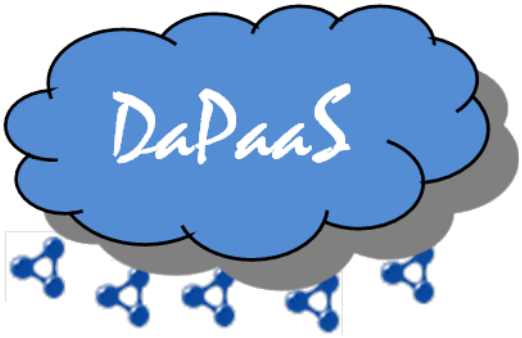
Small or medium-scale focused research project (STREP)

ICT SME-DCA Call 2013
FP7-ICT-2013-SME-DCA

Data Publishing through the Cloud: A Data- and Platform-as-a-Service Approach to Efficient Open Data Publication and Consumption

DaPaaS



Deliverable D2.1:

Open PaaS requirements, design & architecture specification

Date:	30.01.2014
Author(s):	Brian Elvesæter, Dumitru Roman, Martin Fagereng Johansen, Arne J. Berre, Marin Dimitrov, and Alex Simov
Dissemination level:	PU
WP:	WP2
Version:	1.0

Document metadata

Quality assurors and contributors

Quality assuror(s)	Bill Roberts, Rick Moynihan, Amanda Smith
Contributor(s)	DaPaaS Consortium

Version history

Version	Date	Description
0.1	04.12.2013	Initial outline and Table of Contents (TOC).
0.2	09.12.2013	Restructuring of deliverable with comments.
0.3	16.12.2013	Draft section on requirements specification.
0.4	10.01.2014	Updated the requirements specification with description of key roles.
0.5	16.01.2014	Updated the technology review section and description of requirements.
0.6	17.01.2014	Consistency check and update of sections 1 (Introduction), 2 (Requirements Specification) and 3.1 (High-Level Architecture of the DaPaaS Platform).
0.7	21.01.2014	Updated the architecture description of the Platform Layer.
0.8	23.01.2014	Updated the review of relevant technologies.
0.9	28.01.2014	Finalized technology review. Deliverable ready for internal technical review.
0.95	29.01.2014	Addressed comments by internal technical review. Deliverable ready for quality assurors.
1.0	30.01.2014	Addressed comments by quality assurors. Final formatting and layout.

Executive Summary

The main goal of the DaPaaS project is to provide an integrated Data-as-a-Service (DaaS) and Platform-as-a-Service (PaaS) environment, together with associated services, for open data, where 3rd parties can publish and host both datasets and data-driven applications that are accessed by end user data consumers in a cross-platform manner.

This document provides:

- An overview of the DaPaaS Platform and the relevant roles played in the DaPaaS context;
- The requirements for the DaPaaS Platform;
- An initial architecture design for the Platform Layer of the DaPaaS Platform;
- A state-of-the-art overview of relevant solutions and technologies for the Platform Layer and some recommendations on reuse of existing solutions to be considered in the next phase – implementation of the first prototype.

Table of Contents

List of Acronyms

API	Application Programming Interface
CSV	Comma Separated Values (format)
DaaS	Data-as-a-Service
GUI	Graphical User Interface
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation (format)
PaaS	Platform-as-a-Service
REST	Representational state transfer
RDF	Resource Description Framework
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SPARQL	SPARQL Protocol and RDF Query Language
SSH	Secure Shell
UML	Unified Modeling Language
XML	eXtensible Markup Language

List of Figures

List of Tables

1 Introduction

This report represents Deliverable D2.1 "Open PaaS requirements, design & architecture specification" of the DaPaaS project. This deliverable is a result of Task T2.1 "Requirements, analysis & design of the Open Platform-as-a-Service infrastructure".

The aim of this deliverable is two-fold:

1. To introduce the DaPaaS platform, the relevant roles played in the DaPaaS context, and their requirements towards a Data- and Platform-as-a-Service infrastructure for open data;
2. To provide details on the Platform Layer of the DaPaaS platform, with a focus on the architecture and evaluation of existing relevant technologies that could be reused for the implementation of the Platform Layer.

1.1 DaPaaS Overview and Key Roles

The main goal of the DaPaaS project is to provide an integrated Data-as-a-Service (DaaS) and Platform-as-a-Service (PaaS) environment for open data, where 3rd parties can publish and host both datasets and data-driven applications that are accessed by end user data consumers in a cross-platform manner. The DaPaaS project will deliver the software that enables platform operators to deploy such an environment in the cloud. Figure 1 below illustrates the idea that the DaPaaS software (DaaS and PaaS functionalities) can have several deployed instances.

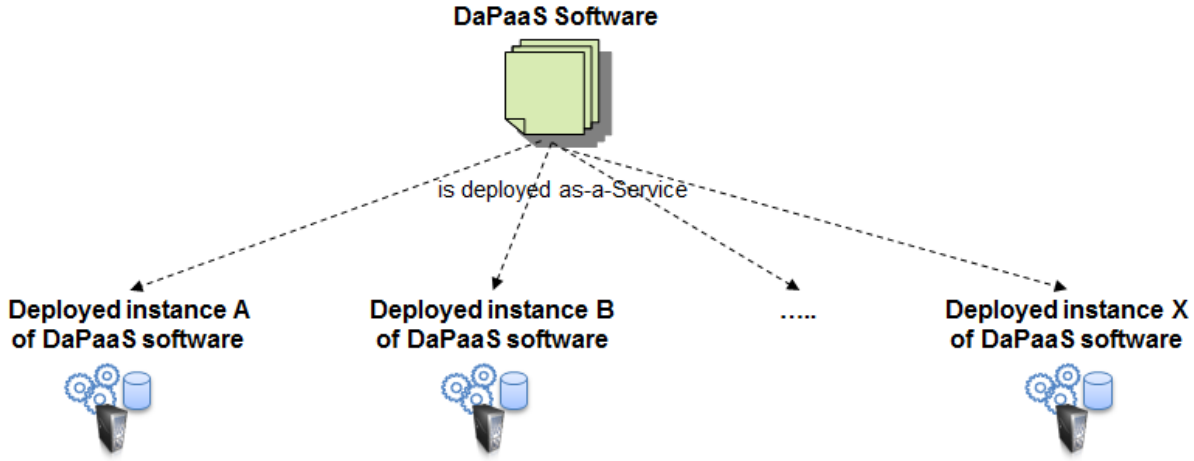


Figure 1: DaPaaS artefacts

As the main results of the DaPaaS project two major artefacts are expected:

1. Software consisting of DaaS, PaaS, and associated services;
2. One deployed instance of the Software in an XaaS manner. In the rest of this deliverable we will refer to this deployed instance as “DaPaaS Platform”.

The key roles involved in a typical DaPaaS context and their relationships to the main DaPaaS artefacts, the software and the platform, are illustrated in Figure 2 below. The roles are:

- The **DaPaaS Developer** implements DaPaaS software components and services for the integrated DaaS and PaaS environment. During the course of the project, this role is expected to be primarily played by the DaPaaS consortium.
- A deployed instance of DaPaaS software, i.e. the DaPaaS Platform, is operated and maintained by an **Instance Operator**. During the course of the project, this role is played by the DaPaaS consortium.
- The **Data Publisher** publishes data on the DaPaaS Platform which stores the data and makes it available for 3rd party application developers and end user data consumers.

- The **Application Developer** develops data-driven applications that use the data made available via the DaPaaS Platform. The applications are deployed and hosted in the DaPaaS Platform.
- **End Users Data Consumers** consume data resulting from the deployed applications.

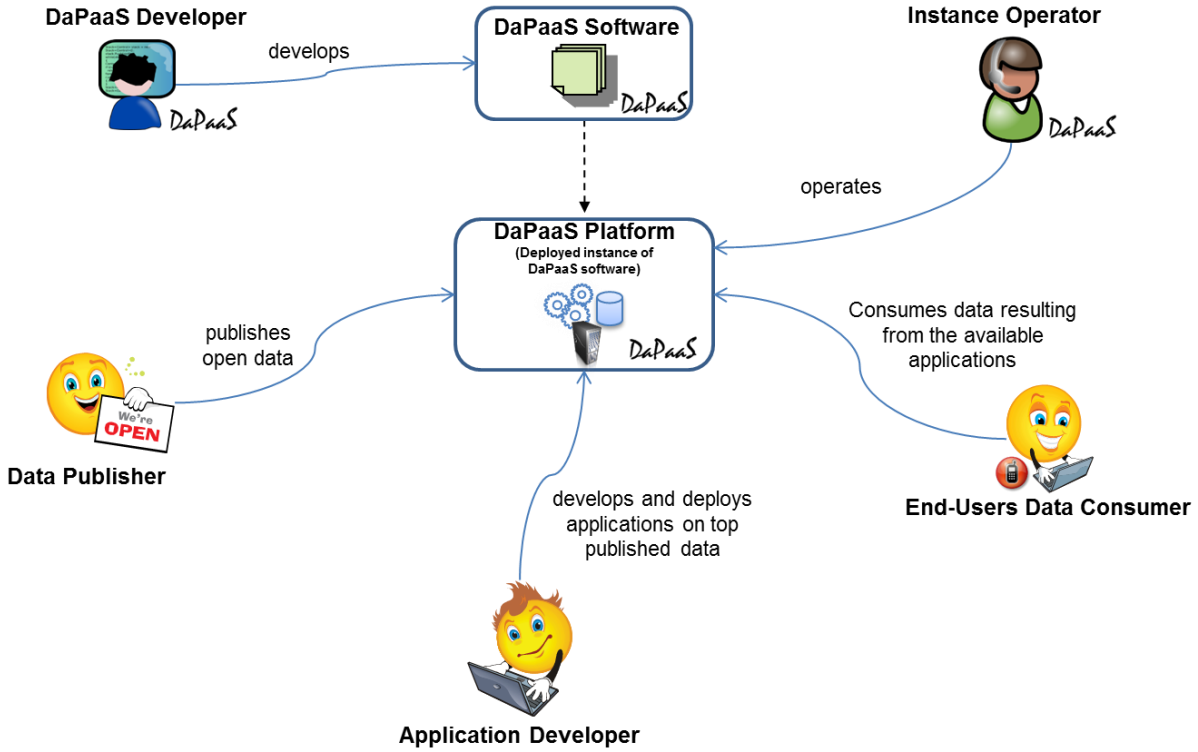


Figure 2: Key roles in a typical DaPaaS context

This document outlines the requirements for the DaPaaS platform from a role-based point of view with a focus on the services and functionality required by the Instance Operator, Data Publisher, Application Developer and End Users Data Consumer.

1.2 Structure of this Report

The rest of this document is structured as follows:

- Section 2 describes the core requirements for the DaPaaS Platform from the perspective of the key roles introduced above;
- Based on the requirements identified in Section 2, Section 3 outlines the high-level architecture of the DaPaaS Platform, and details the Platform Layer in terms of core components and their relationships;
- Section 4 provides a review of relevant open source technologies for the implementation of Platform Layer;
- Section 5 summarizes this document and provides technical recommendations for the implementation phase of the DaPaaS project; and
- Appendix A provides a brief summary of selected commercial/closed source solutions that provide capabilities relevant to the DaPaaS Platform.

2 DaPaaS Platform Requirements Specification

In the following subsections we use a UML use case-inspired notation and technique to describe the requirements of the DaPaaS Platform, e.g. capabilities or services that should be offered for the roles introduced above.

2.1 Instance Operator

The Instance Operator role is played by organizations that want to operate and maintain an instance of the DaPaaS Platform, e.g. acting as data brokers or creating data markets in various domains (e.g. environmental domain). Figure 3 below shows the requirements the Instance Operator poses on the DaPaaS Platform.

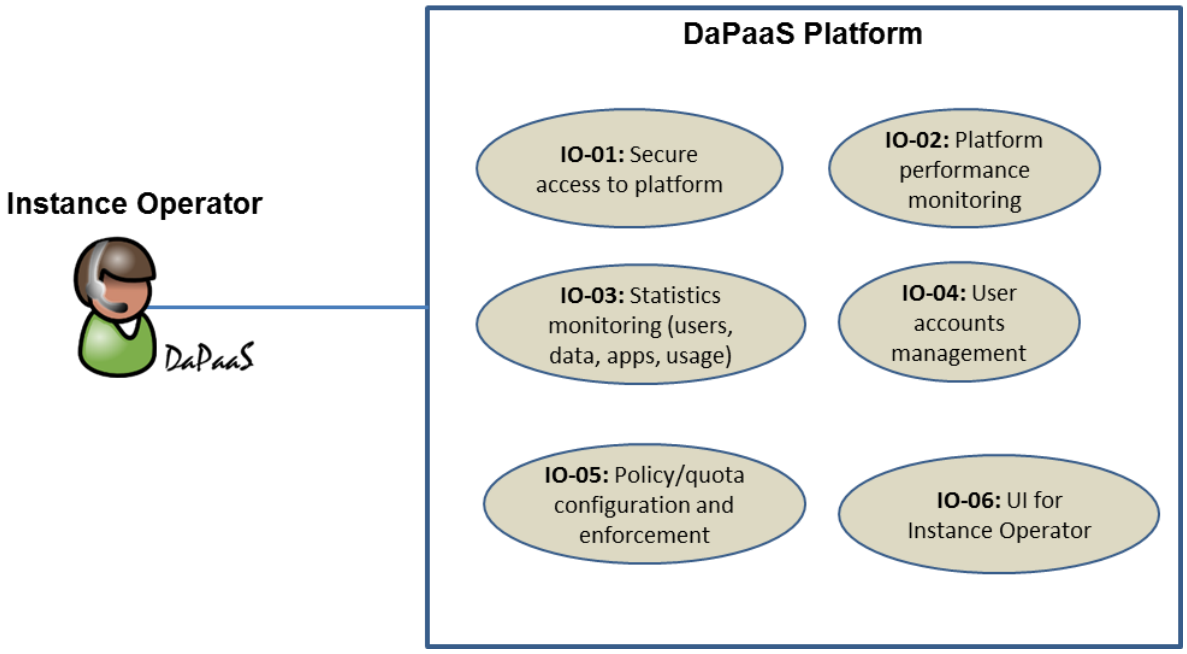


Figure 3: Instance Operator (IO) requirements

Descriptions of these requirements are given in Table 1 below.

Table 1: Description of requirements from Instance Operator (IO)

ID	Name	Brief description
IO-01	Secure access to platform	The Instance Operator shall have secure access (e.g. HTTPS/SSH) to the platform.
IO-02	Platform performance monitoring	The Instance Operator shall be able to monitor the performance (e.g. storage and memory usage, bandwidth, CPU usage, etc.).
IO-03	Statistics monitoring (users, data, apps, usage)	The Instance Operator shall be able to retrieve statistics about users (e.g. number, profiles), data (e.g. number, size), apps and usage (e.g. dataset access, data consumption, number of service calls) as a basis for e.g. billing/invoicing for the usage of the platform.

IO-04	Usage accounts management	The Instance Operator shall be able to manage user accounts (e.g. add, delete, assign roles).
IO-05	Policy/quota configuration and enforcement	The Instance Operator shall be able to configure usage policies, e.g. data/apps quotas per user. The platform shall ensure enforcement of these policies, e.g. support deployment of applications subject to quotas and additional restrictions.
IO-06	UI for Instance Operator	The Instance Operator shall be able to access the platform services through appropriate user interface (graphical and/or console).

2.2 Data Publisher

The Data Publisher role is played by organizations that want to publish data via the DaPaaS Platform. Figure 4 below depicts the Data Publisher poses on the DaPaaS Platform.

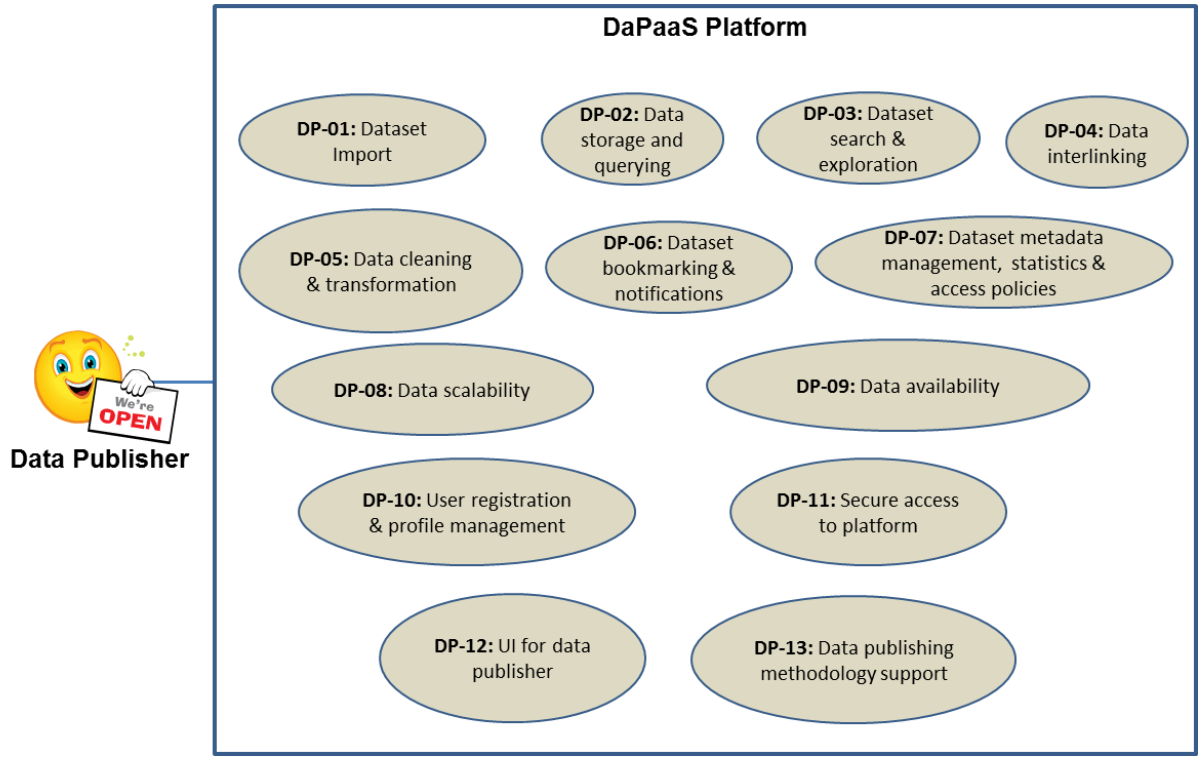


Figure 4: Data Publisher (DP) requirements

Descriptions of these requirements are given in Table 2 below.

Table 2: Description of requirements from the Data Publisher (DP)

ID	Name	Brief description
DP-01	Dataset import	The Data Publisher should have the ability to import open data into the DaPaaS platform. The data is *not* restricted to RDF / Linked Data and it may include other formats such as CSV, JSON, etc.
DP-02	Data storage & querying	The Data Publisher should have access to APIs and query endpoints for accessing, querying and updating data stored on the platform.
DP-03	Dataset search & exploration	The Data Publisher should have the possibility to explore the dataset catalog & select relevant datasets.
DP-04	Data interlinking	The Data Publisher should have the possibility to semi-automatically interlink data from different datasets. This applies only to data which is already in RDF form.
DP-05	Data cleaning & transformation	The Data Publisher should have the possibility to apply simple data cleanup & transformation (incl. RDFization) over legacy data.
DP-06	Dataset bookmarking & notifications	The Data Publisher should have possibility to subscribe to datasets and receive notifications on datasets changes.
DP-07	Dataset metadata management, statistics & access policies	The Data Publisher should have possibility to specify metadata, descriptions and access control policies for the datasets.
DP-08	Data scalability	The platform should include mechanisms to scale to large data volumes.
DP-09	Data availability	The platform should include mechanisms to provide high availability of data and limited downtime.
DP-10	User registration & profile management	The Data Publisher shall be able to register as a data publisher and gain access to the relevant DaaS services.
DP-11	Secure access to platform	The Data Publisher shall have secure access (e.g. HTTPS/SSH) to the platform.
DP-12	UI for Data Publisher	The Data Publisher shall be able to access the DaaS services through appropriate user interfaces (graphical and/or console).
DP-13	Data publishing methodology support	The data publication process should be accompanied by a tool-supported methodology outlining steps containing various data operations

2.3 Application Developer

The Application Developer role is played by Open Data application developers that for various reasons (e.g. transparency, new business models, new services) want to develop new applications and services around data and want to do so as fast as possible and as easy as possible. Figure 5 below depicts the requirements the Application Developer poses on the DaPaaS Platform.

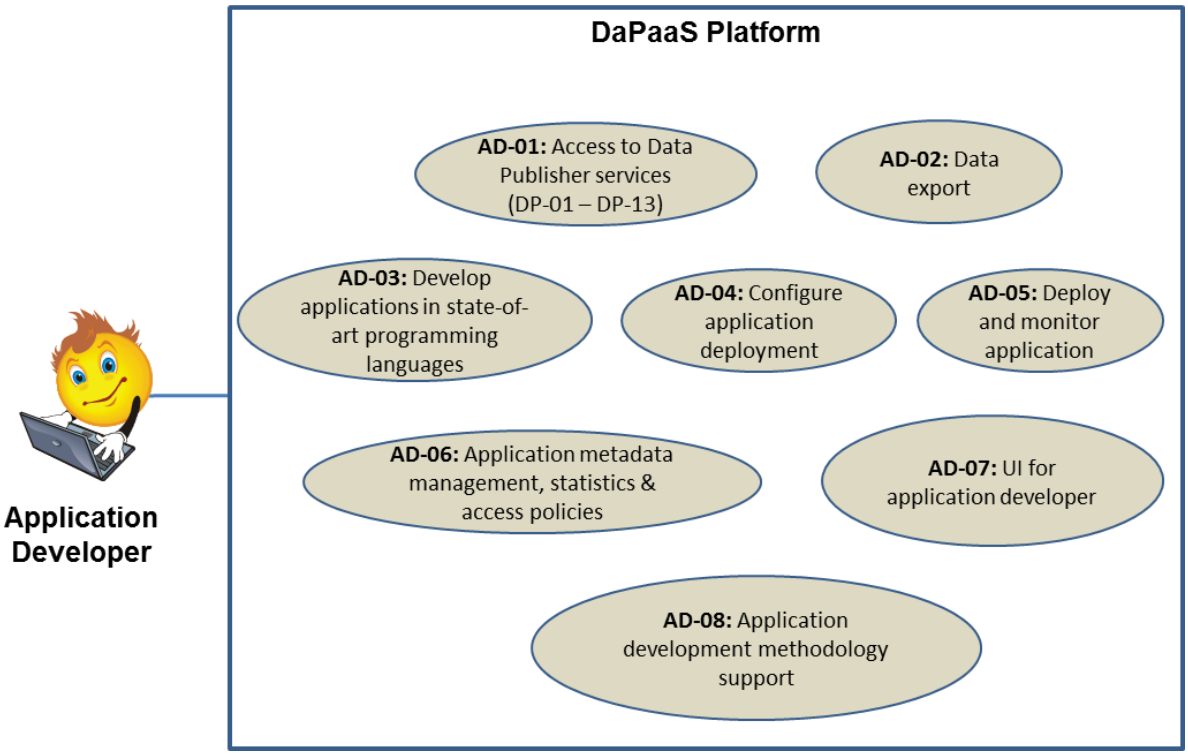


Figure 5: Application Developer (AD) requirements

Descriptions of these requirements are given in Table 3 below.

Table 3: Description of requirements from the Application Developer (AD)

ID	Name	Brief description
AD-01	Access to Data Publisher services (DP-01 – DP-13)	The Application Developer shall have access to APIs and libraries to access, import, transform, store, query, etc., datasets to be used in the development of applications. Basically the Application Developer has similar requirements as outlined in DP-01 – DP-13. This includes also requirements for secure access to the platform, profile management.
AD-02	Data export	The Application Developer shall have the possibility to export data in various formats.
AD-03	Develop applications in state-of-art programming languages	The Application Developer shall have the possibility to develop applications in the common state-of-art programming languages, e.g. Java, Scala, Go, Ruby.
AD-04	Configure application deployment	The Application Developer shall have the possibility to configure use of common cloud resources, e.g. database/storage, possible also graphical widgets.
AD-05	Deploy and monitor application	The Application Developer shall have access to a controlled application hosting environment where data-intensive applications can be easily deployed, as well as monitoring facilities for the deployed applications.

AD-06	Application metadata management, statistics & access policies	The Application Developer shall have the possibility to update metadata about applications (e.g. description) and retrieve statistics about the usage of the application.
AD-07	UI for Application Developer	The Application Developer shall have the possibility to access the relevant DaaS and PaaS services through appropriate user interfaces (graphical and/or console).
AD-08	Application development methodology support	Application Developers should have access to a tool-supported methodology outlining steps for developing and deploying data-intensive applications.

2.4 End User Data Consumer

The End User Data Consumer role is played by organizations or individuals that want to consume data and applications deployed on the platform. Figure 6 below shows the requirements the End User Data Consumer poses on the DaPaaS Platform.

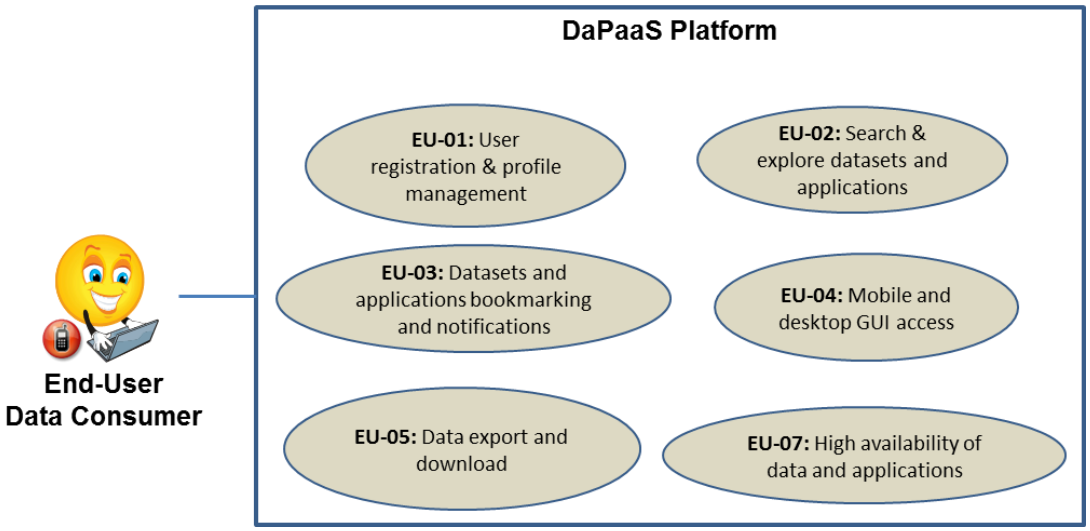


Figure 6: End User Data Consumer (EU) requirements

Descriptions of these requirements are given in Table 4 below.

Table 4: Description of requirements from the End User Data Consumer (EU)

ID	Name	Brief description
EU-01	User registration & profile management	End Users shall be able to register as application consumers and manage their profiles.
EU-02	Search & explore datasets and applications	End Users shall be able to search and explore datasets and applications available in the platform.
EU-03	Datasets and applications bookmarking and notifications	End Users shall be able to bookmark and receive notifications (e.g. updates) of datasets and applications to which they subscribe.

EU-04	Mobile and desktop GUI access	End Users shall be able to access applications on both mobile and desktop devices, which requires UX components to support both mobile and desktop users in an appropriate manner. The End User Data Consumers shall be able to access the relevant platform services, e.g., search for datasets, applications, run applications, visualize datasets, etc., through appropriate graphical user interfaces (GUIs), e.g. pie charts, time series and maps.
EU-05	Data export and download	End Users shall have the possibility to export data in various formats and download data from the platform.
EU-06	High availability of data and applications	High availability of data and apps

3 Architecture Overview

This section outlines the high-level architecture of the DaPaaS Platform (Section 3.1), and details the Platform Layer in terms of core components and their interactions (Section 3.2).

3.1 High-Level Architecture of DaPaaS Platform

The requirements outlined in the previous section imply a layered architecture consisting of a Data-as-a-Service layer (**Data Layer**) for scalable data hosting, a Platform-as-a-Service layer (**Platform Layer**) for application development and hosting, and a User Experience Layer (**UX Layer**) for user-friendly access to data and applications. These three core layers cross-cut vertical layers that are related to methodology support for data publishing and application development, and platform usage monitoring, security, and access control to data and applications. Figure 7 below illustrates a simplified, high-level architecture of the DaPaaS Platform.

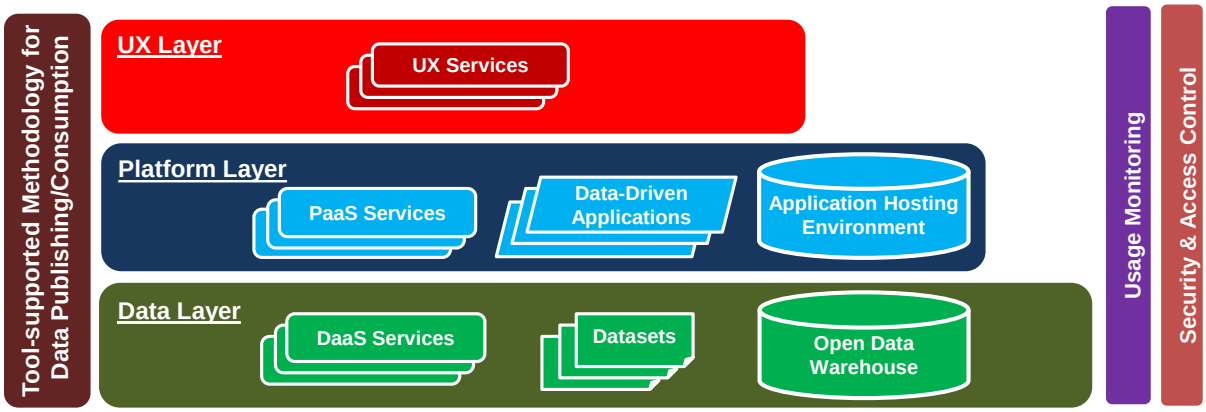


Figure 7: High-level architecture of the DaPaaS Platform

The rest of this deliverable focuses on the Platform Layer of the architecture. An architecture of the Platform Layer is described in the next (Section 3.2) and evaluation of exiting relevant technologies that could be reused for the implementation of the Platform Layer are discussed in Section 4.

The Data Layer and the UX Layer fall within the scope of Deliverable D1.1 and Deliverable D3.1, respectively, and further details about those layers can be found in those deliverables.

3.2 Architecture of the Platform Layer

Figure 8 depicts the main software components, their relationships and associated APIs of the Platform Layer. The software components of the Platform Layer extend the capabilities offered by the Data Layer (described in Deliverable D1.1) in five main service categories plus an administration service:

- **User Management & Access Control** which manages user profiles and secure access control to apps and datasets.
- **Data Cleaning & App Development** which provides functionalities for applications development, and support for data cleaning & transformation and data workflows.
- **Notification** which provides functionality for subscribing to apps and datasets events and notifications.
- **App Management & Deployment** which gives developers control over the deployed applications and configuration settings for the application-hosting environment.
- **Catalog** for searching and exploring apps and datasets.
- **Administration** which allows the management and monitoring of the DaPaaS Platform, focusing on aspects related to the users, apps, datasets and services of the platform.

The functionality of these services can be used by the UX Layer through well-defined APIs. This allows for the creation of e.g. management consoles in the UX Layer. The public APIs of the DaPaaS Platform can also be used by 3rd party applications and services.

The design of the Platform Layer involves a set of software components: User Manager, Access Control Manager, Data Cleaning and Design-Time App Development Services, Run-Time App Hosting Environment, Notification Service and Apps Catalog. Each component is responsible for a functional area of the Platform Layer and communicates with each other through internal service interfaces. The components can access and use the functionality offered through the Data Layer API.

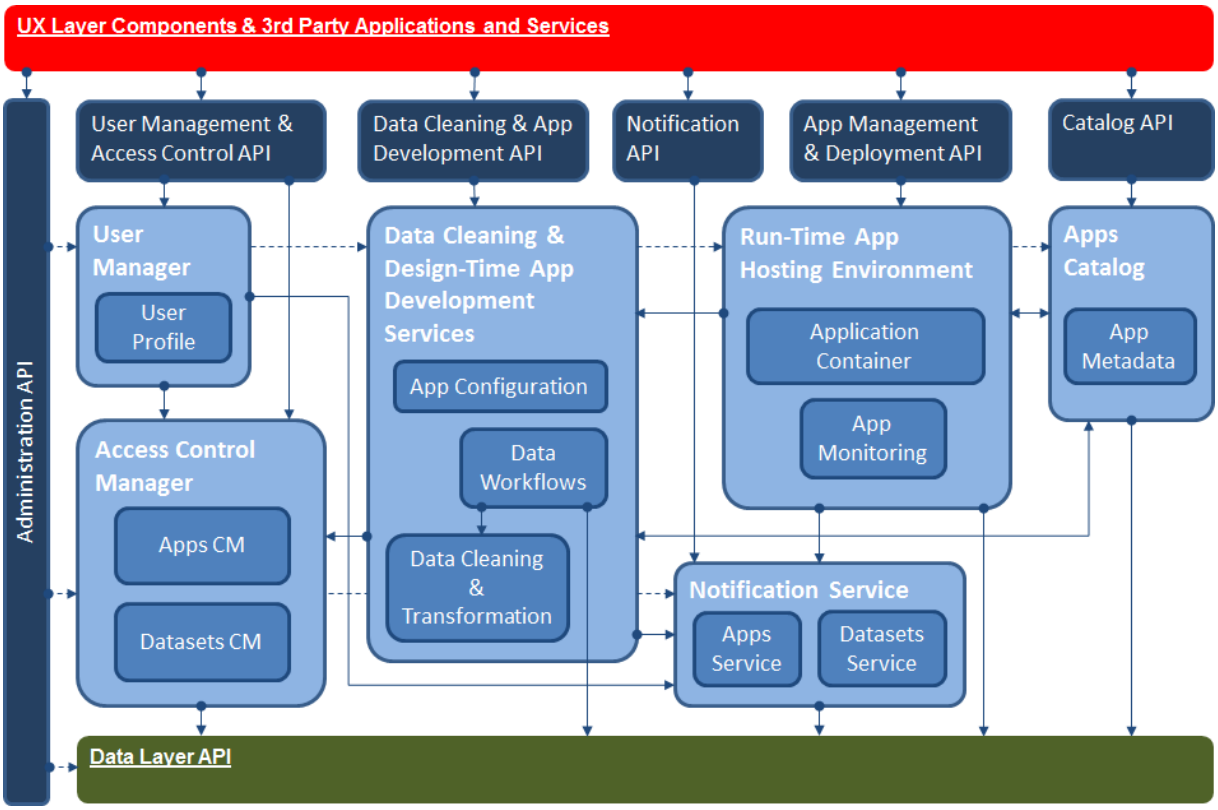


Figure 8: Architecture of the Platform Layer

3.2.1 User Management & Access Control

The **User Management & Access Control** service category will be implemented by the **User Manager** and **Access Control Manager** components. The User Manager component is responsible for managing the registered users of the DaPaaS Platform and their user profiles. Users can register as Data Publishers, Application Developers and End User Data Consumers. The user profile may contain information about data quotas, apps and datasets access which are enforced by the Access Control Manager.

The User Management & Access Control services will be offered through a REST API. The API will provide create, read, update and delete (CRUD) actions for managing accounts, user profiles and access control for apps and datasets.

Table 5: User Management & Access Control

Component	Addressed Requirement	Description
User Manager	DP-06	The user profile stores information about bookmarked datasets. The User Manager component uses the Notification Service to provide notification on dataset changes.
	DP-10	User registration is implemented by the User Manager. It allows users of the DaPaaS Platform to sign up as Data Publisher, Application Developer and/or End User Data Consumer. User account details, preferences and other relevant information are stored in a user profile. The User Manager component provides basic user functionality for all types of users.
	AD-01	Covers the functionality such as DP-06 and DP-10, but for the Application Developer.
	EU-01	Same functionality as DP-10, but for the End User Data Consumer.
	EU-03	The user profiles for an End User Data Consumer provides additional support for bookmarked apps and notifications.
Access Control Manager	DP-07	The Access Control Manager allows access policies for the datasets to be specified.
	AD-06	The Access Control Manager allows access policies for the apps to be specified.

3.2.2 Data Cleaning & App Development

The **Data Cleaning & App Development** service category is a collection of **Data Cleaning & Design-Time App Development Services**, of which **App Configuration**, **Data Workflows** and **Data Cleaning & Transformation** are the main sub-components. The App Configuration is responsible for providing standardized mechanisms to configure cloud resources (e.g. data storage), DaaS services and UX components to be used by the deployed instance of the app at run-time. The Data Cleaning & Transformation component provides additional data management functionalities that complement Data Layer functionality with capabilities for data cleaning (duplicate removal), data transformation, as well as data mapping and alignment. The Data Workflows component provides the capability to define simple data-driven pipelines that use functionality of the Data Layer (e.g., import/export and publish data) and the added functionality offered by the Data Cleaning & Transformation component in order to support simple sequential data transformations.

The App Development services will be offered through a REST API. The API will provide create, read, update and delete (CRUD) actions for managing app configurations. In addition the API will provide actions for accessing and using the DaaS services, including the ability to create simple data workflows.

Table 6: App Development

Component	Addressed Requirement	Description
App Configuration	AD-03	Support for state-of-the-art programming languages will be an evaluation criterion in the review and selection of the PaaS Infrastructure (Section 4).
	AD-04	The App Configuration provides a standard way of configuring the required cloud resources, DaaS services and UX components to be used for app development.

Data Workflows	AD-01	The Application Developer shall have access to a range of DaaS capabilities that can be used for app development. The set of DaaS capabilities will be provided through the Data Workflows component.
	AD-02	The Application Developer shall have the possibility to export data in various formats. Such DaaS functionalities will be provided through the Data workflows component.
	AD-08 and DP-13	Partial methodology support for the development of data-intensive applications can be given by pre-defined data workflows that are useful for application developers and data publishers, e.g., assisting data owners with the process of RDFizing their data. The implementation support for methodology will be developed in the context of WP4 (DaPaaS Methodology) further elaborated in Deliverable D4.1.
Data Cleaning & Transformation	DP-05	The Data Cleaning & Transformation will implement capabilities for data cleaning (e.g. duplicate removal).

3.2.3 Notification

The **Notification** service category is implemented by a **Notification Service**. It provides functionality for subscribing to events and notifications about apps and datasets.

The Notification Service will be offered through a REST API. The API will provide actions for creating, deleting and listing topics, actions for subscribing and unsubscribing to such topics, and actions for publishing messages to subscribers of specific topics.

Table 7: Notification

Component	Addressed Requirement	Description
Notification Service	DP-06	The Notification Service must implement support for publishing and subscribing different types of events and notifications for datasets.
	EU-03	The Notification Service must also implement support for publishing and subscribing events and notifications for users and apps.

3.2.4 App Management & Deployment

The **App Management & Development** service category will be implemented by components such as **Application Container** and **App Monitoring**. The Application Container provides a cloud-provisioned environment where developers can deploy and run applications. The App Monitoring component will provide functionality for monitoring the usage of services and consumption of data on the DaPaaS Platform as well as enforcing resource quotas which guarantee the “fair use” of the Platform by 3rd parties.

The App Management & Development services will be offered through a REST API. The API will provide actions for setting up the application container and deploy and undeploy apps. The API will also provide capabilities for logging relevant data about running apps for monitoring and statistics purposes.

Table 8: App Management & Deployment

Component	Addressed Requirement	Description
Application Container	AD-05	The Application Container provides the ability to deploy and run an application.
	EU-06	The Application Container should provide mechanisms such as load balancing and app scalability to ensure high availability.
App Monitoring	DP-07	The App Monitoring provides functionality to track usage of datasets for which statistics can be collected.
	AD-06	The App Monitoring provides functionality to track usage of the apps for which useful statistics can be collected.

3.2.5 Catalog

The **Catalog** service category will provide capabilities for application metadata management, similar to the dataset catalog in the DaaS layer. The Catalog service will be offered through a REST API. The API will provide search actions for apps and create read, update and delete (CRUD) actions for managing metadata of apps.

Table 9: Catalog

Component	Addressed Requirement	Description
Apps Catalog	AD-06	The Apps Catalog will contain relevant metadata about the apps, e.g. description and number of users.
	EU-02	The Apps Catalog will provide functionality for searching and exploring apps based on metadata.

3.2.6 Administration

The **Administration** service category will provide capabilities primarily targeting the administrator of the platform, i.e., the Instance Operator. Administration actions provide extended functionality that makes it easier for administrators to manage users, access control policies, apps and datasets. This extended functionality will be implemented by the corresponding component of the Platform Layer, but the actions will not be exposed in the public APIs available to the other users of the platform (i.e., data publishers, apps developers, or end user data consumers). Instead, separate APIs are defined for the administrator.

Table 10: Administration

Component	Addressed Requirement	Description
User Management & Access Control	IO-01 and DP-11	The Instance Operator and Data Publisher (and implicitly Application Developer) shall have secure access (e.g. HTTPS/SSH) to the platform.
	IO-02	An Administration API will provide monitoring and logging capabilities of the complete run-time app hosting environment.
	IO-03	An Administration API will provide extended actions for retrieving statistics about users and their usage of apps and datasets.

	IO-04	An Administration API will provide actions for managing user accounts.
	IO-05	An Administration API will provide actions for configuring usage policies, e.g., predefining different policies that users can select during sign-up.

3.3 Summary of Addressed Requirements

Table 11 below depicts which DaPaaS Platform requirements (introduced in Section 2) are addressed by which components at the Platform Layer (a '+' is used to indicate this relation for each requirement).

The requirements that are not addressed by the Platform Layer are marked in grey. The requirements DP-01, DP-02, DP-03, DP-04, DP-08, DP-09 and EU-05 are addressed at the Data Layer and further elaborated in Deliverable D1.1. The requirements DP12 and EU-04 are addressed at the UX Layer and further elaborated in Deliverable D3.1.

The requirements AD-08 and DP-13 focus on methodology support. In the Platform Layer architecture partial methodology support can be provided by the Data Workflows component (see Section 3.2.2). A separate methodology component (e.g. providing online guidelines and wizards) may be introduced in the architecture. This methodological aspect is out of the scope of this document and will be addressed as part of Deliverable D4.1 in WP4 (DaPaaS Methodology).

Table 11: Addressed requirements by components of the Platform Layer

DaPaaS Platform Requirement	User Management & Access Control	Data Cleaning & App Development	Notification	App Management & Deployment	Catalog	Administration
IO-01	+					+
IO-02				+		+
IO-03	+			+		+
IO-04	+					+
IO-05	+					+
IO-06						+
DP-01						
DP-02						
DP-03						
DP-04						
DP-05		+				
DP-06	+		+			
DP-07	+			+		
DP-08						
DP-09						
DP-10	+					
DP-11	+					

DP-12						
DP-13						
AD-01	+	+				
AD-02		+				
AD-03		+				
AD-04		+				
AD-05				+		
AD-06	+			+	+	
AD-07						
AD-08						
EU-01	+					
EU-02					+	
EU-03	+		+			
EU-04						
EU-05						
EU-06				+		

4 Review of Relevant Technologies for the Platform Layer

This section presents the state-of-the-art analysis of relevant technologies for the Platform Layer and includes an explanation of the technology selection (Section 4.1) and review of relevant technologies for the design and implementation of the Platform Layer of the DaPaaS Platform (Section 4.2 and 4.3).

4.1 Technology Selection Approach

There are many PaaS vendors in the market nowadays. The website http://clouds360.com lists the top 20 PaaS vendors. The three dominant players in the PaaS market are Amazon with its Amazon Web Services (AWS)[1] solution, Google with its Google App Engine[2] and Microsoft with its Windows Azure[3]. Another main commercial vendor in the market is Salesforce.com with its Salesforce1 Platform[4]. Open-source alternatives also exist, e.g. AppScale[5] and OpenStack[6].

The implementation strategy for the DaPaaS Platform is to utilize open source technologies and be based on open standards as much as possible. This does not necessarily exclude commercial components to be used in the platform (such as the OWLIM RDF database from Ontotext for the DaaS layer) as long as they are compliant with relevant open standards (such as those produced by the W3C for RDF and SPARQL, in the case of OWLIM) and can easily be replaced by alternative open source products.

Due to many available open source solutions relevant for the Platform Layer, for the development of the Platform Layer the aim is to be based entirely on open source technologies. At the very least the core component of the Platform Layer, i.e., the Data Cleaning & App Development and the App Management & Deployment, will be based on existing open source tools and frameworks. For this reason the review of relevant technologies for the Platform Layer is focused on open source solutions for PaaS and data integration capabilities:

- **PaaS capabilities:** A Platform-as-a-Service (PaaS) provides the capability for 3rd party application developers to deploy on a cloud infrastructure 3rd party applications developed using programming languages, libraries, services and protocols provided by the PaaS provider. The PaaS provides 3rd party developers control over the deployed applications and configuration settings for the application-hosting environment. Typical PaaS service offerings include capabilities for application design, development, deployment, data access, security, scalability, storage, application instrumentation, service monitoring, workflow management, discovery, etc.
- **Data integration capabilities:** The DaaS services of the Data Layer primarily mainly provide automated functions for data management (e.g. import, linking, etc.). The data integration capabilities of the Platform Layer provide additional semi-automated functions that help the Data Publisher in publishing data and the Application Developer in developing data-driven applications. Examples of such capabilities are data cleaning, transformation, workflows and methodology that often requires a graphical user interface to interact with.

Table 12 below lists the selected solutions that have been analysed with respect to PaaS and data integration capabilities for the Platform Layer design and implementation. Further details on the evaluations are provided in the following sub-sections.

[1] http://aws.amazon.com/
[2] https://developers.google.com/appengine/
[3] www.windowsazure.com/
[4] http://www.salesforce.com/eu/platform/overview/
[5] http://www.appscale.com/
[6] http://www.openstack.org/

Table 12: Overview of relevant open source technologies

Capabilities	Solution	Short description	Open source license
PaaS Solutions	Docker	Docker is an open source solution to easily create lightweight, portable, self-sufficient containers from any application.	Apache License Version 2.0
	Cocaine	Cocaine is an open-source PaaS system for creating custom cloud hosting apps that are similar to Google App Engine or Heroku.	GNU Lesser General Public License (GPL) Version 3
	Deis	Deis is an open source PaaS that makes it easy to deploy and scale Docker containers and Chef nodes used to host applications, databases, middleware and other services.	Apache License Version 2.0
	Juju	Juju is a tool for configuring, managing, maintaining and deploying applications based on Charms application configurations.	GNU Affero General Public License (aGPL) Version 3
	Cozy Cloud	Cozy Cloud allows users to build a Personal Cloud Platform-as-a-Service (PaaS) where users can deploy personal web applications (official apps such as Calendar, Contacts and Photos are available from a marketplace) or write apps from scratch.	GNU Lesser General Public License (LGPL) Version 3
	OpenCivic	OpenCivic is an open source resource cataloguing, hackathon and app store management platform designed to help organizations better collaborate in developing, sharing and maintaining information and apps that solve civic problems.	Available as open source on GitHub, but no licensing information is given.
	OpenStack	OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a Web interface.	Apache License Version 2.0
	Ansible	Ansible is an automation engine that aims to make systems and apps simple to deploy.	GNU General Public License (GPL) Version 3
	Puppet Open Source	Puppet Open Source is a flexible, customizable framework designed to help system administrators automate the many repetitive tasks they regularly perform.	Apache License Version 2.0
	Chef	Chef is a systems and cloud infrastructure automation framework that aims to make it easy to deploy servers and applications to any physical, virtual, or cloud location.	Apache License Version 2.0
	Nagios Core	Nagios Core is an open source IT monitoring system that monitors critical IT infrastructure components, including system metrics, network protocols, applications,	GNU General Public License (GPL) Version 2

		services, servers, and network infrastructure.	
Data Integration Solutions	Talend Open Studio for Data Integration	Talend Open Studio for Data Integration provides a set of data integration tools to access, transform and integrate data from any business system in real time or batch to meet both operational and analytical data integration needs.	GNU Lesser General Public License (LGPL) Version 3
	OpenRefine	OpenRefine is a tool for working with messy data, cleaning it, transforming it from one format into another, extending it with Web Services and linking to databases.	Open source Google license with dependencies to a number external open source licenses.
	Karma	Karma is an information integration tool that enables users to quickly and easily integrate data from a variety of data sources including databases, spreadsheets, delimited text files, XML, JSON, KML and Web APIs.	Apache License Version 2.0
	Cascading	Cascading is an application framework for Java developers to develop robust data analytics and data management applications on Apache Hadoop.	Apache License Version 2.0
	Data Pipes	Data Pipes is a service to provide streaming, "pipe-like" data transformations on the web – things like deleting rows or columns, find and replace, head, grep etc.	MIT License

4.2 PaaS Capabilities Solutions

4.2.1 Docker

Docker[7] is an open source project to easily create lightweight, portable, self-sufficient containers from any application. It has a very active user community[8] and has more than 200 contributors.

The main feature of Docker is to provide portable deployment across machines by packaging software in a common kind of container. Docker defines a format for bundling an application and all its dependencies into a single object which can be transferred to any Docker-enabled machine, and executed there with the guarantee that the execution environment exposed to the application will be the same. This allows the packaged applications to run in different environments without being reconfigured again.

Figure 9 illustrates the basic functionality of Docker[9]. A container comprises both an application and all of its dependencies. Containers can either be created manually or automatically in a source code repository (requires a DockerFile). Subsequent modifications to a baseline Docker image can be committed to a new container using the Commit function and then Pushed to a Central Registry. Containers can be found in a Docker Registry using Search. Containers can be pulled from the registry using Pull and can be run, started, stopped, etc. using Run commands. The target of a run command can be self-owned servers, public instances, or a combination.

[7] https://www.docker.io/
[8] http://blog.docker.io/2013/11/docker-project-community-stats/
[9] https://www.docker.io/the_whole_story/#What-are-the-Main-Features-of-Docker

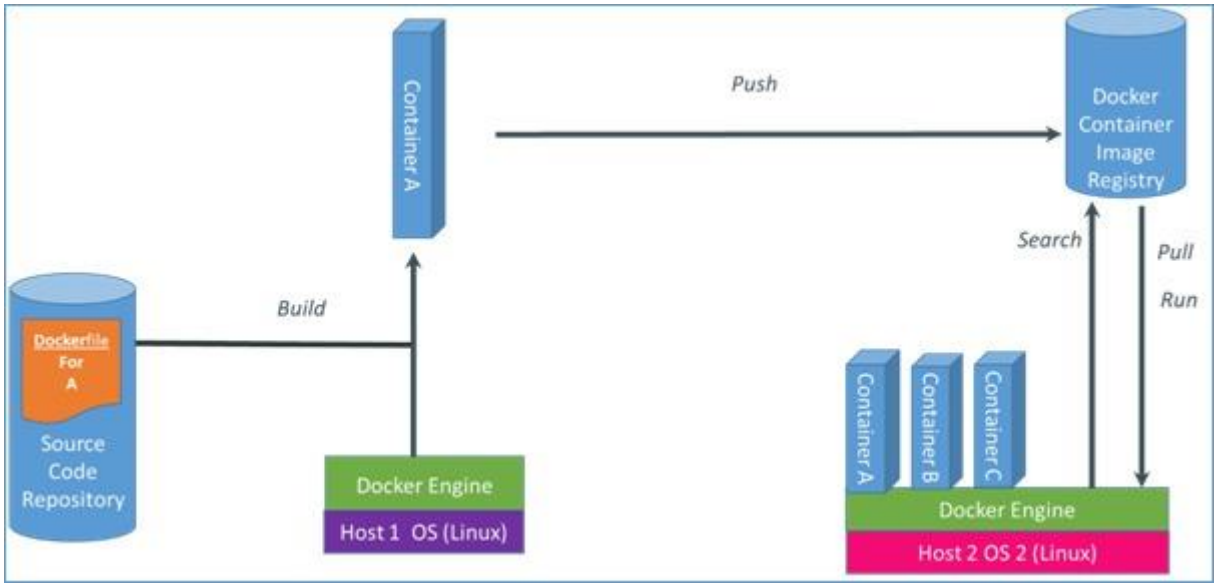


Figure 9: Basic Docker functions[10]

In addition the Docker tool offers features for application deployment, automatic build, versioning and component re-use. Docker also provides an API[11] for automating and customizing the creation and deployment of containers.

Docker is released under the open source Apache License Version 2.0 license[12].

4.2.2 Cocaine

Cocaine[13] (Configurable Omnipotent Custom Applications Integrated Network Engine) is an open-source PaaS system for creating custom cloud hosting apps that are similar to Google App Engine[14] or Heroku[15], which are not open source and locked in to the Google and Amazon cloud platforms respectively. Cocaine is developed by Yandex, the leading search engine in Russia, which organizes the community and conferences.

The Cocaine architecture (see Figure 10) simplifies the creation of cloud hosting apps by hiding the infrastructure details and the applications environment settings from the developer. The developer only needs to send the code to the Cocaine server and write a special manifest for executing the code. It is not necessary to set up anything else, such as databases, as these are handled by services in the infrastructure. Any library or service can be implemented as a service in Cocaine using a special API[16]. From the programmer's point of view, these services look like native modules for the programming language the code is written in.

One of the notable features of Cocaine is that apps are driven by events. There are two sources of events for every app, and there exists lots of predefined plugins providing those sources. Firstly there exists services such as publish-subscribe notification and secondly there are event drivers allowing developers to generate app events.

Cocaine uses Docker as the underlying application container. Cocaine provides a plugin which connects to Docker and controls it using a rich REST API.

[10] Figure taken from https://www.docker.io/the_whole_story/#What-are-the-Main-Features-of-Docker
[11] http://docs.docker.io/en/latest/api/
[12] https://github.com/dotcloud/docker/blob/master/LICENSE
[13] http://api.yandex.com/cocaine/
[14] https://developers.google.com/appengine/
[15] https://www.heroku.com/
[16] https://github.com/cocaine/cocaine-docs/blob/v0.11/doc/contents.md

Cocaine is released under the open source GNU Lesser General Public License (LGPL) Version 3 license[17].

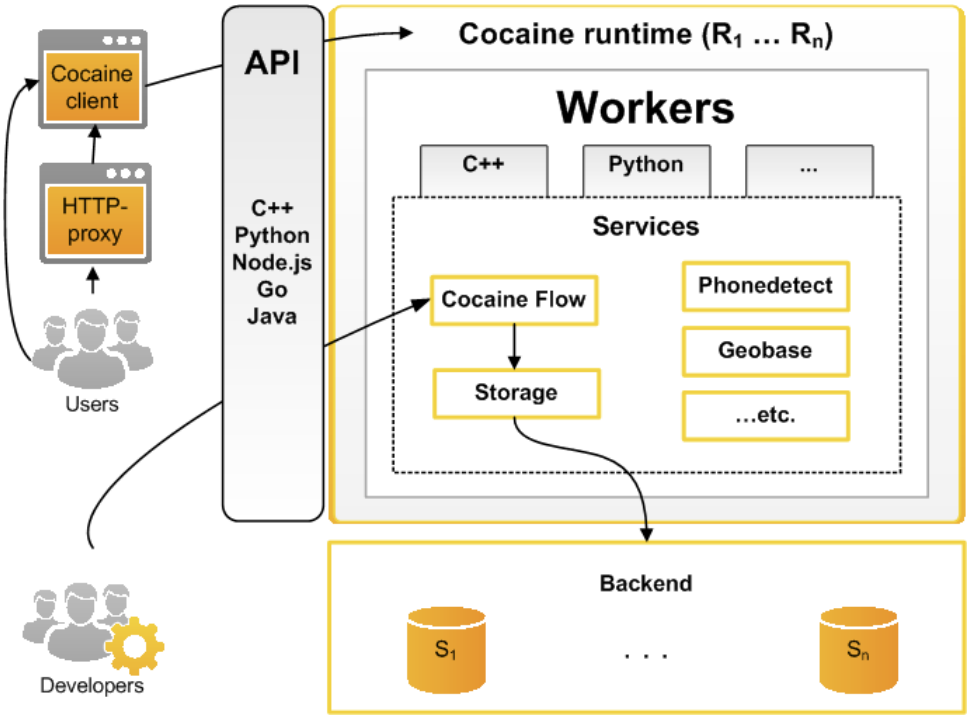


Figure 10: Cocaine architecture[18]

4.2.3 Deis

Deis[19] is an open source PaaS that makes it easy to deploy and scale Docker containers and Chef nodes used to host applications, databases, middleware and other services. Deis leverages Chef, Docker and Heroku Buildpacks[20] to combine a Heroku-inspired application platform for public and private clouds. Deis is a fairly new open source project, but has high ambitions with a regular release schedule with a production ready 1.0 release expected soon[21].

The main component of Deis is the controller component[22]. Controllers are tied to a configuration management backend where data about users, applications and formations is stored. A formation is a set of infrastructure used to host applications.

The controller is in charge of:

- Processing client API calls
- Managing nodes that provide services to a formation
- Managing containers that perform work for applications
- Managing proxies that route traffic to containers
- Managing users, providers, flavors, keys and other base configuration

[17] https://github.com/cocaine/cocaine-core/blob/master/LICENSE
[18] Figure taken from http://api.yandex.com/cocaine/
[19] http://deis.io/overview/
[20] https://devcenter.heroku.com/articles/buildpacks
[21] http://deis.io/deis-devops-and-the-future-of-open-paas/
[22] http://docs.deis.io/en/latest/components/controller/

Deis provides a REST API[23] for integration with other tools and systems. Deis is released as open source under the Apache License 2.0 license[24].

4.2.4 Juju

Juju[25] is a tool for configuring, managing, maintaining and deploying Charms application architectures. Charms[26] encapsulate application configurations, define how services are deployed, how they connect to other services and are scaled. A Charm is essentially a structure bundled of files that contain metadata, configuration data and hooks (e.g., executable files).

Juju is a project launched by Canonical, which are the developers of the Ubuntu Linux-based operating system, and has a strong community behind it. Juju allows users to deploy, manage and scale software and interconnected services across one or more Ubuntu servers and cloud platforms.

Juju provides a GUI (see Figure 11) and command-line interface to define, configure, deploy, manage, monitor and scale services to any public or private cloud.

Juju is released under the open source GNU Affero General Public License (aGPL) Version 3 license[27].

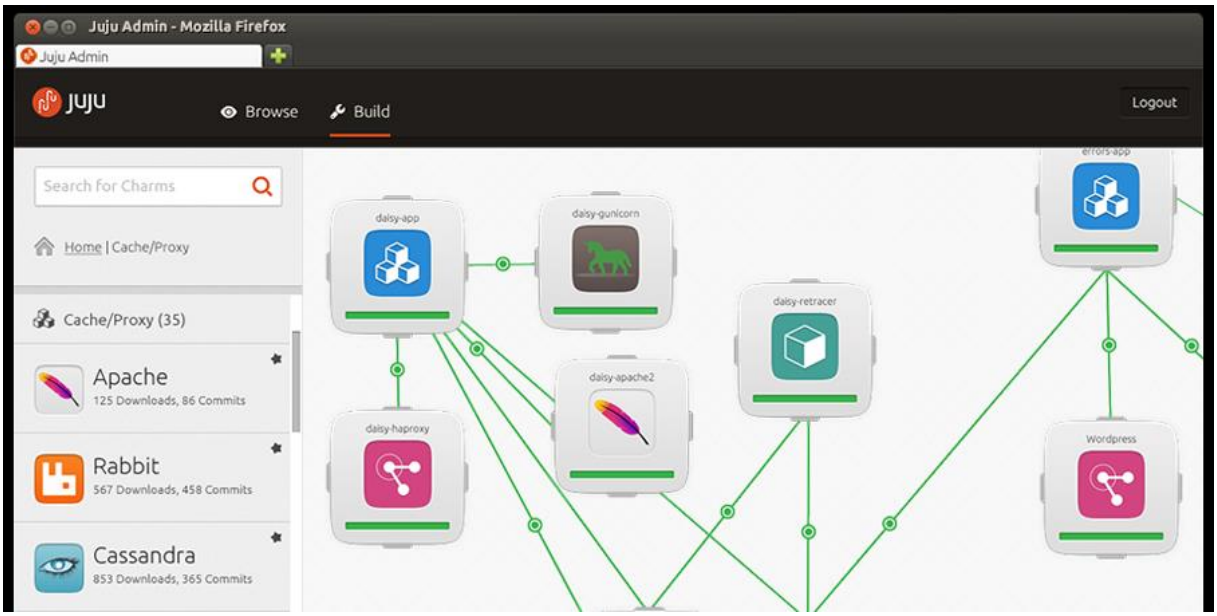


Figure 11: Juju administration GUI[28]

4.2.5 Cozy Cloud

Cozy Cloud[29] allows users to build a Personal Cloud Platform-as-a-Service (PaaS) where users can deploy personal web applications (official apps such as Calendar, Contacts and Photos are available from a marketplace) or write apps from scratch. Cozy Cloud provides a centralized storage with data types and access control that can be shared/used by the apps.

Cozy Cloud is developed by a young startup company located in France with a small team of eight persons. Cozy Cloud was developed to be a private **personal** cloud solution that allows users to host their own personal applications in a single place that they control. This way, users can manage their data from anywhere while protecting their privacy. The design of the platform is very focused on the

[23] http://docs.deis.io/en/latest/server/
[24] https://github.com/opdemand/deis/blob/master/LICENSE
[25] https://juju.ubuntu.com/
[26] https://juju.ubuntu.com/charms/
[27] https://launchpad.net/juju-core
[28] Figure taken from https://juju.ubuntu.com/features/
[29] https://www.cozycloud.cc/

personal single-user perspective. Thus capabilities for handling large data and multi-users have not been addressed in the architecture. Because of the focus on the single-user perspective, the Cozy Cloud solution does not seem suitable for the implementation of the Platform Layer in DaPaaS.

Cozy Cloud is released as open source under the GNU Lesser Generic Public License (LGPL) Version 3 license[30].

4.2.6 OpenCivic

OpenCivic[31] is an open source resource cataloguing, hackathon and app store management platform designed to help organizations better collaborate in developing, sharing and maintaining information and apps that solve civic problems. OpenCivic is based on Drupal. The main goal of the Drupal distro is to help build websites that enable people to share information about software applications.

OpenCivic provides features for app store and hackathons:

- Catalog applications and their metadata, including pictures and their descriptions
- Catalog organizations that use and contribute to apps
- Store and publish open data for developers to use in application development
- Catalog deployments of specific apps by organizations and locations
- Publish and manage hackathons events
- Collect, define, rate and refine problems for the events
- Define and manage development teams for the events

The OpenCivic distribution for Drupal is available as open source on GitHub[32], but no licensing information is given.

4.2.7 OpenStack

OpenStack[33] is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface. OpenStack has a very large community and top industry involvement behind it, organized as the OpenStack Foundation[34].

Figure 12 shows a conceptual architecture of the OpenStack from the operator side of the cloud, i.e. the Instance Operator. The OpenStack provides APIs for:

- Dashboard (Web frontend)
- Store and retrieval of virtual disks/images and associated metadata, as well as the virtual disk files themselves
- Virtual network and storage volumes for computing
- Identity and service authentication

[30] https://github.com/mycozycloud/cozy-setup/blob/master/LICENSE
[31] http://nucivic.com/opencivic/
[32] https://github.com/civic-commons/opencivic
[33] http://openstack.org
[34] http://www.openstack.org/community/

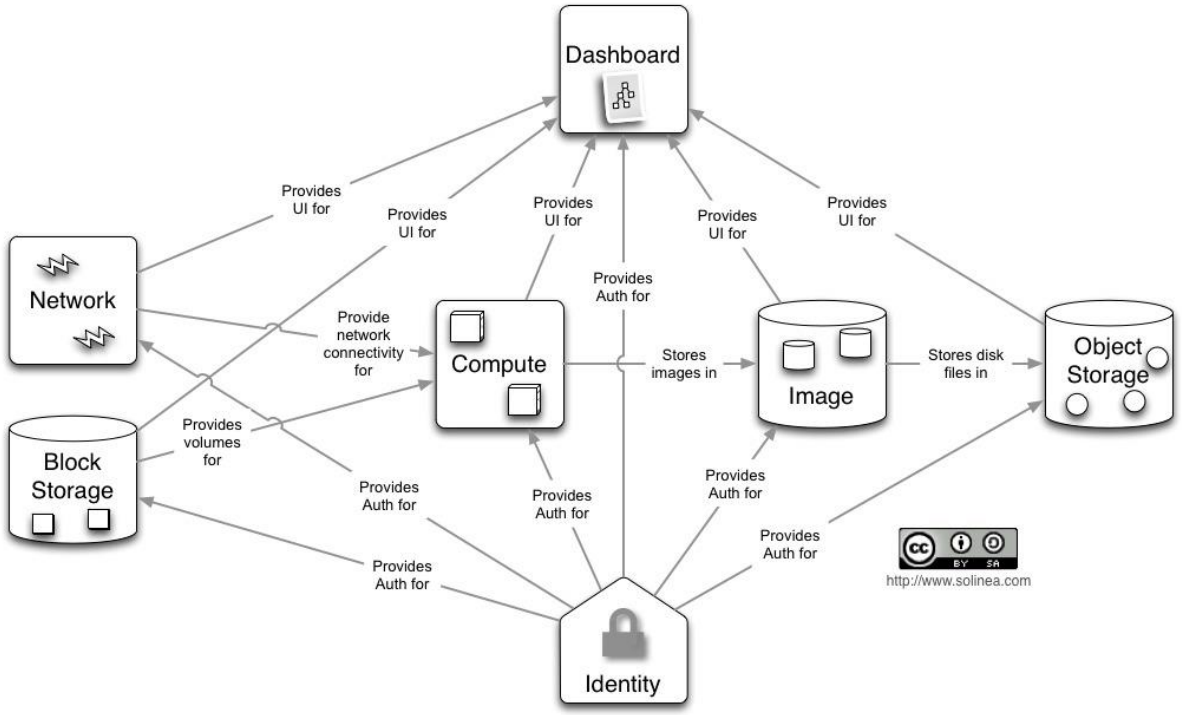


Figure 12: OpenStack conceptual architecture[35]

The OpenStack software supports allocating a large amount of servers to provide resources for computation. These resources can then be consumed in a uniform manner through the OpenStack abstraction layer[36].

The OpenStack project is provided[37] under the open source Apache License 2.0 license[38].

4.2.8 Ansible

Ansible[39] is a powerful automation engine that makes systems and apps simple to deploy. Ansible is an IT automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates. There is a large community around the tool[40].

Ansible provides features such as[41]:

- Command Line Tools
- Application Deployment
- Continuous Delivery
- Multi-Tier Orchestration
- Configuration Management
- Agentless Architecture

[35] Figure taken from http://docs.openstack.org/grizzly/openstack-compute/admin/content//conceptual-architecture.html
[36] http://api.openstack.org/api-ref.html
[37] https://wiki.openstack.org/wiki/Getting_The_Code
[38] https://wiki.openstack.org/wiki/Open
[39] http://www.ansibleworks.com/
[40] http://www.ansibleworks.com/community/
[41] http://www.ansibleworks.com/pricing/

- SSH-Based Security

The tool provides a Python API[42]. Modules and plugins can be developed in any programming language[43].

Ansible is available as an open source engine[44] released under the GNU General Public License (GPL) Version 3 license[45].

4.2.9 Puppet Open Source

Puppet Open Source[46] is a flexible, customizable framework available under the Apache 2.0 license designed to help system administrators automate the many repetitive tasks they regularly perform. As such it is similar to Ansible, namely an IT automation tool. It has a large and active community[47].

Puppet uses a declarative, model-based approach to IT automation[48].

1. Define the desired state of the infrastructure's configuration using Puppet's declarative configuration language.
2. Simulate configuration changes before enforcing them.
3. Enforce the deployed desired state automatically, correcting any configuration drift.
4. Report on the differences between actual and desired states and any changes made enforcing the desired state.

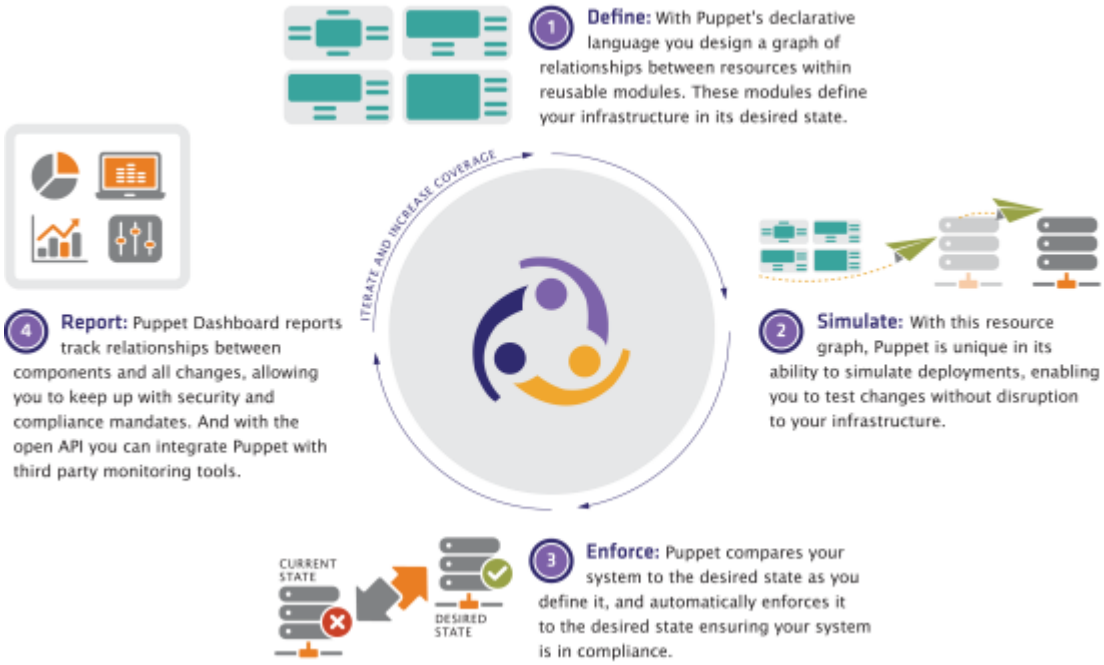


Figure 13: How Puppet works[49]

[42] http://docs.ansible.com/developing_api.html
[43] http://docs.ansible.com/developing.html
[44] https://github.com/ansible/ansible
[45] http://www.ansibleworks.com/opensource/
[46] http://puppetlabs.com/puppet/puppet-open-source
[47] http://puppetlabs.com/community/overview
[48] http://puppetlabs.com/puppet/what-is-puppet
[49] Figure taken from http://puppetlabs.com/puppet/what-is-puppet

4.2.10 Chef

Chef[50] is a systems and cloud infrastructure automation framework that makes it easy to deploy servers and applications to any physical, virtual, or cloud location, no matter the size of the infrastructure. It is an IT automation and configuration management tool similar to Ansible and Puppet. It has a large and active community[51].

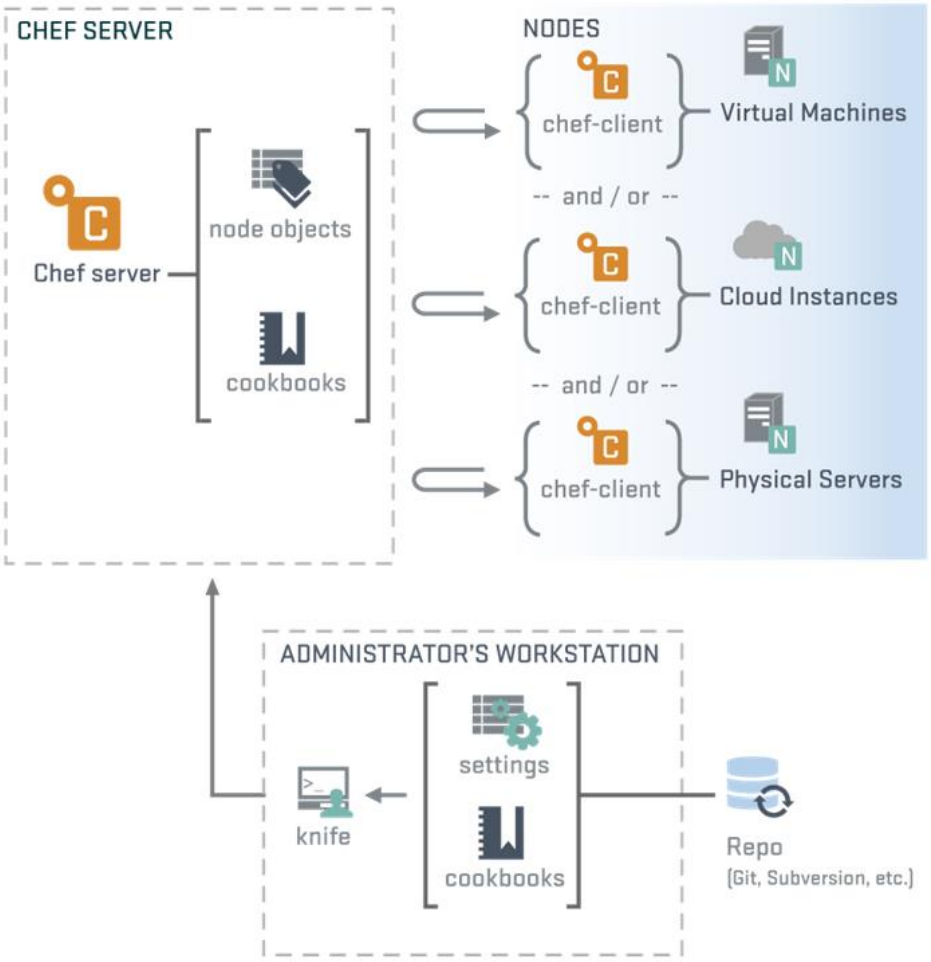


Figure 14: How Chef works[52]

Chef relies on reusable definitions known as cookbooks and recipes (see Figure 14) that are written using the Ruby programming language. Cookbooks and recipes automate common infrastructure tasks. Their definitions describe what your infrastructure consists of and how each part of your infrastructure should be deployed, configured and managed. Chef applies those definitions to servers to produce an automated infrastructure. The Chef server stores your network's configuration data and recipes.

Chef is released under the Apache License Version 2.0 license[53].

[50] http://docs.opscode.com/chef_overview.html
[51] http://community.opscode.com/
[52] Figure taken from http://www.getchef.com/chef/
[53] https://github.com/opscode/chef/blob/master/LICENSE

4.2.11 Nagios Core

Nagios Core[54] is an open source host, service and network monitoring tool. It allows monitoring entire IT infrastructures to ensure systems, applications, services, and business processes are functioning properly. In the event of a failure, it can alert technical staff of the problem, allowing them to begin remediation processes before outages affect business processes, end users, or customers. It is capable to manage different types of services and hosts running on different operating systems such as Linux, Netware, Windows, AIX, etc. It's flexible in configuration and can be extended as much as it is necessary. It's configured within text files and managed with a Web browser.

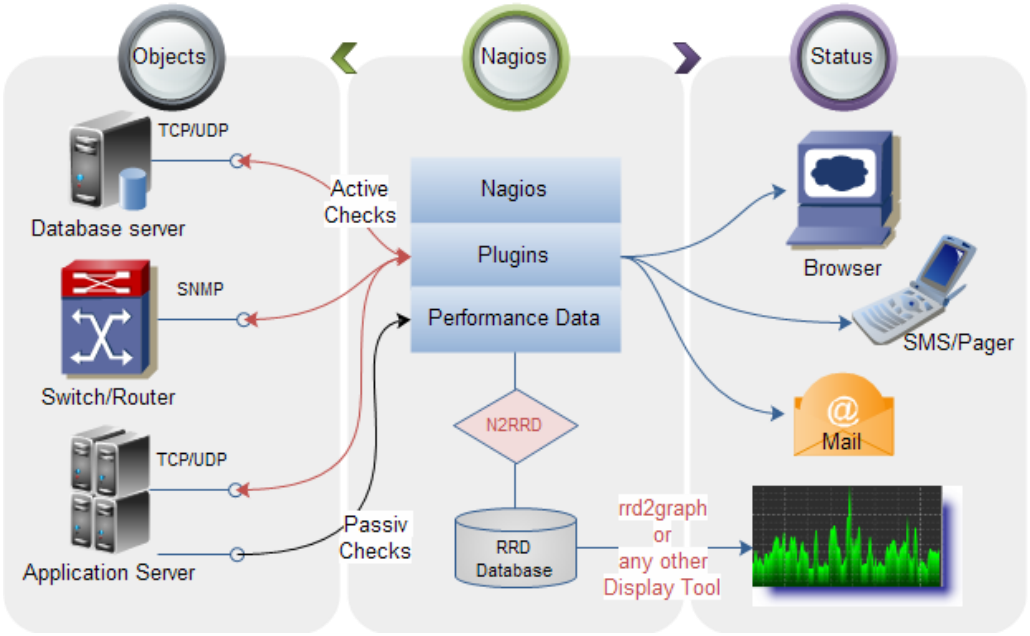


Figure 15: Operating principle of Nagios[55]

Features overview:

- Monitoring of network services (SMTP, POP3, HTTP, NNTP, PING, etc.)
- Monitoring of host resources (processor load, disk usage, etc.)
- Simple plugin design that allows users to easily develop their own service checks
- Parallelized service checks
- Ability to define network host hierarchy using "parent" hosts, allowing detection of and distinction between hosts that are down and those that are unreachable
- Contact notifications when service or host problems occur and get resolved (via email, pager, or user-defined method)
- Ability to define event handlers to be run during service or host events for proactive problem resolution
- Automatic log file rotation
- Support for implementing redundant monitoring hosts
- Optional web interface for viewing current network status, notification and problem history, log file, etc.

[54] http://www.nagios.com/products/nagioscore/
[55] Figure taken from http://en.wikipedia.org/wiki/Nagios

Nagios provides a support forum[56] and an exchange site[57] for the community. Nagios Core is released as open source under the GNU General Public License (GPL) Version 2 license[58].

4.3 Data Integration Capabilities Solutions

4.3.1 Talend Open Studio for Data Integration

Talend Open Studio for Data Integration[59, 60] provides a set of data integration tools to access, transform and integrate data from any business system in real time or batch to meet both operational and analytical data integration needs.

The product provides a graphical business modeller tool for designing business logic for data-intensive applications with data flow sequencing using components and connectors (see Figure 16). The product provides 450+ native database and storage connectivity components[61] that allow users to connect to almost any data source, including databases and data warehouses such as such as Amazon S3, DB2, Ingres, JDBC, Microsoft SQL Server, MySQL, Oracle and PostgreSQL.

There is an active community around the tool and the Talend teams are willing to answer any question on their forum[62]. The product is released as open source [63] under the GNU Lesser General Public License (LGPL) Version 3 license[64].

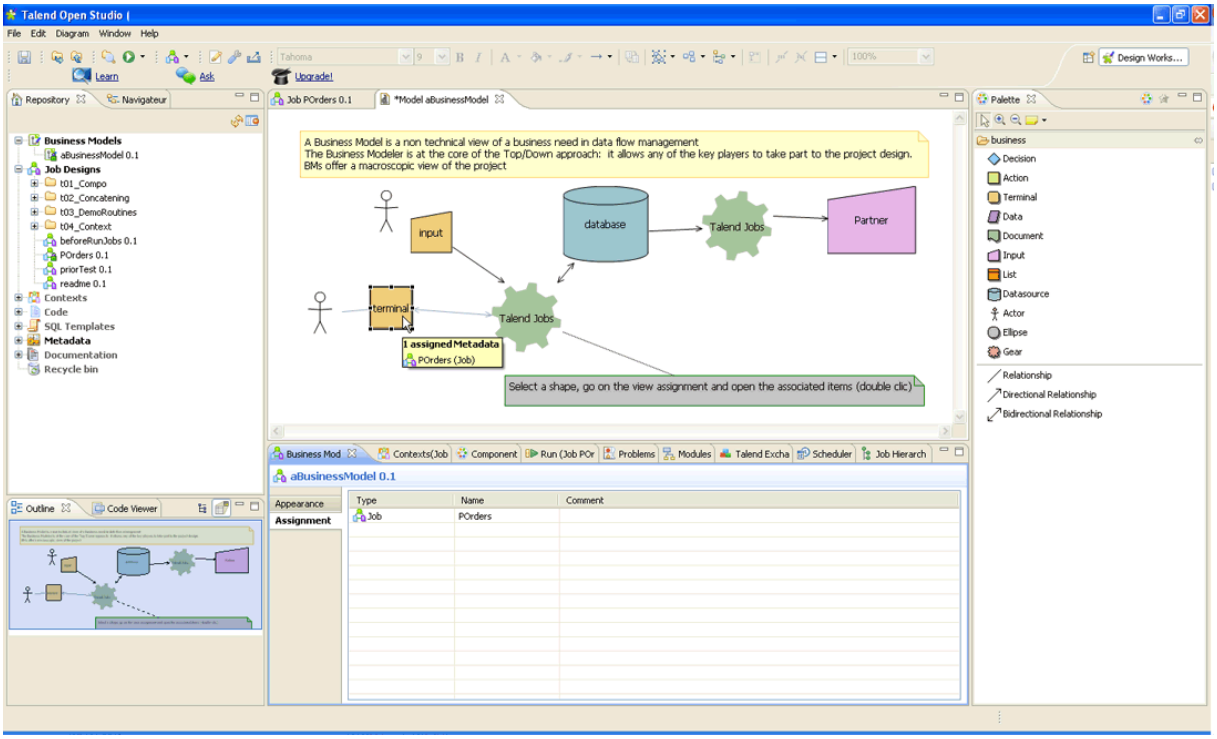


Figure 16: Talend Open Studio for Data Integration

[56] http://support.nagios.com/forum/
[57] http://exchange.nagios.org/
[58] http://sourceforge.net/p/nagios/nagioscore/ci/master/tree/LICENSE
[59] http://www.talend.com/products/data-integration
[60] http://www.talend.com/download/data-integration
[61] http://www.talend.com/products/specifications-data-integration
[62] http://www.talendforge.org/forum/
[63] http://www.talendforge.org/trac/tos/
[64] http://www.gnu.org/licenses/lgpl.html

4.3.2 OpenRefine

OpenRefine[65] is a tool for working with messy data, cleaning it, transforming it from one format into another, extending it with Web Services and linking to databases. The main features of the tool are:[66]

- Importing (formats TSV, CSV, Excel, XML, RDF as XML, JSON, Google Spreadsheets and RDF N3 triples)
- Filtering and faceting (exploring data by applying multiple filters)
- Editing cells, columns and rows (see Figure 17)
- Editing with Google Refine Regular Expression Language (GREL)[67]
- Exporting (formats TSV, CSV, Excel, HTML tables and JSON)
- History (undo and redo)
- Reconciliation against FreeBase[68] schemas using schema alignment dialogs
- Extending data calling Web Services

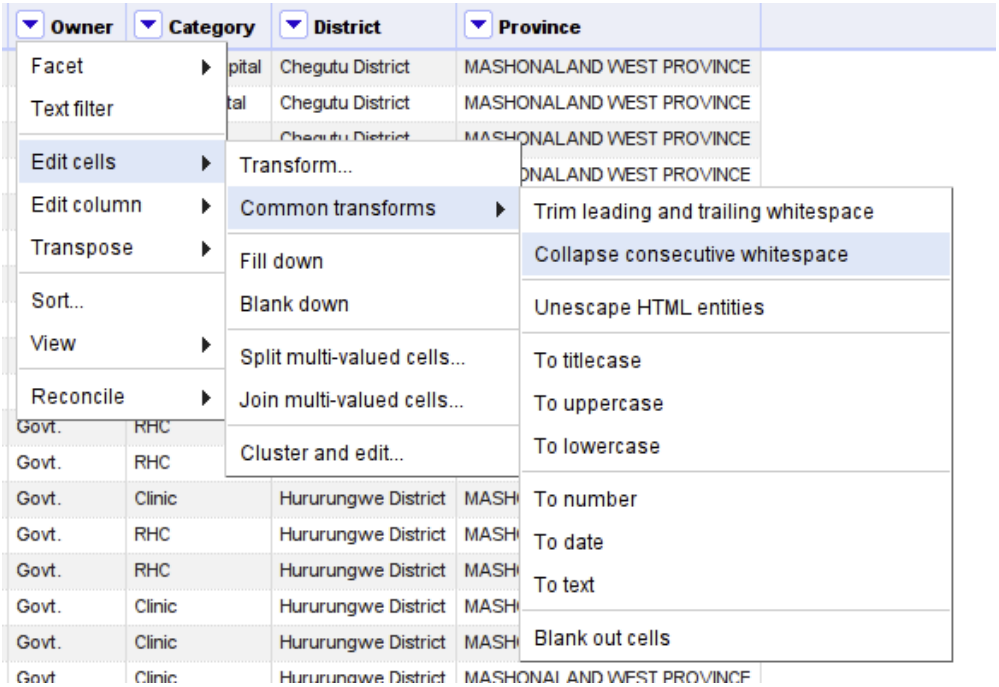


Figure 17: Edit cells in OpenRefine – Common transformations[69]

OpenRefine was originally developed by Google and thus named Google Refine. In July 2010 the product was renamed OpenRefine as it transitioned to a community-supported product[70]. OpenRefine is licensed under an open source Google license[71] with dependencies to a number external open source licenses.

There are several extensions for OpenRefine. Among them, the most relevant for this project would be:

[65] http://openrefine.org/
[66] https://github.com/OpenRefine/OpenRefine/wiki/Documentation-For-Users
[67] https://github.com/OpenRefine/OpenRefine/wiki/Google-refine-expression-language
[68] http://www.freebase.com/
[69] Figure taken from http://schoolofdata.org/handbook/recipes/cleaning-data-with-refine/
[70] http://openrefine.org/community.html
[71] https://github.com/OpenRefine/OpenRefine/blob/master/LICENSE.txt

- CKAN storage extension[72] which allows RDF data to be registered and uploaded on CKAN storage.
- RDF Refine[73] which is used to reconcile and link data (against SPARQL endpoints, or search for related RDF datasets) or export data as RDF format.

4.3.3 Karma

Karma[74] is an information integration tool that enables users to quickly and easily integrate data from a variety of data sources including databases, spreadsheets, delimited text files, XML, JSON, KML and Web APIs. Users integrate information by modelling it according to an ontology of their choice using a graphical user interface (see Figure 18) that automates much of the process.

Karma learns to recognize the mapping of data to ontology classes and then uses the ontology to propose a model that ties together these classes. Once the model is complete, users can published the integrated data as RDF or store it in a database.

Karma is a research prototype developed by the Information Sciences Institute at the University of Southern California (USC)[75] and does not seem to be used by a large community outside of the research group at USC.

Karma is available as open source[76] under the Apache License 2.0 license.

Figure 18: Modelling data in Karma[77]

4.3.4 Cascading

Cascading[78] is an application framework for Java developers to simply develop robust data analytics and data management applications on Apache Hadoop[79]. Apache Hadoop is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

Cascading provides APIs (see Figure 19) for:

- Data processing, enabling complex data flows and data-oriented frameworks.
- Data integration, enabling creation and testing of integration points.
- Process scheduler, allowing scheduling units of work from 3rd party applications.

[72] http://lab.linkeddata.deri.ie/2011/grefine-ckan/
[73] http://refine.deri.ie/
[74] http://www.isi.edu/integration/karma/
[75] http://www.isi.edu/integration/
[76] https://github.com/InformationIntegrationGroup/Web-Karma
[77] Figure taken from https://github.com/InformationIntegrationGroup/Web-Karma/wiki/Modeling-Data
[78] http://www.cascading.org/
[79] http://hadoop.apache.org/

Community support is provided through a mailinglist, IRC and GitHub[80]. Cascading is released as open source under the Apache License 2.0 license[81].

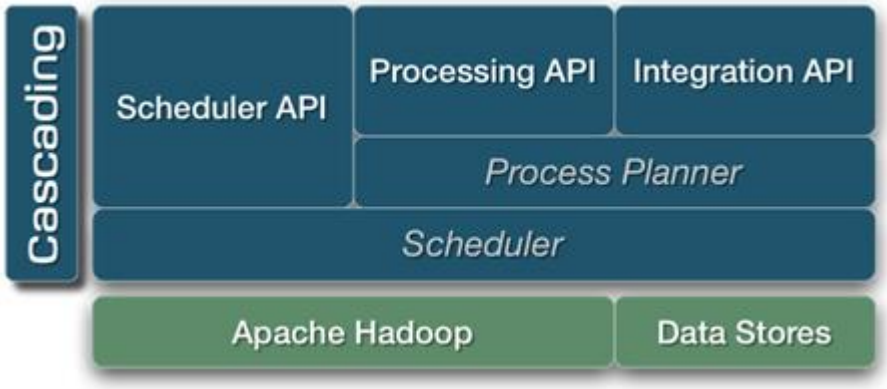


Figure 19: Cascading architecture[82]

4.3.5 Data Pipes

Data Pipes[83] is a service to provide streaming, "pipe-like" data transformations on the web – things like deleting rows or columns, find and replace, head, grep, etc. It provides a REST API that supports the following data transformation operations:

- none (aka raw) = no transform but file parsed
- csv = parse / render csv
- head = take only first X rows
- tail = take only last X rows
- delete = delete rows
- strip = delete all blank rows
- grep = filter rows based on pattern matching
- cut = select / delete columns
- replace = find and replace (not yet implemented)
- html = render as viewable HTML table

Data Pipes seems like a small tool project developed and used by the Open Knowledge Foundation Labs (OKFL). Data Pipes is released as open source under the MIT license[84].

[80] http://www.cascading.org/support/
[81] https://github.com/Cascading/cascading/blob/2.5/LICENSE.txt
[82] Figure taken from http://www.cascading.org/about-cascading/
[83] http://datapipes.okfnlabs.org/
[84] https://github.com/okfn/datapipes/blob/master/LICENSE.md

5 Summary and Outlook

This document provided an overview of the DaPaaS Platform, introduced the relevant roles played in the DaPaaS context and outlined a set of requirements for the DaPaaS Platform from the perspectives of the key roles. Furthermore, the document focused on the PaaS aspect of the platform and provided an initial architecture design for the Platform Layer of the DaPaaS Platform. A state-of-the-art overview of relevant solutions and technologies for the Platform Layer has been presented.

Following the technology evaluation performed as part of this document, the following remarks are to be considered for the implementation phase of the DaPaaS Platform, with a particular focus on the Platform Layer:

- Docker is a promising solution to be reused as an application packaging system for DaPaaS. It has a very active user community and looks like a very promising application container technology which is used by other open source PaaS solutions such as Cocaine and Deis.
- Docker together with either Cocaine or Deis are promising solutions for the implementation of the deployment and hosting environment of the Platform Layer. Cocaine and Deis provides the ability to deploy and run the applications packaged by Docker. Further investigation is needed, i.e., setting up a test infrastructure and experimenting with writing and deploying sample applications, to decide which provides the better foundation. There are differences in the capabilities offered by the two solutions, e.g., Deis provides support for user accounts and SSH public key for authentication and Cocaine provides a publish-subscribe notification service in development that one can build on.
- Talend Open Studio for Data Integration and OpenRefine for data cleaning and integration are promising frameworks for addressing the data integration aspect of the DaPaaS Platform Layer. However further investigation is needed to see if the tools should be fully integrated, whether only a useful subset of the functionality are exposed through DaPaaS-developed frontends, or whether they should be used as inspiration for relevant aspects of the Platform Layer implementation.
- Ansible, Puppet and Chef are IT automation tools that can be used to administrate and manage a set of nodes from a single place, typically the Instance Operator. All three have strong communities and the choice for one of them may depend on the technology choice for the PaaS solution. For example, the Deis PaaS solution already leverages Chef.
- Nagios Core is a monitoring engine that, with necessary adaptations, is relevant to and may be used by the Instance Operator to monitor cloud infrastructure resources where the DaPaaS platform will be deployed.
- DaPaaS-specific extensions to the user management and access control will be implemented on top of the PaaS solution that is chosen, extending it with user profile, and dataset and app access control. Application monitoring in DaPaaS will be coupled to user quotas and policies.
- The features offered by OpenCivic should be considered for the implementation of Catalog features of the Platform Layer. However, since it is based on Drupal which is an open source content management system tailored for creating websites and not PaaS, the features of OpenCivic will likely only be an inspiration for the implementation of the Catalog in Platform Layer.
- For the DaPaaS-specific data workflows services, Cascading is probably too generic for the needs of DaPaaS. It is unclear if we can easily benefit from it as compared to OpenRefine, Talend or even Karma, and further investigation is needed to decide on what solutions can be used for the data workflows.

The above remarks will be taken into consideration for the implementation of the first prototype of the DaPaaS platform, due at M12. In parallel, we will monitor the development of the integrated DaaS/PaaS commercial / closed source solutions, in particular Windows Azure Marketplace, Datameer, and Splunk (briefly introduced in Appendix A).

6 Appendix A: Commercial / Closed Source Integrated DaaS & PaaS Solutions

In this section we briefly introduce some relevant commercial / closed source offerings that fit within the as-a-service platform for load/store/analysis/visualization/publication of data and development/hosting of 3rd party apps. Whereas DaPaaS targets an open source environment for data publishing/hosting and apps development/hosting, the commercial / closed source offerings briefly presented here cover functionalities that are relevant to the overall requirements outlined at the beginning of this document.

6.1 Datameer

Datameer (http://www.datameer.com) was founded in 2009 by some of the original contributors to Apache Hadoop, which is an open source software framework for storage and large-scale processing of datasets. Datameer provides analytics software (see http://www.datameer.com/product/) on top of the Hadoop framework that allows to integrate, analyze and visualize data. The software application comes in three versions: 1) use on your desktop computer, 2) install on your infrastructure as a server, 3) use a provided enterprise instance with your data.

Datameer is an analytics application natively built on Hadoop, focused on leveraging the linear scalability and flexibility of Hadoop for data analytics. Its product is focused on data integration (structured such as relational data, tabular, etc, as well as unstructured data such as social data, email, log files, etc.), multi-device drag-and-drop-style data visualization, and data management (data import, export, data links, storage, data partitioning, compression, etc.).

Datameer is built on open standards/technologies, with a focus on Hadoop and HTML5. It offers extensibility features in the form of plugins. Datameer includes an SDK for writing custom plugins for import, export, functions, and visualizations (documentation is available at http://documentation.datameer.com/documentation/). Datameer's components are exposed through REST APIs.

Datameer provides an applications market at http://www.datameer.com/apps. At time of this writing the application store consists of 48 apps.

6.2 Splunk

Splunk (http://www.splunk.com) shipped its first software in 2006 designed to manage unstructured data generated by machines (websites, applications, servers, networks, mobile devices, sensors and RFID assets). The Splunk Enterprise software product provides features for collecting and indexing any machine data, including the capability to handle massive live datastreams, statistical analysis and real-time dashboards. The software can also be provides as a Software-as-a-Service in the Cloud. This version of the product is named Splunk Cloud. The company also provides a software product named Hunk for analyzing and visualizing data in Hadoop.

Splunk was initially developed to allow organizations to search and analyze data generated by applications, servers and network devices in IT infrastructures. It includes capabilities for developing applications and provides an applications store at http://apps.splunk.com/. At time of this writing the application store consists of 488 apps.

6.3 Windows Azure Marketplace

Windows Azure Marketplace (http://datamarket.azure.com/), supported by Microsoft, is an “online market for customers and partners to share, buy, and sell finished Software-as-a-Service applications, building block components and premium datasets.” The marketplace comes with a set of APIs targeted at developers to work with datasets in the same way on many different platforms, enabling the developers to develop applications for desktop, Web, mobile, and other clients. The marketplace also offers developers common security, billing, auditing, and authentication mechanisms. At time of this

writing the application store consists of 644 apps. Further info about the Windows Azure Marketplace can be found in Deliverable D1.1.

6.4 GoodData

GoodData (http://www.gooddata.com/) offers a business intelligence platform. It provides support for extract, transform and load (ETL) processes, connectors to various data sources, scalable data storage, a data analytics engne, reporting and various forms of data visualization.

GoodData was primarily designed for the cloud and is built on technologies such as Vertica, MongoDB, Cassandra, NetApp, and Rackspace.

GoodData is an open platform in the sense that it can be embedded in existing applications, or allows applications to be built on top of the platform, through a set of RESTful APIs.

6.5 Tableau Software

Tableau software (http://www.tableausoftware.com) provides a set of interactive data visualization products focused on business intelligence, ranging from desktop to hosted solutions. The focus of Tableau is on providing usability and ease of use for its software for business intelligence with a strong emphasis on the data visualization capabilities. Tableau offers four main products: Tableau Desktop, Tableau Server, Tableau Online, and Tableau Public.

Tableau Desktop is a drag&drop-style data analysis and visualization desktop tool. It provides data connection capabilities, data visualization, and creation of interactive dashboards.

Tableau Server is a business intelligence application that provides browser-based analytics. It comes with a Web-based dashboard where users can import and integrate data, and analyse and visualize it.

Tableau Online is a hosted version of Tableau Server. It provides a central place to manage data, data sources and metadata, and focuses on scalability.

Tableau Public is delivered as a service and aims at creating interactive visuals and publishing them without the help of programmers or IT. It targets organizations that want to enhance their websites with interactive data visualizations. It offers carious visualization types, such as maps, bar and line charts, lists, heat maps, etc.

6.6 Infochimps

Infochimps (http://www.infochimps.com/) provides a managed cloud service called Infochimps Cloud (http://www.infochimps.com/infochimps-cloud/overview/) that streamlines building and managing complex Big Data environments, and aims to make it faster and less complex to develop and deploy Big Data applications. The platform provides capabilities for data streaming, storage, queries and administration.