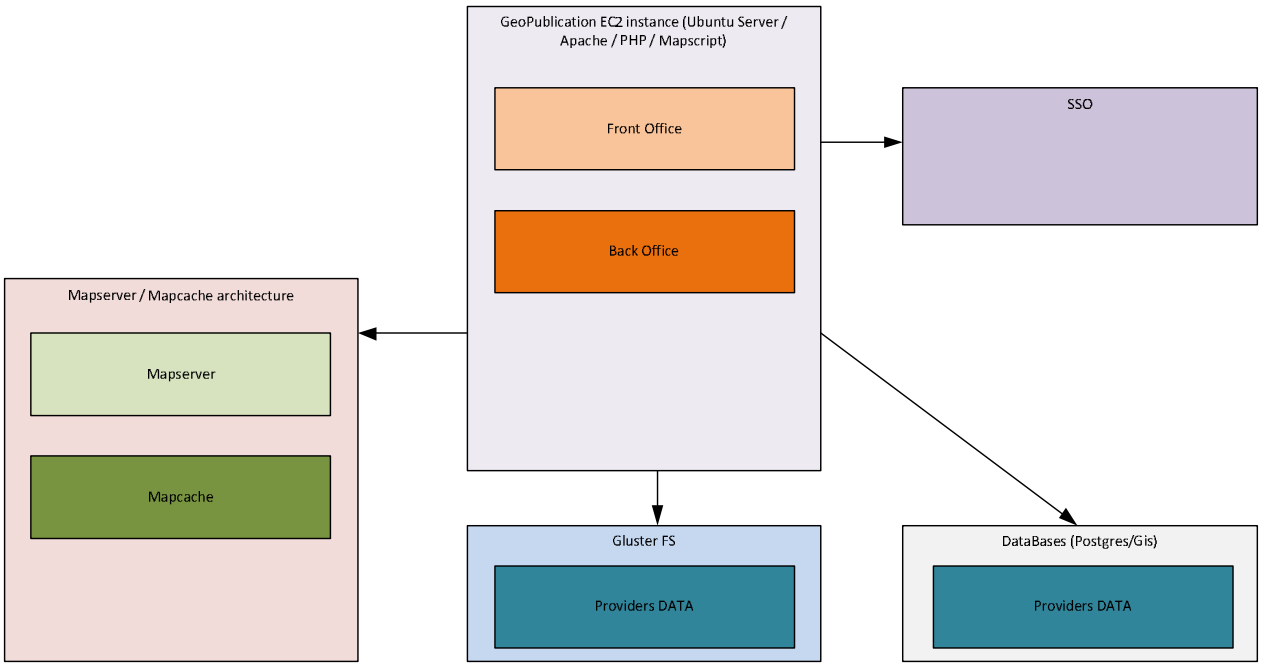


Figure 5: Geopublication implementation (Front office/back office)

API

The API takes care on the creation of Amazon instances (GeoPublication, GeoNetwork, MapServer…) and on the management of mapfiles (creation of WMS INSPIRE services).

Geonetwork

Geonetwork is a simple tool for spatial data and services catalogue. It has been improved by AKKA to make it compatible with the projet authentification system (SSO). It gives services allowing metadata management.

2.2.4. DATA MANAGEMENT

2.2.4.1. Off-the-shelf Tools and Frameworks

Technical Asset	Version used	Inside InGeoCloudS
Virtuoso Triple Store	V6.1.7	Linked Data Management API
GeoTools	10-SNAPSHOT	Linked Data Management API
uSeekM	V1.2.0-a5	Linked Data Management API

2.2.4.2. Key Design and Implementation Issues

In Pilot-2, we have required the installation/exploitation of the last two components in order to enable support for the evaluation of GeoSPARQL queries. Geotools is exploited in order to enable the production of feature collection representations in different formats (i.e., KML, GML, GeoJSON, and Shape). No other technical asset is planned to be exploited based on the functionality to be exposed by the Linked Data Management API. The last statement does not currently take into consideration the possible, future integration with the use cases designated in the project where some functionality, not foresssen by now, may need to be implemented.

2.2.5. DATA PUBLICATION WEB PLATFORM

2.2.5.1. Off-the-shelf Tools and Frameworks

Technical Asset	Version used	License	Inside InGeoCloudS
Apache	V2.2.2	Apache License	GeoPublication UC
RabbitMq	V3.1.5	Mozilla	GeoPublication UC
MapServer + MapScriptphp	V6.2	MIT	GeoPublication UC
MapCache	V6.2	MIT	GeoPublication UC
Apache / Tomcat	V7.0.26	Apache License	GeoPublication UC
php	V5.X	BSD	GeoPublication UC

2.2.5.2. Key Design and Implementation Issues

In Pilot-2, the GeoPublication application was improved with the integration of evolutions of the middleware API and data publication components.

2.2.6. PORTAL

2.2.6.1. Off-the-shelf Tools and Frameworks

Technical Asset	Version used	License	Inside InGeoCloudS
Sitools2 SITools2 for administrators /data providers is the basis for the Web Portal	V2.1.2	GPLv3	Portal Data Management Smart LOD Queries module

component of InGeoCLOUDS

2.2.6.2. Key Design and Implementation Issues

The Portal includes several ad-hoc modules developments and adaptations for the InGeoCloudS purposes:

- the default WebGIS client
- Web Site editor (for online documentation)
- Sparql-endpoint data source and Opensearch queries
- User Management modules accommodated for Authentication (SSO) and Authorization management in InGeoCloudS
- Web pages branding and visual adaptations

2.3. SUMMARY OF INGEOCLOUDS PLATFORM SERVICES

The platform exposes through its API the following services (in parenthesis the number of methods per service):

InGeoCloudS API - Platform (87)
https://portal.ingeoclouds.eu/ingeoclouds-api/platform/
Accounting Service (4)
Elastic Database Service (24)
Elastic File System Service (15)
Elastic Linked Data Storage Service (3)
Elastic Map Server System Service (4)
Elastic Web Server Service (4)
GeoNetwork Service (3)
Geo Processing Service (5)
GeoPublication Service (3)
Kriging Service (3)
Master Service (10)
Monitoring Service (3)
Registered users service (6)

InGeoCloudS API - Data import: (10)
https://portal.ingeoclouds.eu/ingeoclouds-api/data-import/
File service (5)
Harvest service (5)

InGeoCloudS API - Data publication (19)
https://portal.ingeoclouds.eu/ingeoclouds-api/data-publication/
Data publication - Data store service (5)
Data publication - Layer template service (5)
Data publication - MapFile service (9)

InGeoCloudS API - Linkeddata (13)
https://portal.ingeoclouds.eu/ingeoclouds-api/linkeddata/

GeoLinked Data Service (2)
Linked Data Service (11)

2.4. SUMMARY OF DEPENDENCIES ON AMAZON'S AWS

Usage of the CSP specific services like S3 storage, ESB and ElasticIP has been selected only in cases where either no equivalent technical solution could be developed in the Consortium or where our resources where too limited: we did not want to lose time/money on very technical low-level services where equivalent cheap and efficient solutions are offered by almost all CSPs, and especially by AWS on their catalogue. Dependencies to these services have been isolated in terms of methods calls and encapsulated in a minimal number of modules:

- ElasticCompute: calls to a couple of methods of the Amazon SDK and Amazon S3 as persistency solution for backups,
- Tile Caching in mapserver: on Amazon S3 for better performance and security.

Most of the other components (like ElasticDB, LinkedDataAPI and processing components like ShakeMaps, Landslide prediction and GeoProcessing) do not have any direct dependency on the Amazon Service infrastructure and can be moved easily to other clouds.

2.5. PHYSICAL VIEW AND CLOUD RESOURCES MOBILIZATION

2.5.1. STRATEGY

One of the InGeoCLOUDS design objectives is to ensure portability – as much as possible - to any cloud platform provider. This is mainly achieved by implementing a specific middleware component, the Elastic Compute component, which is designed to ensure loose coupling with the underlying cloud platform. This module acts like an interface between the platform and the cloud provider, by implementing the basic operations regarding the management of compute and storage devices, virtual machines images, load balancing and other cloud related operations. Note that the *Elastic Compute* module is not intended to be as a simple interface to a set of cloud service providers, but rather as a sort of translator between the specific InGeoCLOUDS needs and the services exposed by the cloud service provider.

The consequence of this would be that, if at some point in time we will decide to migrate to a different cloud provider, we should mainly focus our attention/effort on first changing the Elastic Compute module. Of course, we should also find a cloud provider which is more or less compatible with the services provided by Amazon or alternatively we should implement on our own missing services. For example, the load balancer service we use inside the IGC platform could be replaced by other load balancers, as we have already done with PgPool in the Elastic Database layer.

The relational database (RDS) is another critical service used for monitoring/accounting inside the platform. But a lot of other cloud providers offers it, so it should not be a blocking point. On the other hand, the S3 storage service used for backup, the mechanism for tagging every resource inside Amazon for accounting purposes as well as the usage of Elastic IPs could employ a lot of development time in order to obtain a viable solution.

In the next section we will describe the dependency between the IGC services and the Amazon resources.

2.5.2. WHAT DO WE HAVE IN PILOT2

The project has opened in October 2013 (M21) a Pilot2 service that reflects main Period #2 achievements for WP2, WP3 and WP4. Below, we revise the resource mobilization on the basis of the experience gained after the deployment of the second pilot on the Amazon AWS platform.

We used three kinds of instances, which are named m1.small, m1.medium and m1.large, according to the Amazon AWS nomenclature. They correspond to increasingly powerful instances. Their main features can be summarized in Table 2:

AWS Names	*m1.small*	*m1.medium*	*m1.large*
References	*S*	*M*	*L*
CPU Cores	*1*	*2*	*4*
Memory	*1.7 GB*	*3.75 GB*	*7.5 GB*

Table 2: Summary table of the kind of Amazon instances used

We recall the InGeoCLOUDS architecture deployment and report on the cloud resources used. The deployment of InGeoCLOUDS can be described in terms of components as shown in Table 3:

IGC component	***Instances***	***Other AWS resources***
Elastic File Server	1 x S	
Elastic Database Server	3 x S	1 x Elastic IP
Elastic Map Server	1 x S	1 x Load Balancer (apache) S3 storage (tile caching)
Elastic Linked Data	1 x L	
Elastic Web Portal	2 x S	1 x Load Balancer
GeoPublication	1 x M	

GeoNetwork	1 x S	
IGC Backend API server	1 x S	S3 storage (monitoring) RDS (monitoring)
GeoComputational	(on demand)	
TOTAL	**1 x L** **1 x M** **9 x S**	**2 x Load Balancer** **2 x S3 storage** **1 x RDS**

Table 3: Split of resources on a per component basis

The total amount of running instances required by the platform is thus 11. This is not very different from our initial estimate. More, it can increase in size depending from the platform usage and if partially benefits from economy of scale due to the sharing of the resources and services across the data providers.

2.5.3. CURRENT LIMITATIONS AND FURTHER PLANS

The development of the pilot 2 raised some platform limitations that still need to be solved in order to improve the availability of the platform as well as its security. Here is a list of problems with some possible solutions to adopt:

- **API server is a Single Point of Failure:** implement a master/slave solution with the slave that will automatically take care to replace the failed instance. Probably need to use the VPC service provided by amazon for manually admin the IP inside a private network
- **Gluster server is currently running on single instance**: configure the gluster service to run on replication mode, so that we delete the single point of failure in the elastic file service.

How to cross-reference different instances in a multi node system: some services need explicitly to know the IP address of other services, in order to work properly or for being able to support a wide range of features (e.g. improve availability). The VPC service again would easily solve the problem.

3. DATA AND SERVICES INTEGRATION IN INGEOCLOUDS

3.1. DATA PROVIDERS ACCOUNTS

A data provider account provides a unique working space with access to InGeoCloudS Elastic File System and optionally to Elastic Database.

A data provider needs to register to obtain a specific InGeoCloudS account and access credentials. These credentials provides the user with access to the dedicated workspace.

The registration process for data providers is supervised by an InGeoCloudS administrator who must check and authorize the Data Provider account request. In return, the data provider receives its credentials for accessing InGeoCloudS as a Data Provider. Registration process is done by mail.

3.2. WORKSPACES

Each data provider has a dedicated workspace with file system and database.

File system:

Availability:

The workspace on file system is automatically created when a new account is set up.

File hierarchy:

The file hierarchy is predefined and must not be changed:

- /mnt/glusterfs/<PROVIDER_ID>/ is the root of the data provider workspace. For security reasons, it is not writable.
- /mnt/glusterfs/<PROVIDER_ID>/workspace is the top directory a data provider can write in. Data provider manages content as he wants.
- /mnt/glusterfs/<PROVIDER_ID>/workspace/public_web is a special directory to store Web application code. Web application is automatically exposed on the Internet.

Accessibility:

All instances on the platform automatically mount the data providers' workspaces, so a data provider's workspace is shared and accessible as a local directory by any application running on any instances (according to the permissions).

Accessibility from application running on the platform:

A data provider can statically configure the location of the workspace in its application:
/mnt/glusterfs/<PROVIDER_ID>/workspace

However, the location could change in future. It is recommended to check the location with the RESTful service provided by the ElasticFS API (authentication required)

Maintenance:

A data provider can request for a temporary SSH access. Workspace is the home directory, so the data provider can maintain its applications and manage its data.

Accessibility from the Internet:

A data provider must use the Data Import API (authentication required) to transfer files from/to the Internet.

Backup/restore:

All data provider's workspaces are automatically backed-up with the file system by the ElasticFS each night.

Only administrators can restore the file system.

Database:

Availability:

The database is not automatically created when a new account is set up.A data provider must request by mail the creation of a dedicated database.

Database types:

The following types of database are supported:

- PostgreSQL 9.1
- PostgreSQL 9.1 + PostGIS 1.5
- PostgreSQL 9.1 + PostGIS 2.0

Currently, choosing a database type is definitive: It is not possible to change the type of database once it is created. Data provider that wants to migrate must manage and implement the migration process.

Accessibility:

All instances on the platform can access the database, so the data provider's database is shared and accessible by any application running on any instances.

By default, access is denied from the Internet. A data provider must request by mail authorization to access its dedicated database from the Internet.

Accessibility from application:

Applications access the data provider's database with the PostgreSQL driver available in the application language.

A data provider can get connection information on its database with the RESTful service provided by the ElasticDB API. However, password is not returned. Data providers must know the password of its dedicated technical user (and the password of the optional dedicated read-only technical user)

Maintenance:

A data provider can request for a temporary SSH access. Then the data provider can use the psql tool to connect to its dedicated database and manage it.

Of course, data providers that have request access to their database from the Internet can use any tools like pgAdmin to manage their database.

Transfer data from/to the database:

A data provider can use any tool to export/import data from/to its database.

If done inside the platform (e.g. through a SSH session), data are exported/imported from files located on the data provider's workspace. The data provider can use the Data Import API to transfer those files from/to the platform.

3.3. SECURITY

Platform access:

Each data provider has a dedicated account with specific credentials. These credentials provide the user with access to the portal (HTTP) and optionally to the platform (SSH).

By default, SSH access is denied. A data provider must request for a temporary SSH access on selected instances.

File system:

The workspace on file system is dedicated to the data provider.
The workspace's content (files and sub-directories) is owned by the data provider's account.

The data provider account has all privileges on its workspace, so a data provider can manage its workspace, including files or directory creation, modification and deletion, etc.

The data provider's workspace's is only accessible by the data provider's account and technical accounts ('igcservices' group).

Accessibility:

As all instances on the platform automatically mount the data providers' workspaces, any application running inside the platform can access a data provider's workspace as a local directory by (according to the permissions).

Accessibility from the Internet:

Data provider's workspaces are not exposed to the Internet. A data provider must use the Data Import API (authentication required) to transfer files from/to the Internet.

Database:

A technical user is automatically created with the database.

The default password for the dedicated technical user is given in the mail sent after the database has been created.

A data provider must use the dedicated technical user to manage its database.

The dedicated technical user has all privileges on the database's objects, so a data provider can manage its database, including schema, tables, etc.

A data provider must also use the dedicated technical user to manage other technical user privileges on its database's objects.

Data providers cannot create databases and users.

However, a data provider can create a read-only technical user with read-only privileges on its database with the RESTful service provided by the ElasticDB API (authentication required) A read-only user is useful to improve security in applications that must access the database in read-only.

The read-only technical user login and default password are the same.

It is recommended to change the default password of the technical user and read-only technical user as soon as possible.

Accessibility:

All instances on the platform can access the database, so the data provider's database is shared and accessible by any application running inside the platform.

By default, access is denied from the Internet. A data provider must request by mail authorization to access its dedicated database from the Internet.

4. HIGH-LEVEL SERVICES IN PILOT2

4.1. PLATFORM SUPERVISION AND MONITORING

The monitoring service provides a specific user friendly web interface that allow an administrator to access, query and analyze the behaviour of the platform at any point in time, even when the platform is not running. The tools is accessible from the Portal and the data are stored inside a database provided by the cloud provider.

The Figure 6 illustrates the monitoring interface. Here we are able to detect the following parts:

1. A list of indicators, from which the administrator can choose which one to display on the right panel. Some indicators are collected directly from the running instances (e.g. cpu usage), others from Amazon CloudWatch (e.g. average latency of an auto scalable group), others from REST APIs methods (e.g. the status indicators).
2. The chart representing the behaviour of the selected indicators, in the selected time period. On the x-axis is the time, on y axis is the sampled value of the indicator. Each service is represented by a specific colour
3. The chart legend, containing the different services as well as their colour inside the panel above. The administrator can choose to display or not the specific service.
4. The time period of the current chart. The administrator can use this form to retrieve historical data of a different time period.

The Figure 7 shows the alarms web interface. Here we are able to detect:

1. The indicator used by the alarm
2. The triggering threshold set up for the indicator
3. The icon for remove the current alarm
4. The icon for create a new alarm. It will move the administrator to a new page where he can choose the indicator, the threshold condition as well as the action to undertake.
5. A list of alarms raised in the selected period, with the description of the time they were raised, the service as well as the condition exceeded.
6. The time period the administrator can change

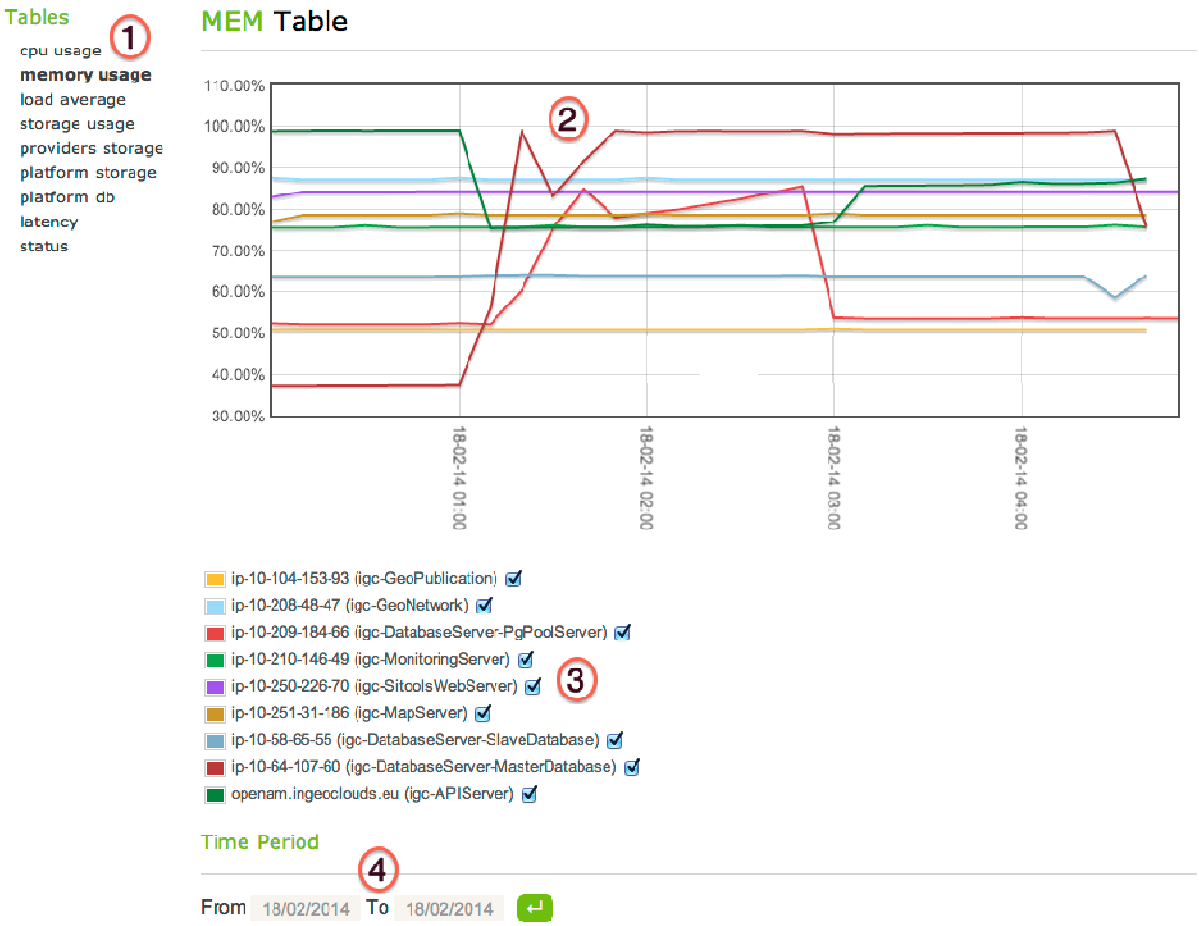


Figure 6: Monitoring web interface

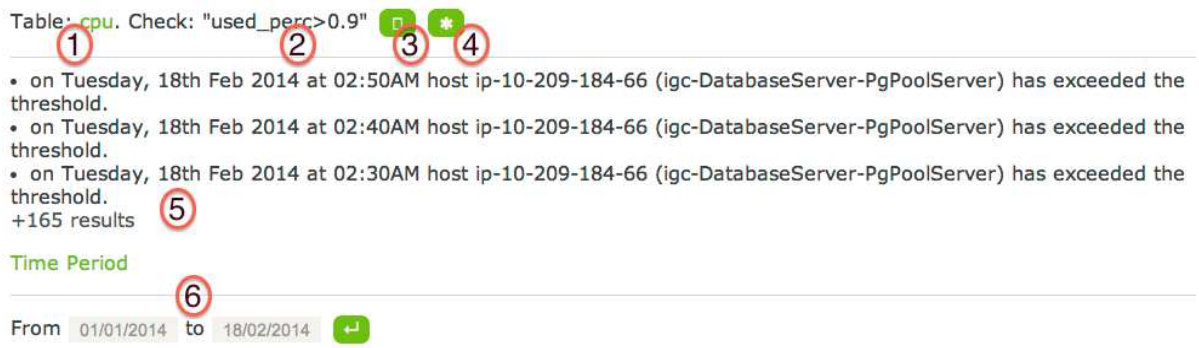


Figure 7: Alarms web interface

4.2. DATA IMPORT

The Data Import provides REST web services and protocols that allow data providers importing data from outside into their dedicated workspace inside the InGeoCloudS platform.

The Data Import is designed to provide the following services:

- **/data-import/fs**: the File Service provides methods to import files. Data providers use this service to access their dedicated workspace
- **/data-import/harvests**: the Harvest Service is in charge of managing the schedule of the harvesting tasks of the data provider.

Importing files:
The File Service provides the method **/data-import/fs/mediaFs** that returns a media link to the workspace or a sub-directory of the workspace. The media link is valid for a period of time. FTP or FTPS are the supported protocols.

Harvesting tasks:
The Harvest Service provides methods to schedule the execution of harvesting tasks. A harvesting task is a specific application that collects data located in the data providers' infrastructure and imports those data into the InGeoCloudS platform. Data providers use this service for registering, unregistering or modifying a harvesting task that will be triggered at a defined frequency.

4.2.1. OFF-THE-SHELF TOOLS AND FRAMEWORKS

Technical Asset	Version used	Inside InGeoCloudS
vsftpd	2.3.5	Data Management

4.2.2. KEY DESIGN AND IMPLEMENTATION ISSUES

The FTP and FTPS transfer are provided by the Very Secured FTP Daemon. Vsftpd was designed and implemented with security in mind. Security facilities implemented by vsftpd include Linux capabilities, chroot and secure coding techniques to avoid buffer overflow, Lightweight nature of vsftpd allows it to scale very efficiently. Vsftpd supports transfer of large data for hundreds of simultaneous users.

The Data Import mediasFs method drives the list of authorized users modifying the vsftp configuration in live. The connected user is chrooted to its dedicated workspace. In this way, a connected user cannot browse through the file system to see other data providers' files.

See [R5]for more details.

4.3. SMARTQUERIES & SMARTQUERIES AUTHORING TOOL

During General Assembly in Pisa in M17, a workshop session addressed the issue of integrating in InGeoCloudS simple but yet powerful means for data providers to directly exploit GSOM-based datasets without software development. These needs lead to the definition of the SmartQueries tool which aims at:

- Providing to any user a quick and simple way to query published datasets using a kind of "Frequently Asked Queries" library composed of queries that have been authored by experts.
- Letting users change some significant parameters of the queries through a web interface without particular knowledge of the underlying technology burden or data storage model.
- Formally describing the provided queries (the "SmartQueries") using web standards (Opensearch)
- Letting Data Providers the possibility of attaching some scientific knowledge / explanation about their published queries
- Letting Data Providers experiment with private queries in various declinations before publishing the most significant outputs about their data
- Offering a geographical vizualisation of query results thus leading to further exploitation (consolidation in table presentation, geographical selection of features, passing results to WPS, exporting resulting data towards INSPIRE compliant formats…)
- Being a showcase of how the comprehensive LinkedData API can be used by other Web applications

The result is a Web-based tool for data consumers called "SmartQueries" and available in the Portal as well as an Authoring Tool integrated in the Data Providers Toolkit:

SmartQueries
The queries will be used by users to perform search, view results (map and table view) and possibly download.

SmartQueries Authoring Tool
This tool helps data providers to define SmartQueries. The tool relies on the management of a set of SPARQL queries templates that can be customized into various instances. The authoring tool is integrated in the Data Providers toolkit in the Portal.

4.3.1. DEFINING SMARTQUERIES

The authoring tool exposes a user interface for the edition of SPARQL code, user-friendly selection of filters and parameters. On the server side, generated instances are described using Opensearch description files (see 4.3.2below)

```
select ?location ?movement_type ?slope_tilting ?lithology ?rainfall
?year ?municipality ?altitude ?landcover from <http://ekbaa.data>
where {
?landslide a sci:S32_Landslide;
           crm:P1_is_identified_by ?id;
           crm:P2_has_type ?mov_t;
           sci:O25_generated_feature ?feat;
           crm:P4_has_time-span ?year;
           crm:P7_took_place_at ?place;
           sci:O17_has_dimension ?d.
filter(isNumeric(?year)).
filter(regex(?id, "Object_Id")).
?mov_t crm:P3_has_note ?movement_type.
?d crm:P2_has_type ?sl_til.
?sl_til crm:P3_has_note ?slope_tilting.
filter(regex(?slope_tilting, "<SLOPE_TILT>")).
?curv sci:O26_is_section_of ?feat;
      sci:O17_has_dimension ?lith.
?lith crm:P3_has_note ?lithology.
?event sci:O21_triggers ?landslide;
       sci:O17_has_dimension ?d1;
       crm:P2_has_type ?m_src.
?d1 crm:P90_has_value ?rainfall.
?place crm:P87_is_identified_by ?alt;
       geo:hasGeometry ?point;
       crm:P89_falls_within ?sett;
       crm:P3_has_note ?landcover.
?point geo:asWKT ?location.
filter(REGEX(?alt, "Alt")).
?alt sci:O20_has_value ?altitude.
?sett crm:P1_is_identified_by ?municipality.
filter(regex(?municipality, "Municipality/Name/")).
filter(regex(?landcover, "<LABEL3>")).
?impl crm:P3_has_note ?implication.
?m_src crm:P3_has_note ?mov_source.
#####
filter(?year > {year_min_val} && ?year < {year_max_val}).
filter(?altitude > {altitude_min_val} && ?altitude <
{altitude_max_val}).
filter(regex(?landcover, {landcover_val})).
filter(regex(?movement_type, {movement_type_val})).
filter(regex(?mov_source, {mov_source_val})).
filter(regex(?lithology, {lith_val})).
filter(regex(?implication, {impl_val})).
filter(regex(?slope_tilting, {slope_tilt_value})).
filter(regex(?municipality, {municipality_value})).
}
```

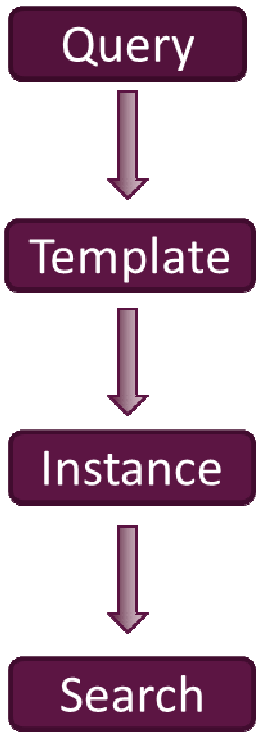


Figure 8: Example of SPARQL query and principles for generating "Smart" queries with the tool.

The Opensearch/SPARQL support that currently exists in the InGeoClouds portal has the following functionalities:

- Server side
  - Define a SPARQL query on a specific SPARQL endpoint with simple filter definition
  - Query with user parameters
  - Opensearch.xml definition

- Client side

- Query using a simple form
- Display results on a map and a grid
- Save query parameters in personal workspace (only for connected users)

4.3.2. WHAT IS OPENSEARCH

The Smart Queries module uses the Opensearch standard for queries. Opensearch is pretty simple; it is a collection of simple formats for sharing the search results. The opensearch description document is a simple XML file which describes the way to make queries including query parameters, type of document retrieved and other metadata about the search engine. More on http://www.opensearch.org/Home.

Opensearch description documents can be extended with foreign markup, there are already some extensions defined for time or geo queries for instance.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
 <OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/">
   <ShortName>Web Search</ShortName>
   <Description>Use Example.com to search the Web.</Description>
   <Tags>example web</Tags>
   <Contact>admin@example.com</Contact>
   <Url type="application/rss+xml"
template="http://example.com/?q={searchTerms}&amp;pw={startPage?}&amp;format=rss"/>
 </OpenSearchDescription>
```

4.3.3. How OpenSearch is used in the Smart queries module?

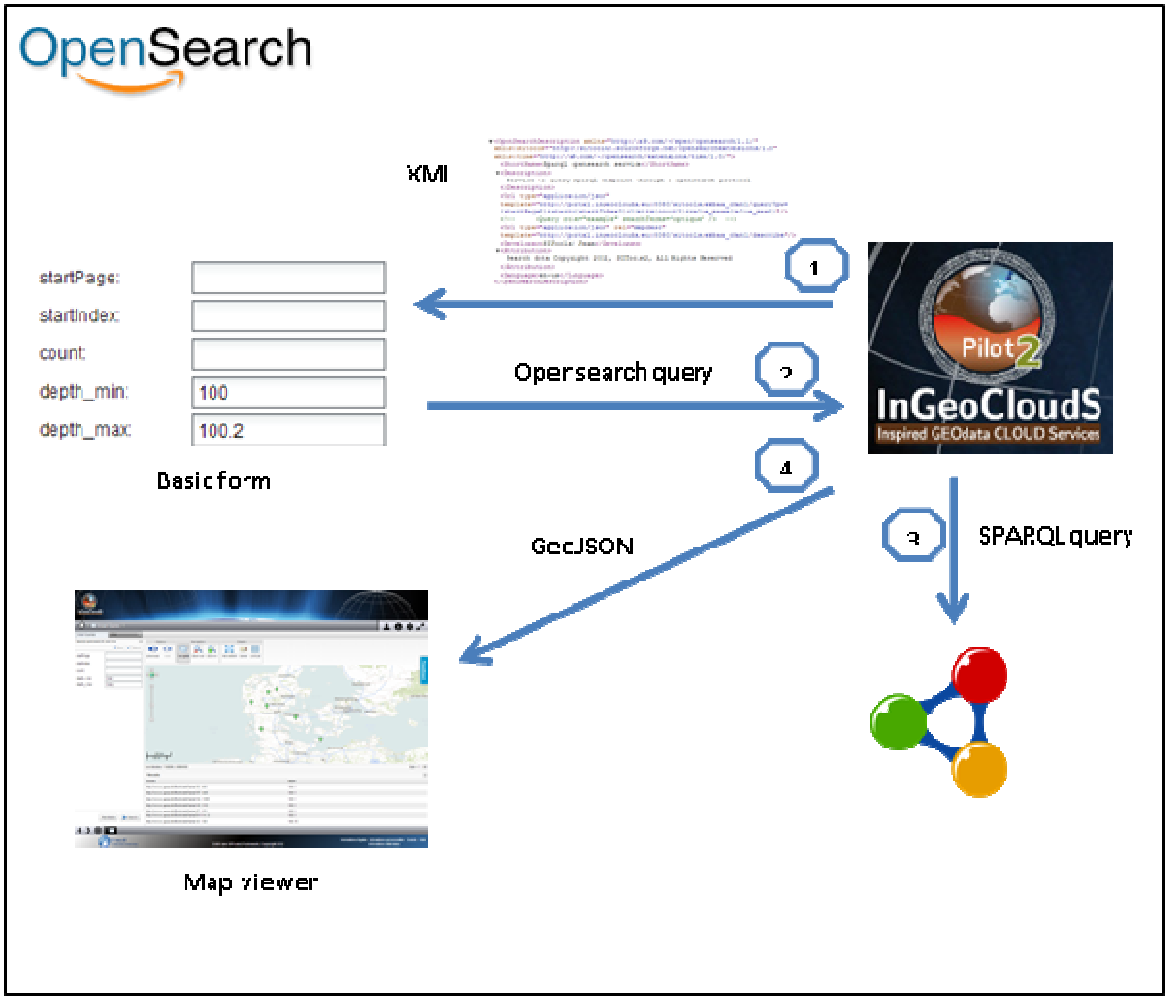


Figure 9: Example of SPARQL query and principles for generating "Smart" queries with the tool.

The smart queries module asks for the opensearch.xml file. That file contains the URL template of the query with the parameters. The module creates a simple form from those parameters and the user can enter unformatted values. The SITools2 server transforms that request into SPARQL and returns the result as GeoJSON which is displayed on a map.

Some evolutions are envisaged for a better user-experience and more flexibility for generating forms with different pre-defined constraints on the input parameters (type of the SPARQL field, list of possible values (enum or min/max values). Thus, the scenario would be:
The smart queries module asks for the opensearch.xml file. That file contains the url template of the query with the parameters. Then it asks for a JSON file containing the description of each parameter. For each of them is defined:

- The name
- A description
- The type (int, string, date, enumeration)
- For the enumeration type, all the accepted values
- For int type, the maximum value

This list can be enriched in the future according to the customization needs we might need for user queries.

The Smart queries module creates a complex form from those parameters and the user can choose from enumerated values, choose a date from a date picker or check that the values are correct.
The SITools2 server transforms that request into SPARQL, calls the LD API methods and returns the result as GeoJSON which is displayed on a map and on a grid with sortable columns that also can be ordered at will.
Persistency mechanisms allow data providers to keep generated Queries instances in a private space (e.g. for testing, refining and designing a pool of useful queries) but also to publish them on the portal whenever wished. Popularity indicators give a trace of how often the published queries are used by visitors/peer users.

4.3.4. IMPLEMENTATION

SITools2's architecture has been made to be pluggable, so we used only plugins to implements the SmartQueries applications.

SITools2 is REST based and exposes representations through resources. A resource is attached to an application which is attached to a component.
The component can be viewed as the server.
The application is state full (can be started or stopped) and only one instance of each application exists.
On each call, depending on the url and the method asked, a resource is instantiated and it will produce a representation sent back to the client.

SmartQueries Authoring Tool
The following operations are available:

URL	Method	Action
/templates	GET	Get all templates
/templates	POST	Create a new template
/templates/{sparqlTemplateId}	GET	Get a specific template
/templates/{sparqlTemplateId}	PUT	Modify a specific template
/templates/{sparqlTemplateId}	DELETE	Delete a specific template
/instances	GET	Get all instances
/instances	POST	Create a new instance
/instances/{sparqlInstanceId}	GET	Get a specific instance
/instances/{sparqlInstanceId}	PUT	Modify a specific instance
/instances/{sparqlInstanceId}	DELETE	Delete a specific instance
/instances/{sparqlInstanceId}/notify	PUT	Internal resource
/instances/{sparqlInstanceId}/publish	PUT	Publish a instance
/instances/{sparqlInstanceId}/unpublish	PUT	Unpublish a instance

SmartQueries
The SmartQueries application is in charge of many features including:

- Producing JSON representation of instances which have been published
- Executing queries with OpenSearch API and produce GeoJSON
- Executing queries with OpenSearch API and produce basic representation (JSON, XML…)
- Executing queries with OpenSearch API and produce advanced representation (GML, KML…)
- Producing OpenSearch protocol definition (opensearch.xml)
- Producing JSON OpenSearch protocol definition (describe)
- Updating number of execution of an instance.

Each of those functionalities is served by a specific resource.

The following steps are needed to execute a Sparql query:

1. The user performs a query using the SmartQueries tool on the portal, filling some parameters value.
2. The server creates the SPARQL query from the SPARQL template and the HTTP parameters.
3. Then the LdAPI (ld/ldquery) is called with this query. The API returns some XML containing the results of the SPARQL query.
4. This XML is sent to the GeoLdAPI (geold/geoldtransform), which transforms the XML into GeoJSON. The GeoJSON is sent back to the client.

The same process is used to produce KML, GML or ShapeFile through the SparqlExportResource

In order to retrieve some XML or JSON directly, the process stops after the ldAPI call is made.

The SPARQL query is created from a template string. If this template contains a $Filter tag, this tag is replaced with some SPARQL filter statements, depending on the filters defined Instance model. 3 types of filter are available currently, but this is possible to add new filters easily.
A filter is basically a Java class (that inherits from SparqlFilter class).

4.4. INSPIRE DATA EXPORTS

FORTH has provided a mapping from GSOM to INSPIRE which enables the exporting of the providers' data into INSPIRE-compliant format both conceptually and texhnically. The mapping is not complete as there exist notions from GSOM (mapping to particular theme-related information) which are not covered by INSPIRE. To this end, not all data providers' data can be actually exported to INSPIRE. This was actually anticipated since from the beginning of the project we have noted that INSPIRE has specific limitations on how information is recorded (you can refer to Deliverable 2.2 for more explanations on that).
Two different methods have been realized in the Linked Data Management API which allow exporting provider's data into XML-based INSPIRE-compliant specifications:

- *inspire_export*: this method takes as input: one or more themes, a set of filtering constraints, a limit on the amount of returned results as well another information which indicates whether data on which the filters do not apply will actually be returned, and returns the theme-related data which satisfy the filtering and amount of results constraints. The actual realization mechanism employed is the one where each theme is mapped to a skeleton SPARQL query which is enriched with the constraint and result limit information (when the latter is provided) in order to draw the respective theme-related information. Then, all the latter information is transformed into INSPIRE according to the GSOM-to-INSPIRE mapping developed.
- *inspire_query_export*: this method takes as input a SPARQL query and an optional limit on the amount of returned results and returns the related data in INSPIRE-compliant form. It is up to the posed query which data will actually be exported based also on the fact that the GSOM-to-INSPIRE mapping is incomplete. The actual realization mechanism employed is to map the SPARQL query variables to mapped notions of GSOM (which thus correspond to INSPIRE notions) and then include only the returned information from them in order to apply the GSOM-to-INSPIRE transformation based on the mapping developed.

The first method is applicable in cases where the data provider is not fully familiar with SPARQL but only knows how to specify SPARQL filtering constraints. It is also more suitable in cases where all the desired information to be exported can be deduced from the themes themselves (actually the underlying skeleton queries used to drawn the respective information) and the constraints posed thus leading to the portion of providers' data (related to the desired themes), which can all be exported to INSPIRE. On the other hand, the second method gives to the user the ability and freedom to precise exactly the data that he/she desires to export with the obvious disadvantage that it is not certain whether the desired portion of data will eventually be transformed to INSPIRE due to the missing mappings.
Apart from these two methods, there is also another method, which can further assist in their proper exploitation. This method is called *theme_query_info* and can be used to obtain important information,

such as the underlying skeleton SPARQL query which is used to draw data from a particular theme, the related data providers and their respective RDF graphs stored mapping to the theme-related information as well as information about which SPARQL variables can be constrained. The latter information covers the semantics of the SPARQL variables, their value type as well as the range of values that can obtain (which is sometimes specific to each data provider - so in this case all possible ranges of values are returned). Concerning the INSPIRE mapping, as INSPIRE is a standard which undergoes various modifications, it is expected that the respective INSPIRE meta-models and XSDs will evolve. This will mean that: (a) the GSOM-to-INSPIRE mapping can become even more complete when the respective missing, theme-related information is eventually captured by INSPIRE and (b) changes to the GSOM-to-INSPIRE mapping might need to be introduced. We need to have a mechanism to efficiently manage these changes and to update the GSOM-to-INSPIRE mapping respectively so that the providers' data are exported according to the latest and possibly even more complete INSPIRE version.

4.5. INSPIRE TECHNICAL SERVICES

The design and implementation can be reported

- Data push to the cloud
- Data metadata
- Service set up

4.5.1. PUSHING DATA TO THE CLOUD

Three different type of feeds are allowed:

- Files such as Tab, MIF/MID, Shape files
- Direct link to the database
- WebServices: Web Feature Services

The mapping to the relevant data model is ‘hard coded’ for the time being. For example the BRGM Water Level Measurement was made as such:

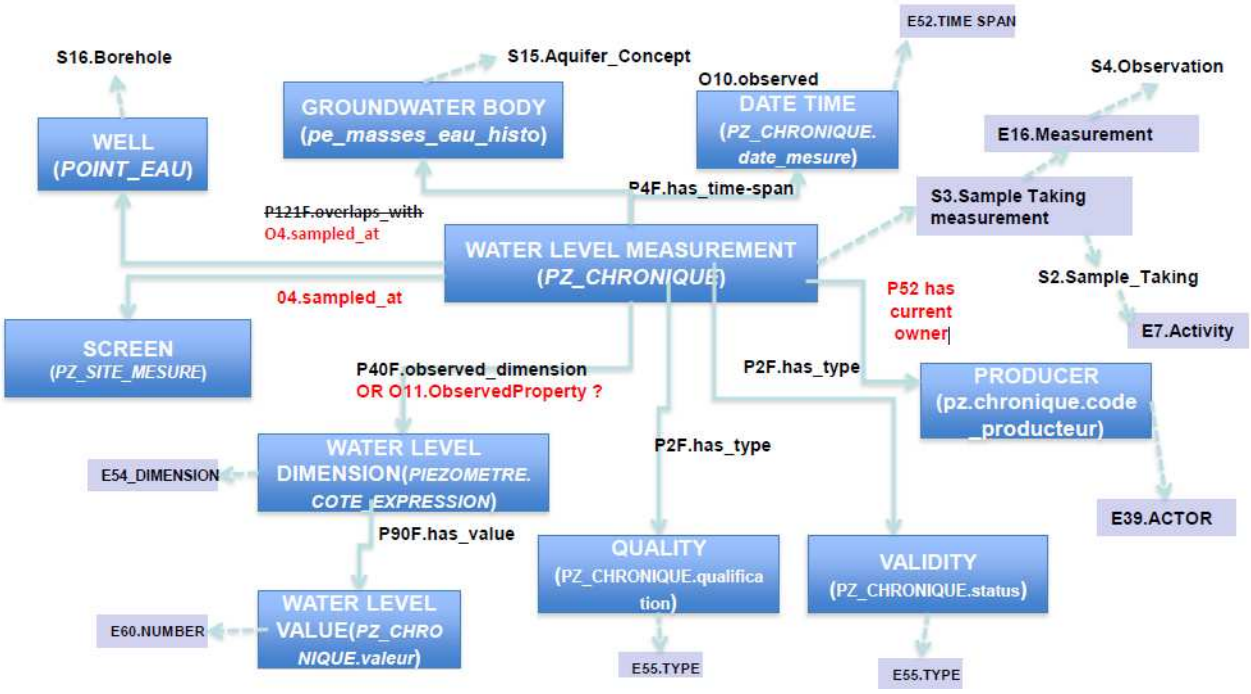


Figure 10: Mapping of BRGM water data to relevant model

4.5.2. DATA METADATA

The generation of the data metadata (DMD) was made according to Inspire implementig rules.
The DMD are ISO 19115/139 compliant, and pre-filled by the GeoPublication module.

Genetwork is an included is a catalogue service that generates the Metadata form the Geopublication data input.

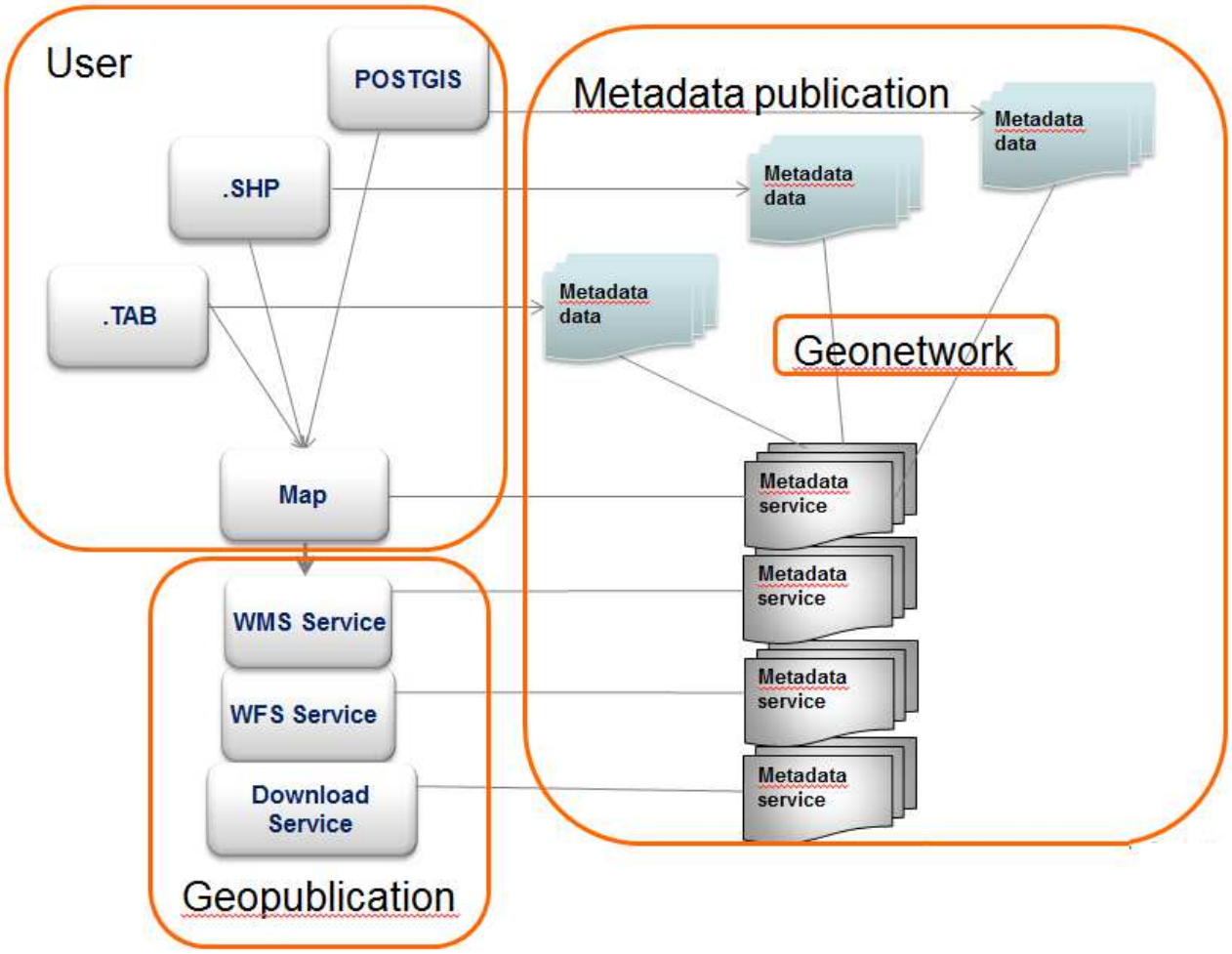


Figure 11: Map and catalogue interaction

4.5.3. SERVICES SET UP

All generated services URLs are public. The service metadata generation is almost automatic.

The Discovery service is a CSW 2.0.2 provided by Geonetwork and automatically fed by data metadata. The View service .is a WMS 1.3 provided by MapServer. For the publication module the user can choose the projection system, the symbology definition…

For example, the metadata generation interface is:

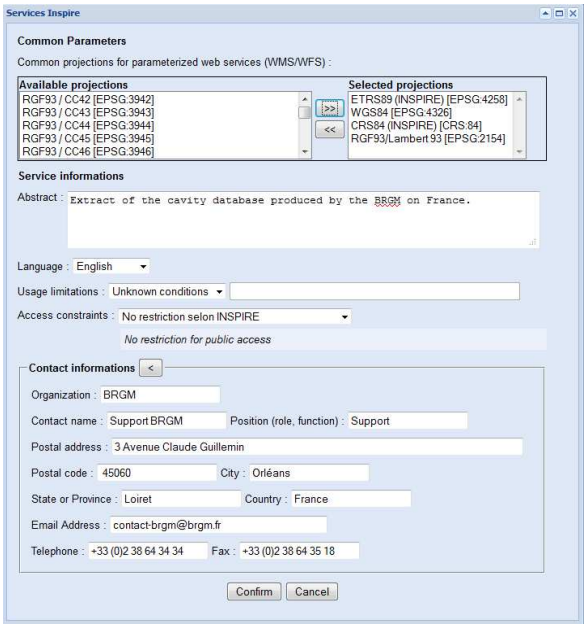


Figure 12: Metadata generation interface

One more example with the Symbology interface:

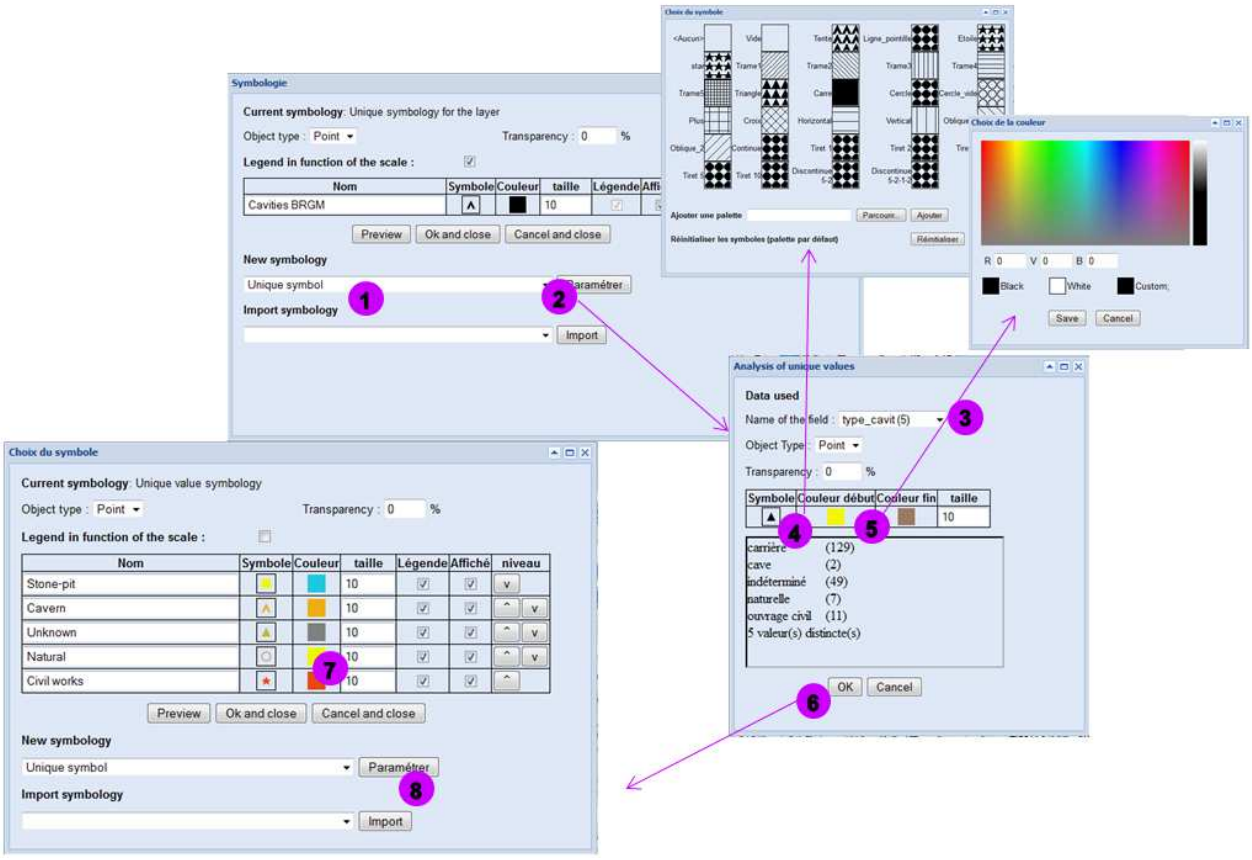


Figure 13: Symbology interface

The pre-defined dataset download provides ATOM, Shpare, Tab.(GML application schema compliant to come)
The direct access download is WFS 1.1.

4.6. ACCOUNTING AND PROVISIONING

The accounting module provides a RESTful APIs that allows both platform administrator as well as data providers to retrieve a detailed per-data-provider and per service split of costs. It provides also the overall usage indicators used to split the costs of the shared services among the different data providers.

The accounting service is designed to provide the following services:

- **/accounting/costsAll**: it can be used only by the administrator and provides the detailed split of costs of all the data providers inside the platform.
- **/accounting/costsProvider**: it can be used by any data provider and provides its own detailed split of costs
- **/accounting/usageIndicatorsAll**: it can be used only by the administrator and provides the usage indicators parameter computed for all the data providers inside the platform.
- **/accounting/ usageIndicatorsProvider**: it can be used by any data provider and provides its own usage indicator.

In addition to the RESTful APIs, the accounting service provides also a specific user friendly web interface that allows an administrator to access, query and analyze the split of costs computed by the service at any point in time, even when the platform is not running. The tools is accessible from the Portal and the data are stored inside a database provided by the cloud provider.

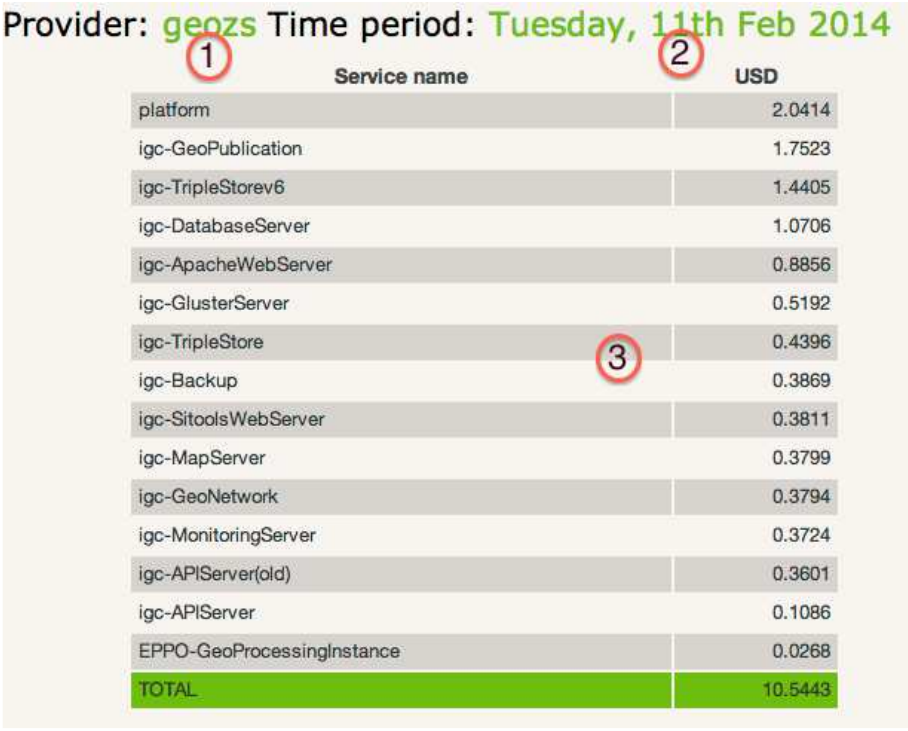
The Figure 14 illustrates the accounting interface. Here we are able to detect the following parts:

1. The data to show on the right panel. The administrator can choose the per service split of costs or the per data providers split of costs, both on a daily basis as well as on a monthly basis.
2. The chart representing the split of costs. Each column is composed by different cells (providers or service, depending from the chosen chart).
3. By moving with the mouse on top of a cell, it display on overlay the specific cost. By clicking on top of it, a new page is opened with some additional information about the costs of the specific data provider.
4. The chart legend with the different services/providers and their colours
5. The time period form

The Figure 15 illustrates the detailed view of the accounting interface. Here we are able to detect:

1. The data provider selected
2. The time period selected
3. The costs of the data provider in the specified time period, on a service basis, ordered by cost descending (with the total on the bottom).

Figure 14: Accounting web interface



Service name	USD
platform	2.0414
igc-GeoPublication	1.7523
igc-TripleStorev6	1.4405
igc-DatabaseServer	1.0706
igc-ApacheWebServer	0.8856
igc-GlusterServer	0.5192
igc-TripleStore	0.4396
igc-Backup	0.3869
igc-SitoolsWebServer	0.3811
igc-MapServer	0.3799
igc-GeoNetwork	0.3794
igc-MonitoringServer	0.3724
igc-APIServer(old)	0.3601
igc-APIServer	0.1086
EPPO-GeoProcessingInstance	0.0268
TOTAL	10.5443

Figure 15: Accounting web interface, detailed view

5. DEVELOPMENT AND TESTS OF INGEOCLOUDS PLATFORM

5.1. TOOLS AND DEVELOPMENT RULES

The REST API defined and implemented by all the partners is accessible as web Restful services provided by java web applications.

5.1.1. OFF-THE-SHELF TOOLS AND FRAMEWORKS

Technical Asset	Version used	Inside InGeoCloudS
JavaSE SDK	1.7.0_11	All API
Jersey	1.17.1	All API
Spring framework	3.2.2.RELEASE	All API

RESTful services implementation:

Jersey:

RESTful services implementation follows the Java API for RESTful Web Service (JAX-RS), a Java API that provides support in creating web services according to the Representational State Transfer (REST) architectural pattern.

Jersey is the reference implementation of the JAX-RS standard (JSR-311)..

Spring framework:

The Spring framework provides a lot of features to integrate technical components (MVC, data access, security, etc.) and business code.

Injection mechanism is very flexible, and allows developers to control the type of life-cycle of each bean. The standard JSR-330 annotations are supported.

The Spring Security framework allows controlling access at different levels in the application: filtering the URL access and securing the method access.
The standard JSR-250 annotations are supported, which ensures business code does not dependent of Spring Security framework.

Documentation generation:

Enunciate is a tool that allows generating a powerful documentation for Web services. Documentation is embedded in the final web application so it is accessible online.

Documentation generation is part of the Maven build process. Because the documentation generation takes some time, it makes the build process slower. That's why by default, the documentation generation is disabled. The property doc.generation.skip allows enabling or disabling documentation generation.

Build process:

The Java web applications are built using Maven. The organization of the projects is based on the notion of multi-module enterprise projects in Maven, but applied in a pragmatic way.

The build process is designed to address a specific running environment: Amazon or cloud mockup.

Cloud mockup is a virtual cloud environment is a useful for developers: it allows executing the API applications on host or a virtual machine that is not an EC2 instance.

5.1.2. KEY DESIGN AND IMPLEMENTATION ISSUES

Multiple applications:

Multi-module Maven projects enhance the loose coupling between the different APIs:

- It avoids conflicts between constraints of each API:
  - Ensuring stability and reliability of each API, independently of constraints of the other APIs,
  - Implementing security for each API, independently of constraints of the other APIs.
- It facilitates developer life:
  - Decreasing risk of errors while merging,
  - Improving compilation time,
  - Avoiding conflicts with versions of libraries,
  - Encouraging test coding,
- It enhances deployment flexibility:
  - allowing deployment of all APIs on a single tomcat server for development environment,
  - allowing deployment of APIs on different tomcat servers for production environment,
  - allowing deployment, debugging and testing on different target than Amazon EC2 instance (when possible),
  - allowing creating dedicated deployment scripts for development and production environments,
  - enhancing the cost control.

Modularity:

All APIs are defined and implemented in the same way: definition and implementation are separated:

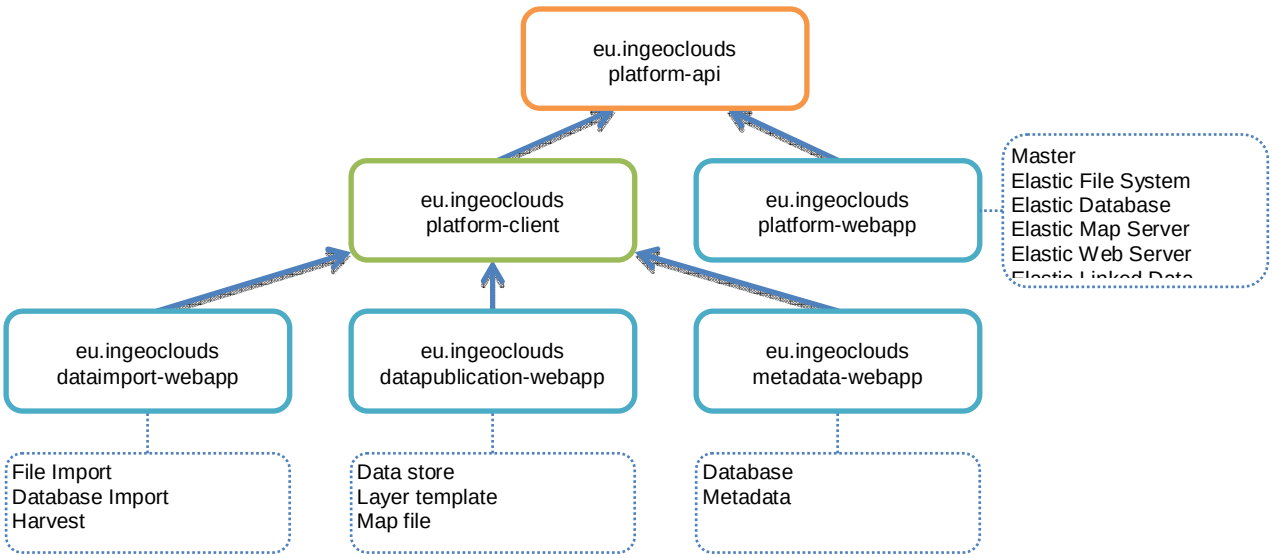
- Interfaces in the eu.ingeoclouds.api package define the API,
- Classes in the eu.ingeoclouds.domain package define the related domain objects,
- Classes in the eu.ingeoclouds.impl package implement the API.

Each Maven projects defined for an application (see above) can be split (or not) in multiple Maven projects:

- `<project>-api` builds the REST API definition library (jar), including the following packages:
  - `eu.ingeoclouds.api` with the interfaces,
  - `eu.ingeoclouds.domain` with the domain objects
- `<project>-client` builds the client REST API library (jar) with the implementation of the client part of the REST API, including the following packages:
  - `eu.ingeoclouds.client` with the classes that allow clients calling the RESTful services.
- `<project>-webapp` builds the application (war) with the implementation and documentation of the REST API, including the following packages:
  - `eu.ingeoclouds.impl` with the classes that implement the REST APIs.

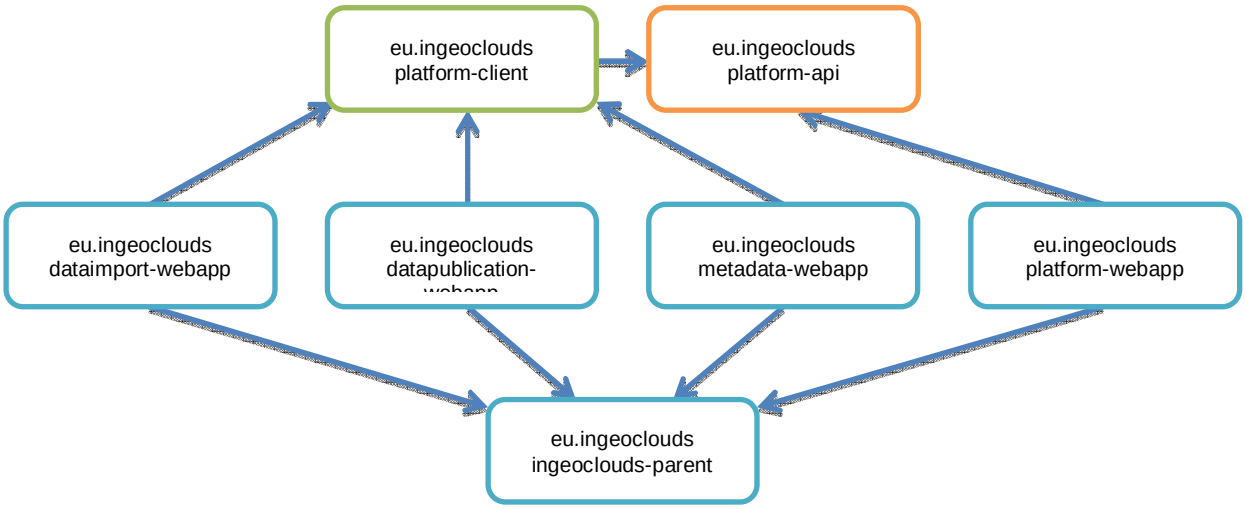
However, we tried and remained pragmatic. Juggle three Maven projects to define and implement an API is more complicated than using a single Maven project for developers. If no project depends on a project `<project>-client`, this project is unnecessary. This means also the project `<project>-api` can be merged with project `<project>-webapp`. So, in this case, it is not necessary to split the Maven project defined for the application.

The only APIs that are called by the other ones are provided by the `platform-webapp` application. So splitting the Maven project for the `platform-webapp` application is necessary. However, all other API web applications can be built using a single Maven project.



Common code:
Generally, when a project is composed of multiple applications, there is always some piece of code that is common. In order to avoid having code duplicated in multiple Maven projects, the common code is moved in one project on which other project depends.

Final dependencies are as follow:



5.2. INTEGRATION AND DEPLOYEMENT TESTS

5.2.1. OBJECTIVES

Integration and Deployment tests for main platform components aimed at dimensioning correctly the resources provisioned and assessing their capability in providing scalable and elastic services. We report in the following about most significant tests that assessed good practices for cloud design for benefiting from high-availability and elasticity.

5.2.2. MIDDLEWARE – ELASTICDB

Different scalability and stress tests scenarios have been defined and classified into three categories:

- Scalability of user connections
- Connection of simultaneous users: how many simultaneous sessions are supported without error/blocking
- Downgrading of user connections: how do CPU, memory and session occupations decrease if user connection requests decrease.

The objective has mainly been to

- Define most appropriate type of instances for Pgpool, PostGreSQL Master, slave
- Define most appropriate settings of PgPool and Postgres parameters.
- Identify any indicators and threshold for pertinent monitoring and supervision.

Each scenario has been based on data consultation both on the Geonetwork instance and on GEUS Postgres/PostGis Database (e.g. Simulate the display of a particular Geo- zone with 15000 to 30000 results returned). Select queries are the most common type of load for IngeocloudS; no scenario with insert, update and delete data have been extensively tested. We took as hypothesis that insert or update operations would be realized occasionally or planed during low occupancy periods by data providers.
The tool selected for designing and implementing test suites and scenarios has been Tsung[2]. Tsung is free software and it is designed to stress an application or technical component like HTTP, WebDAV, SOAP, PostgreSQL, MySQL, LDAP and Jabber/XMPP servers. Tsung allows capturing specific scenario (through an HTTP application by listening on a port). After that, Tsung could execute this scenario by implementing scalability of connections or by the simulation of several users connected simultaneously.

For each scenario, two or three types of shoots are realized. Each shoot represents different activities of consultation (for example: in the case of pollution of groundwater in a country)

- Normal profile (e.g. 4 users connect every second, execute a complex query and quit)
- Mean/average profile (e.g.5 users)
- High profile (e.g.10 users connect per second)

Tests have been conducted both on a local cloud (based on OpenStack) and on Amazon Cloud.
Main conclusions lead to the choice of Medium type instances for Pgpool and Master Postgres/PostGiS (it allows for example to support 800 client connections on-line on Pgpool server). Even if all resources are not used (which is our case up to now in Pilot trials), it is necessary to have free memory and capacity of CPU for processing other requests and maintenance batches (like rebuilding indexes, vaccum, data import…).Multiplying slave instances in most complex scenarios defined only brought minor evolution of execution time but permitted, as expected, to support more simultaneous users connected.
Comprehensive documentation about the implementation of tests and their main results is to be found in [R12] technical note.

5.2.3. MIDDLEWARE – ELASTICFS

Two set of experiments have been conducted both on a set of local machines and on Amazon Cloud in order to:

- choose the best GlusterFS configurations to use (replication vs distribution vs striping)
- assess the GlusterFS scalability

The tool selected for designing and implementing test suites has been IOzone (http://www.iozone.org) file system benchmarking tool. We used this tool to measure the throughput of each specific configuration. IOzone can generate and measure a variety of file system operations also exploiting multiple clients

[2] http://tsung.erlang-projects.org/

simultaneously accessing the file system. We tried to remove as much as possible any caching effect from our experiments. This makes it a stronger test, but we must point out that caching is very likely and desirable when the InGeoCLOUDS applications are actually running since it strongly improves the performance of the file system. We tested all the three configuration mechanisms provided by GlusterFS.

The first set of experiment showed us that the best results were achieved with the *distributed* setting, i.e., where files are spread across the available servers. We did not measure performance improvements with the replication mechanism as well as with the partitioning, with the latter that on the other hand increased significantly the amount of storage space employed. Of course replication should be used for improving data reliability.

The second set of experiment was consequently conducted using the distribution configuration, in order to test how GlusterFS was able to scale. We varied the number of GlusterFS server from 1 to 8, with the stress test tool that slowly increase the number of clients requesting for a file operation during the time of the test. The results confirm the goodness of GlusterFS. The write performance scales almost linearly while the read performance are also better.

Main conclusion was to adopt the distribution configuration for GlusterFS, and to switch to a replicated configuration (which uses two times the number of the instances of the distribution configuration) when the reliability starts playing a key role inside the platform. Secondly, to adopt a small instance type because of the linear scalability showed by GlusterFS (so that we can also reduce the costs in the meanwhile).

A detailed description of the test made with GlusterFS can be found in D3.2 [R5].

5.2.4. DATA MANAGEMENT

FORTH has performed stress testing in order to assess the performance of the Linked Data Management API when an increasing number of concurrent clients is issuing a specific amount of different types of SPARQL queries. For particular query types – especially queries that are materialized as a series of joins, the performance was getting worse but it remained at acceptable limits. For other types of queries, the performance was lower and remained inside the acceptable limits. To this end, more instances were included as well as a load balancer in order to cater for the increased user query load. The subsequent stress tests revealed that when one instance was actually maintained, there was the danger that the performance again goes under the performance limits. However, by providing scale-up conditions on the load-balancer, a new instance was deployed on time and the performance was able to rise up again and stay at an appropriate level until most of the concurrent requests were satisfied, where again the performance was starting to rise.
Thus, it was concluded, after performing various stress tests with the load balancer and the different types of queries that particular performances limit should be in place in order to increase/decrease the number of instances. This limit concerning the CPU usage should not be over 65%. It was also deducted that the initial amount of instances is better to be one and then grow based on whether the CPU limit is reached. Moreover, it was decided not to scale-down immediately after scale-up occurs (which seems to be the normal case if we consider that we can have bursts of client requests at particular time points) as this makes no sense and will lead to increased costs when client bursts occur again inside the time limit of one hour.
Concerning the integration, we can report that we made integration efforts in order to ensure that our services will be part of the development and especially of the production platforms, meaning that their start-up/shutdown will be facilitated as the rest of the platform. Integration work was also performed for the realization of the SMART Query Services where the major issue there concerning our part was the exploitation of the *ldquery* method of the Linked Data Management API which was ensured through the compliance of this method according to the requirements of the SMART Query application.

5.2.5. PORTAL

A stress test was performed on the portal service with the objective to assess how the elasticity offered by the Elastic Compute module works (and on the underlying level, the Amazon Load Balancer and Elastic Group). It was chosen the Portal service for this kind of test because of its stateless nature (in the first test scenario the sessions were disabled).

A first test was realized by putting an ad hoc PHP script on the public area of the apache web server. The script was designed to perform some random calculations for a given period of time (so not sleeping, which could falsify the test). On the client, we used a traditional tool for stress test web servers, tool configured to increment the number of requests per second made to the server in a linear way. The test spend almost one our to finish.

The results obtained are shown in Figure 16. The dotted line here represent the threshold configured for raising the alarm with action to add a new instance to the pool of servers. It's clear to note how the average load of the elastic group drop off when a new instance is added to the elastic group, and how with an higher number of servers in the pool the slope of the average load decrease, with the meaning that the system is able to better sustain peaks of requests.

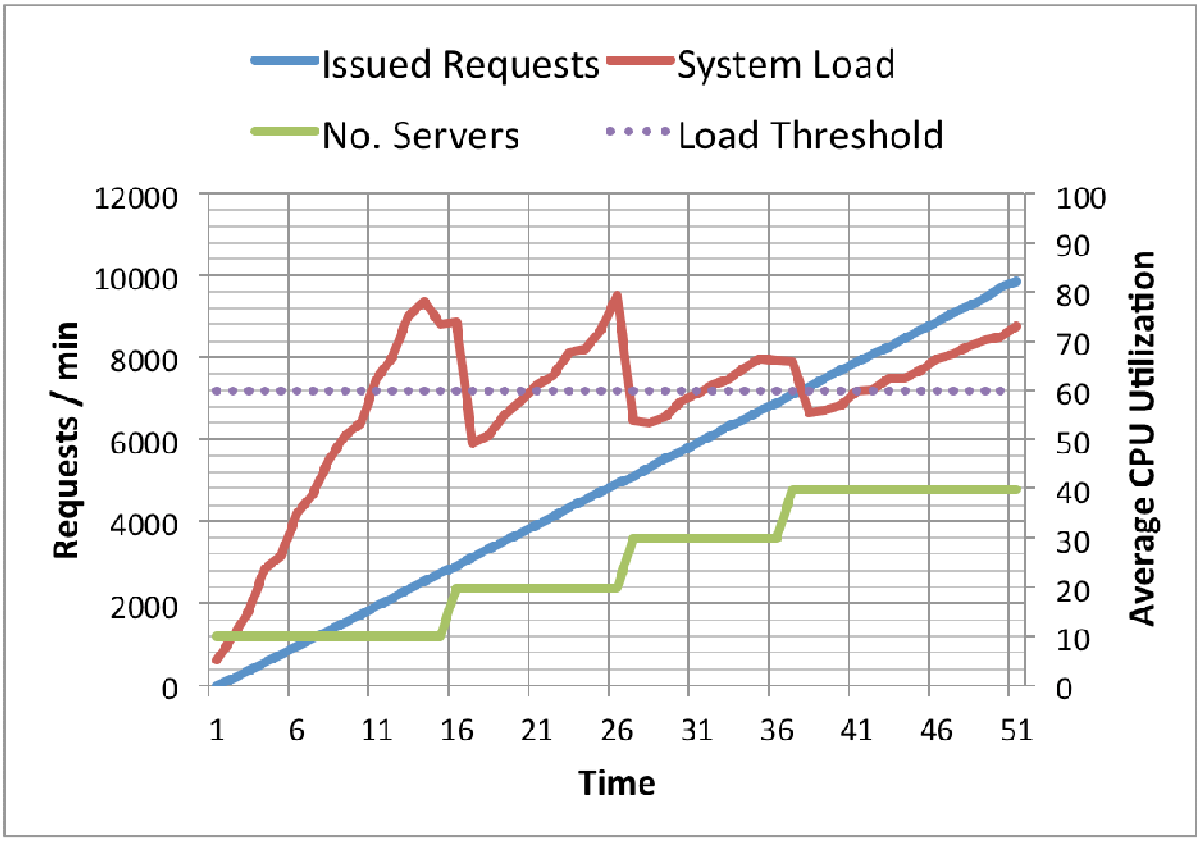


Figure 16: Testing web server layer

A second test was made to check if session affinity was correctly implemented inside InGeoCloudS. We enabled the session on the pool of web servers (the test was made with 4 servers) and created a new PHP script which this time uses the session. Then by manually calling this script we were able to understand if

the session affinity was working or not. In fact we were serving all the time by the same server, randomly chosen at the first request.

Thus we were able to conclude that the session affinity was properly working and that our stress test validated the implementation of the elasticity inside the Elastic Compute module.

6. CONCLUSION

In this document we gave an overview of the most recent developments and projects practices. Some contents of this deliverable will be used for feeding the user documentation on the wiki and the open source software project that is being put in place for giving access to a wider community of developers.

In Period3 further corrective maintenance actions will be necessary (undertaken in WP3 and WP4 activities). Section 2.5.3 listed some limitations that we might overcome yet at the architecture deployment level. Additionally, some further wishes for evolutions will likely appear through the use of the Pilot by (new) data providers and preliminary WP5 results. Some examples are listed below:

- A web-based graphical interface to manage Data Import API (e.g. with visual kind of drag&drop of files from local environment to the cloud).
- Possibility of visualizing several SmartQueries results on the same and unique map view for facilitating correlation and discovery of linked data.
- Easier access to data management facilities of the SITools framework (e.g. meta-catalogue et multi-channels harvesting, customised visualisation of datasets...), which have not been exploited so far.

Some other more technical issues that we might contemplate in Period#3:

- Upgrading OpenAM in order to accommodate for OAuth 2.0 and OpenID Connect Authentication technologies that are supported by latest releases
- Evolution of Smart Queries authoring tools with tighter integration of Openseach capabilities.
- More facilities for ***analytics***: not only at the highest level; at the moment, the ”popularity” measurements are limited to the 1st level projects (UCs) in the portal”. In order to allow data providers to define and implement more detailed analytics procedures for their services, e.g. usage and popularity of WMS, some Piwik tools configuration/plugin developments might be decided.

**** End of the document ****