D6.2.1

CONTENT-DEDICATED PLATFORM AS A SERVICE - INITIAL RELEASE

September 2013

ABSTRACT

This document accompanies the prototype named “Content-dedicated Platform-as-a-Service”: it first presents the notion of Platform-as-a-Service and explains the rationale for the adoption of a Platform-as-a-Service to simplify the development of server-side applications in the FI-CONTENT 2 project. It then shows how the current initial version of the prototype works.

This document is a deliverable of the FI-CONTENT 2 integrated project supported by the European Commission under its FP7 research funding programme, and contributes to the FI-PPP (Future Internet Public Private Partnership) initiative.

DISCLAIMER

DELIVERABLE DETAILS

[Full project title]:	Future media Internet for large-scale CONTent experimENTation 2
[Short project title]:	FI-CONTENT 2
[Contract number]:	603662
[WP n°]:	WP6 – Technology Enablers
[WP leader]:	Mario LOPEZ-RAMOS, Thales
[Deliverable n°]:	D6.2.1
[Deliverable title]:	Content-dedicated Platform as a Service - initial release
[Deliverable nature]:	Report (R)
[Dissemination level]:	Public (PU)
[Contractual delivery date]:	M6 – September 2013
[Actual delivery date]:	1 October 2013
[Editor]:	Mario LOPEZ-RAMOS, Thales
[Internal Reviewers]:	Dirk KRAUSE, Pixelpark / Marcel Lancelle, ETHZ
[Suggested readers]:	Web application developers
[Keywords]:	PaaS, Cloud, DevOps, CloudFoundry
[File name]:	FI-CONTENT 2_WP6-002_D6.2.1_V1.0

EXECUTIVE SUMMARY

The present report is aimed to be associated with the prototype “Content-dedicated Platform-as-a-Service”. In the first part of this document, we describe the principle of Platform-as-a-Service and detail how the use of a Platform-as-a-Service can make server-side applications development easy in the scope of the FI-CONTENT 2 project. Then, in the second part of the document, we describe the functioning of the current prototype version.

LIST OF AUTHORS

Organisation	Author
Thales	Mario LOPEZ-RAMOS

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

LIST OF FIGURES

1 - Introduction

Within FI-CONTENT 2, WP6 delivers the technical components common to the Social Connected TV, the Smart City Guide and the Gaming Platform. Such components are targeted at developers of applications: those in WP2, WP3 and WP4, those from the Open Calls, as well as third parties attracted by WP5.

Within WP6, task 6.2 (“Content-dedicated Platform-as-a-Service”) has the objective of delivering to developers:

- a running PaaS for the developers of content-specific server-side applications
- build, deployment, monitoring and management tools to simplify development and operations

Within task 6.2, this deliverable describes the initial release of the Platform-as-a-Service customized and enhanced for FI-CONTENT 2 needs.

2 - CLOUD COMPUTING AND PAAS

2.1 - Cloud Computing service models

The services available to a Cloud Consumer are typically Infrastructure-as-a-Service (Iaas), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). These services can be recursive: one can be delivered on top of another. The following picture illustrates the positioning of well-known examples of each of them.

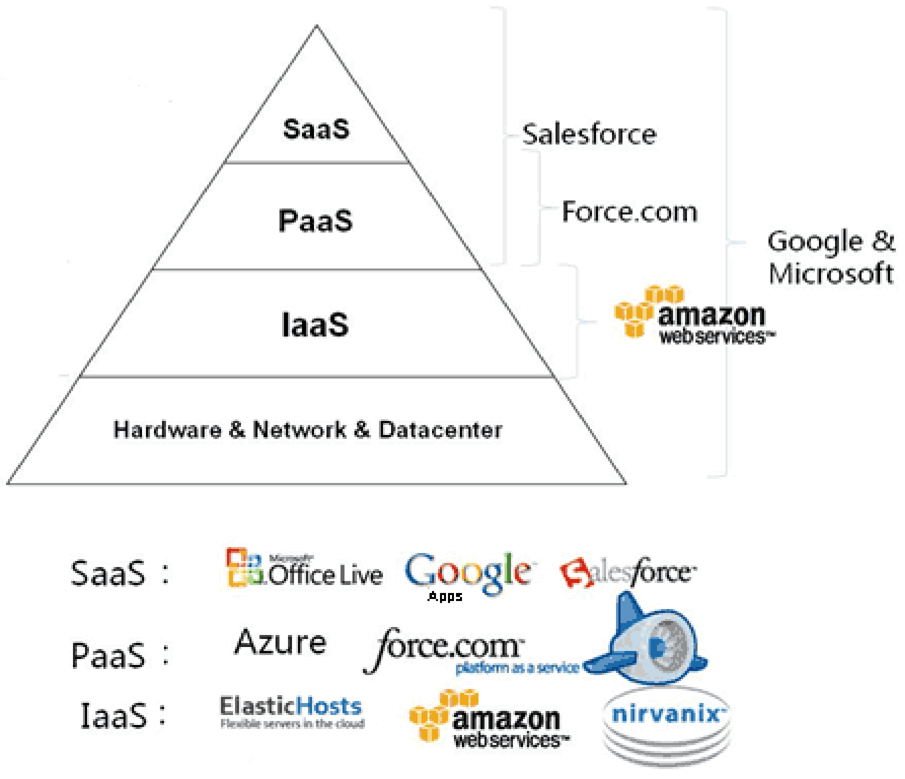


Figure 1 - Cloud Computing service models

SaaS, PaaS and IaaS are basic concepts in the cloud landscape. NIST (National Institute of Standards and Technology) describes them as follows:

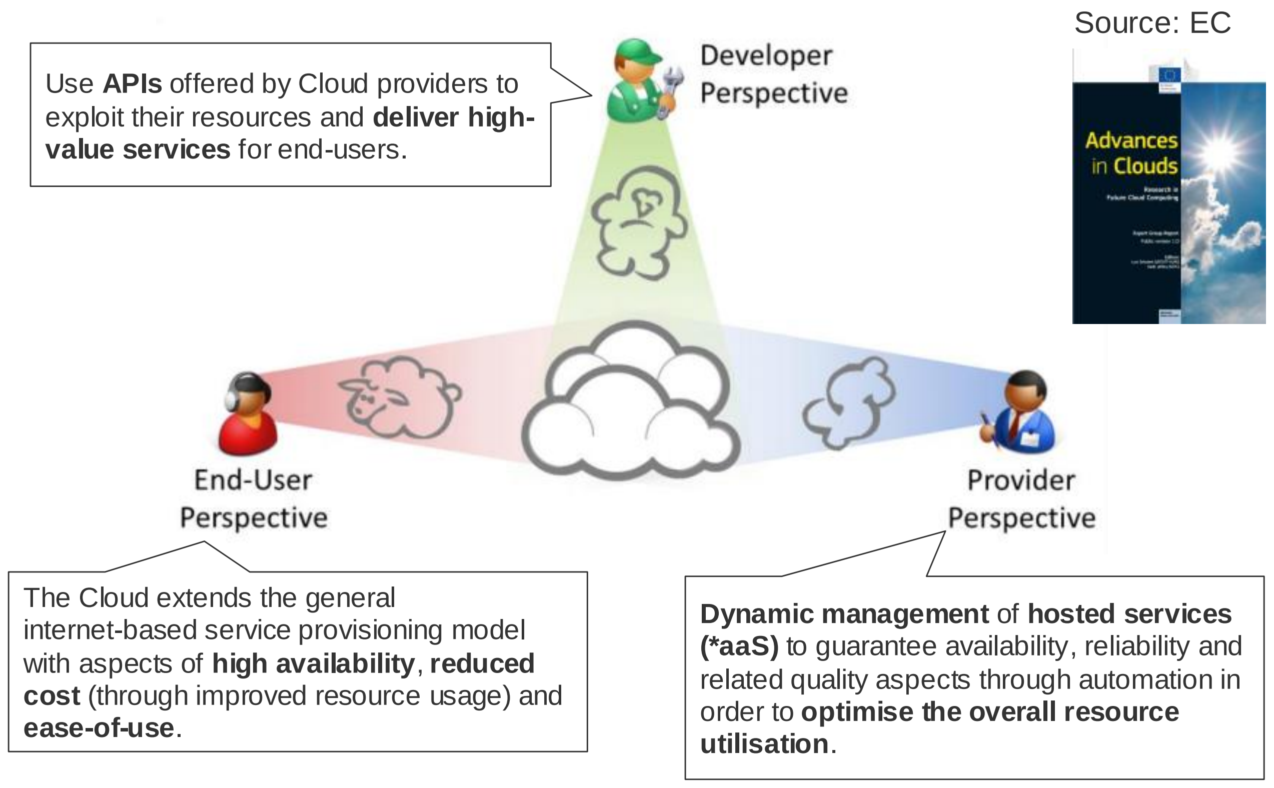
- SaaS are software applications made accessible via a network to the SaaS consumers. The consumers of SaaS can be organizations that provide their members with access to software applications, end users who directly use software applications, or software application administrators who configure applications for end users. SaaS consumers can be billed based on the number of end users, the time of use, the network bandwidth consumed, the amount of data stored or duration of stored data.
- Cloud consumers of PaaS can employ the tools and execution resources provided by cloud providers to develop, test, deploy and manage the applications hosted in a cloud environment. PaaS consumers can be application developers who design and implement application software, application testers who run and test applications in cloud-based environments, application deployers who publish applications into the cloud, and application administrators who configure and monitor application performance on a platform. PaaS consumers can be billed according to processing, database storage and network resources consumed by the PaaS application, and the duration of the platform usage.
- Consumers of IaaS have access to virtual computers, network-accessible storage, network infrastructure components, and other fundamental computing resources on which they can deploy and run arbitrary software. The consumers of IaaS can be system developers, system administrators

and IT managers who are interested in creating, installing, managing and monitoring services for IT infrastructure operations. IaaS consumers are provisioned with the capabilities to access these computing resources, and are billed according to the amount or duration of the resources consumed, such as CPU hours used by virtual computers, volume and duration of data stored, network bandwidth consumed, number of IP addresses used for certain intervals.

In the context of FI-CONTENT 2:

- IaaS services are provided by the FI-WARE Testbed and FI-LAB
- PaaS services will be provided by task 6.2, based on FI-WARE Cloud GEs
- SaaS services should be the applications delivered by end users, based either on FI-WARE's IaaS or Content PaaS

From a technical perspective, Cloud Computing platforms can be seen from different perspectives depending on the stakeholder. The following picture summarizes it:



Use **APIs** offered by Cloud providers to exploit their resources and **deliver high-value services** for end-users.

The Cloud extends the general internet-based service provisioning model with aspects of **high availability**, **reduced cost** (through improved resource usage) and **ease-of-use**.

Dynamic management of **hosted services (*aaS)** to guarantee availability, reliability and related quality aspects through automation in order to **optimise the overall resource utilisation**.

Figure 2 - Cloud Computing, different stakeholder perspectives

The aim of FI-CONTENT 2 is not to deliver final applications for end-users, but to simplify the development of new applications to generate revenue streams from the use of the platform. Therefore FI-CONTENT 2 partners are providers and the target stakeholders are the developers.

2.2 - What is a Platform-as-a-Service?

PaaS represents a shift in responsibility compared to the IaaS model:

- PaaS consumers (i.e. application developers) are no longer responsible of deploying and configuring VMs, OSs and runtimes, but only applications.
- PaaS providers are responsible of shared services, and can ensure high availability, scalability, disaster recovery, etc. out of the box. They design such services so that:
    - they offer a self-service API
    - scale on demand

- isolate tenants

The following classification initiated by Gartner tries to categorize the different services provided by existing PaaS platforms, both in the public and the private landscape.

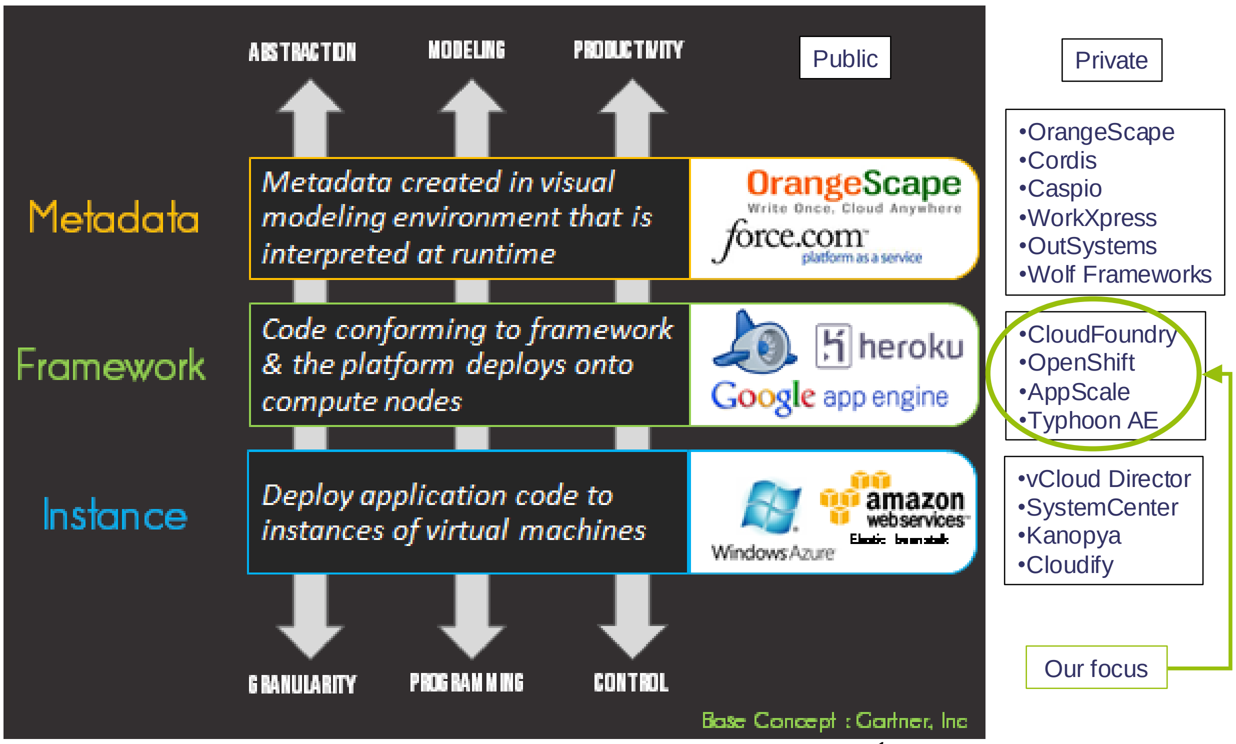


Figure 3 - Categories of PaaS providers[1]

The following subsections analyze the responsibilities and tools typically provided by each of the categories, together with examples of providers.

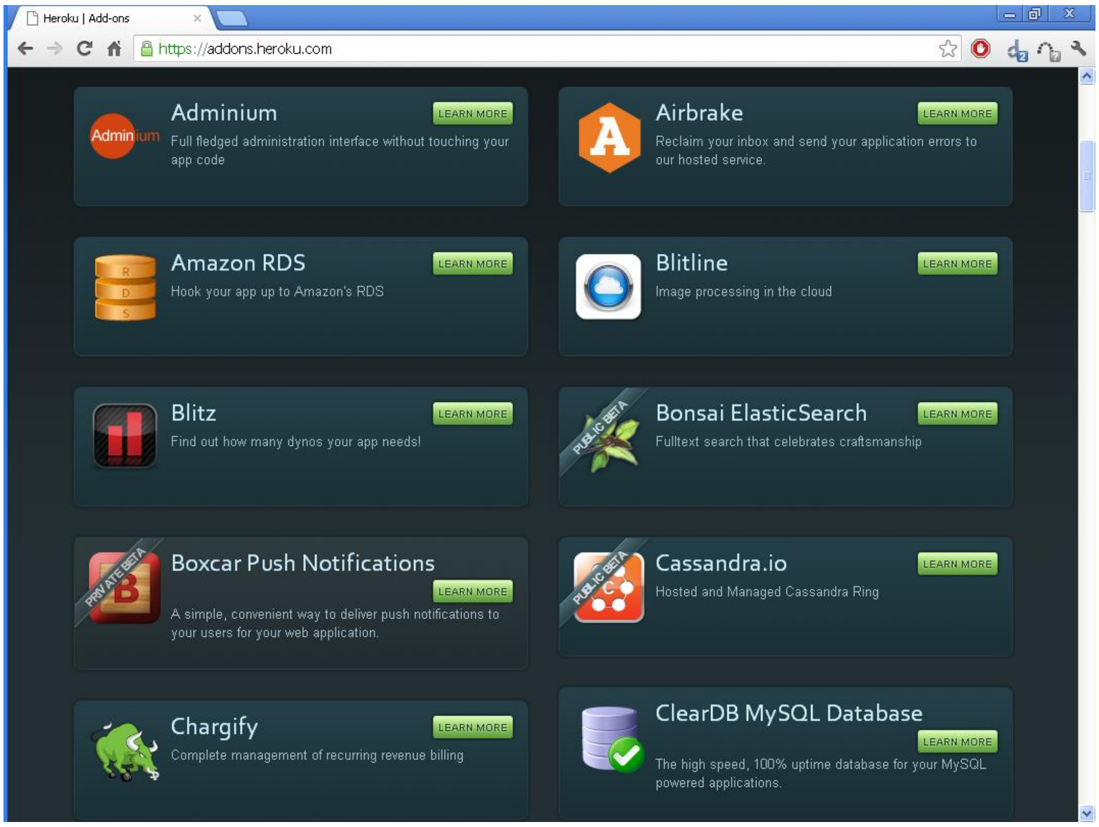
2.2.1 - Instance PaaS (also known as IaaS++ or custom PaaS)

- Target: developers with sysadmin knowledge
- From the developer perspective:
  - Applications rely on automatically generated and deployed VMs that the developer can access and has to manage (configure, update, back-up…).
  - Existing applications can be deployed with no changes. Cloud benefits (high availability, scalability…) require a parallelized architecture and the creation of automation recipes.
- Provided tools:
  - App deployment mechanism (Eclipse plugin…)
  - VMs with runtime environment
  - Service catalog (DB, Auth, task queue...)
  - Tools to add new services to the catalog
  - Separation of responsibility:
    - PaaS customer: VM configuration, OS, runtime containers, technical services, apps
    - PaaS provider: virtualization infrastructure
- Examples:
  - Public: Amazon Beanstalk, Microsoft Azure
  - Private: VMWare vCloud Director, MS SystemCenter

[1] Source :http://blog.orangescape.com/2011/10/18/1265/

2.2.2 - Framework PaaS (commonly called just ‘PaaS’)

- Target: developers with NO system administration knowledge
- From the developer perspective:
  - Runtime choice is limited to a catalog of preconfigured runtimes and services automatically managed.
  - A catalog of services simplifying the development, integration and runtime phases
  - In order to benefit from high availability, elasticity… applications need to be adapted to use the provided framework that forces to create parallelized architectures.
  - Developers don’t have access to the underlying layers (VMs, OS, containers…)
- Provided tools:
  - App deployment mechanism (Eclipse plugin…)
  - Multi-tenant runtime environment
  - Service catalog, which can include:
    - Persistence
    - Data processing, indexation, workers…
    - Logging
    - Messaging (message queues, e-mails, SMS…)
    - Monitoring
    - Caching
    - Media (image, video transformation, streaming…)
    - Billing
    - Software lifecycle management
  - PaaS Service manager: high availability, elasticity…
  - Separation of responsibility:
  - PaaS customers: application code and configuration
  - PaaS provider: VMs, OSs, multi-tenancy containers
- Examples:
  - Public: Google App Engine, Heroku
  - Private: CloudFoundry, OpenShift



Heroku | Add-ons

https://addons.heroku.com

Adminium — LEARN MORE
Full fledged administration interface without touching your app code

Airbrake — LEARN MORE
Reclaim your inbox and send your application errors to our hosted service.

Amazon RDS — LEARN MORE
Hook your app up to Amazon's RDS

Blitline — LEARN MORE
Image processing in the cloud

Blitz — LEARN MORE
Find out how many dynos your app needs!

Bonsai ElasticSearch — LEARN MORE
Fulltext search that celebrates craftsmanship

Boxcar Push Notifications — LEARN MORE
A simple, convenient way to deliver push notifications to your users for your web application.

Cassandra.io — LEARN MORE
Hosted and Managed Cassandra Ring

Chargify — LEARN MORE
Complete management of recurring revenue billing

ClearDB MySQL Database — LEARN MORE
The high speed, 100% uptime database for your MySQL powered applications.

Figure 4 - Catalog of services provided by third-parties in Heroku

2.2.3 - PaaS for non-technical creators or metadata PaaS

- Target: Not « coders ». Business-level power users.
- From the developer perspective:
  - The client doesn't need anything to code (even coding knowledge).
  - He is forced to use the IDE provided and is limited in term of functionalities.
- Provided tools:
  - Domain-specific WYSIWYG IDE
  - Domain-specific test and runtime environment
  - Separation of responsibility:
    - PaaS customers: application model or DSL
    - PaaS provider: VMs, OSs, multi-tenancy containers
- Examples :
  - workXpress, Force.com, Caspio, Cordys, Zoho editor, OrangeScape

Figure 5 - Example of platform for non-technical users for rapid application development

2.3 - Why and what kind of PaaS in FI-CONTENT 2?

In the FI-CONTENT 2 context, the objective of the Social Connected TV, Smart City Guide and Gaming Platforms in FI-CONTENT 2 is to create a “platform effect” by helping third parties deliver content applications to end-users so that it creates a mutual benefit. It can be achieved in a number of different ways depending on the domain and participants by offering:

- Content sharing (e.g. Youtube, Wikipedia…)
- Social networks and community websites (e.g. Facebook, Twitter…)
- Code reuse and improvement (e.g. open-source communities such as github.com)
- Software Development Kits for hardware platforms with an associated marketplace for revenue sharing: smartphones (e.g. iOS, Android), smart TVs…
- Web mashups that enable the combination of information from different sources (e.g. Google Maps mashups)
- Web APIs that provide services through a programmatic interface (e.g. programmableweb.com)
- Cloud hosting services that simplify the on-demand consumption of IT resources

By analyzing the case of the gaming platform developed in WP4, according to feedback from partners, the profiles of target platform developers are mainly:

- Non-expert content creators
- Developers:
    - Client-side (e.g. Android, iOS) application developers
    - Server-side (e.g. HTML5 game, server-side game logic) application developers
- Game platform (SaaS) providers

The tools identified for those types of users are significantly different:

- Non-expert content creators prefer dedicated web UIs for content creation and sharing.
- Expert game developers need a subset of the following tools:
    - Client-side enablers:
        - SDKs for 3D game development such as the popular Unity3D
        - Javascript libraries for 3D, such as XML3D
    - Server-side enablers:

- The most demanded are the APIs that can be remotely consumed from the client: game services such as leaderboards, achievement boards, matchmaking and multi-player.
- Games with more complex server-side logic require a hosting platform enabling the deployment of e.g. web applications and databases.

Specific development tools such as WYSIWYG editors could be created for specific uses (e.g. 3D modelling, virtual world edition, etc.) but WP6 has the objective of providing common tools for all the platforms. Therefore, the most appropriate type of platform is the so-called "Framework PaaS".

The Content-dedicated PaaS will offer developers runtime containers and a catalogue of homogeneous APIs to use:

- Server-side Generic Enablers from FI-WARE and Specific Enablers from FI-CONTENT;
- Services developed for social TV (WP2), Smart City Guide (WP3) and gaming (WP4) applications, which can be combined to deliver new cross-domain applications;
- Tools and services to simplify the process of building, deploying, monitoring and managing content-related applications on top of Cloud infrastructures and Content Delivery Networks so that the appropriate Quality of Service is met.

The following diagram depicts the position of the PaaS in the overall FI-CONTENT 2 environment:

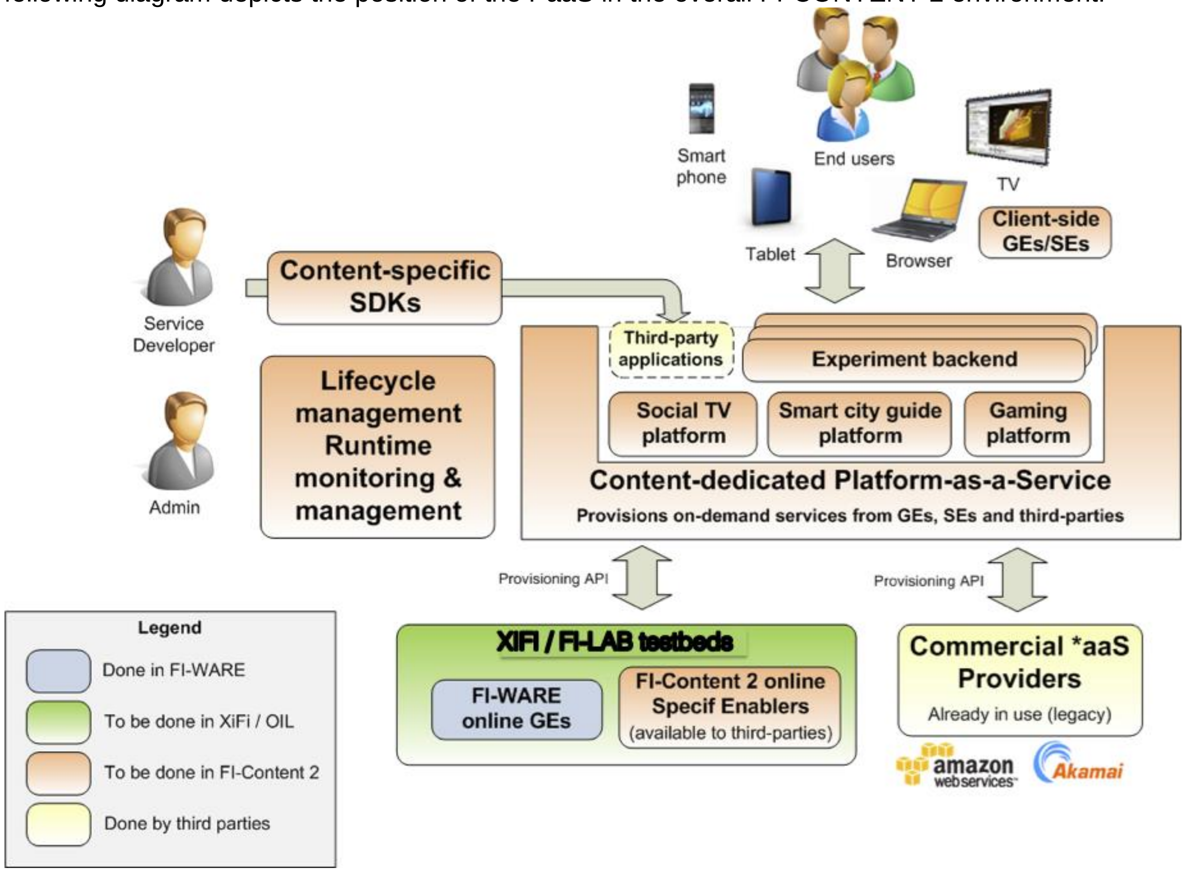


Figure 6 - FI-CONTENT 2 high-level architecture with the PaaS

3 - ARCHITECTURE OF THE SOLUTION

3.1 - CloudFoundry v1 architecture

The CloudFoundry v1 architecture is structured as follows[2]:

- At the core, the app execution engine is the piece that runs your application. It's what launches and manages the Rails, Java, and other language app servers. As your app is scaled up more app execution engines will launch an app server with your code. The way the app execution engine is architected is nice in that it is fairly stand-alone. It can be launched on any suitably configured server, then it connects to the other servers in the PaaS and starts running user applications (the app execution engines can be configured to run a single app per server or multiple). This means that to scale up the PaaS infrastructure itself the primary method is to launch more suitably configured app execution engines.
- The request router is the front door to the PaaS: it accepts all the HTTP requests for all the applications running in the PaaS and routes them to the best app execution engine that runs the appropriate application code. In essence the request router is a load balancer that knows which app is running where. The request router needs to be told about the hostname used by each application and it keeps track of the available app execution engines for each app. Request routers are generally not scaled frequently, in part because DNS entries point to them and it's good practice to keep DNS as stable as possible, and also because a small number of request routers go a long way compared to app execution engines. It is possible, however to place regular load balancers in front of the request routers to make it easy to scale them without DNS changes.
- The cloud controller implements the external API used by tools to load/unload apps and control their environment, including the number of app execution engines that should run each application. As part of taking in new applications it creates the bundles that app execution engines load to run an application. A nice aspect of the cloud controller is that it is relatively policy-free, meaning that it relies on external input to perform operations such as scaling how many app execution engines each application uses. This allows different management policies to be plugged-in.
- A set of services provide data storage and other functions that can be leveraged by applications. In analogy with operating systems these are the device drivers. Each service tends to consist of two parts: the application implementing the service itself, much as MySQL, MongoDB, redis, etc. and a Cloud Foundry management layer that establishes the connections between applications and the service itself. For example, in the MySQL case this layer creates a separate logical database for each application and manages the credentials such that each application has access to its database.
- A health manager responsible for keeping applications alive and ensuring that if an app execution engine crashes the applications it ran are restarted elsewhere.

[2] Source: http://www.rightscale.com/blog/cloud-management-best-practices/cloud-foundry-architecture-and-auto-scaling

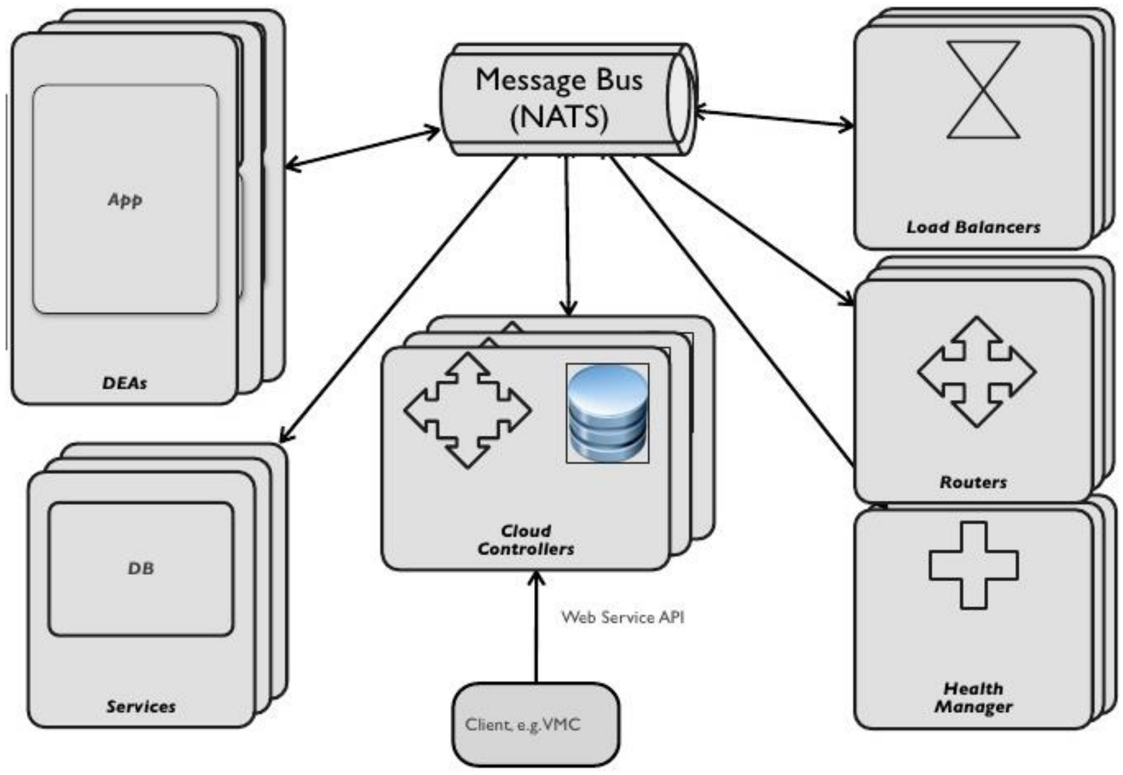


Figure 7 - CloudFoundry v1 Architecture, used for the first release

3.2 - CloudFoundry v2 architecture

The open-source CloudFoundry project has underwent a major upgrade of the architecture. V2 offers a significant number of enhancements, but it is not yet stable. The final release of the Content PaaS will be based on the new version v2.

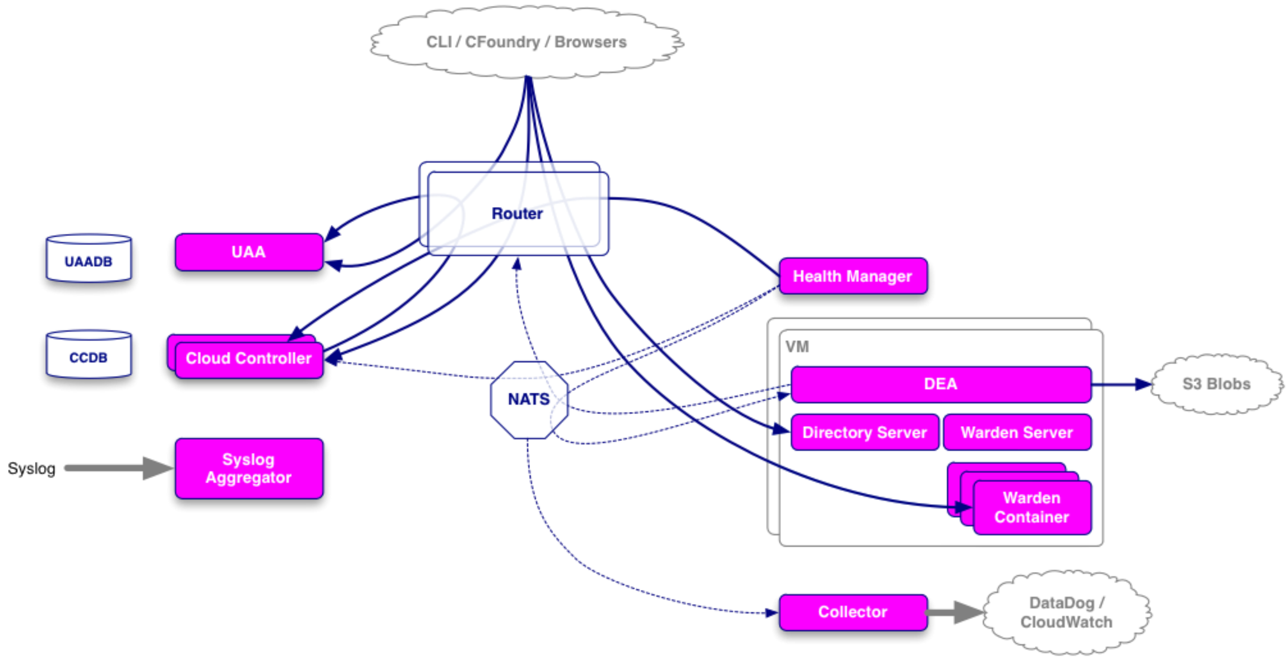


Figure 8 - Architecture of CloudFoundry v2, will be used for the next release

The role of each of the components is detailed hereafter:

- Cloud Controller - Maintains a database with tables for orgs, spaces, apps, services, service instances, user roles, and more.
- Droplet Execution Agent - Manages the lifecycle of an application instance. Tracks started instances and broadcasts state messages.
- Health Manager – Monitors applications and restarts them in case of failure.
- Messaging (NATS) - Lightweight publish-subscribe and distributed queuing messaging system.
- (Go)Router - Routes incoming traffic to the appropriate component, usually the cloud controller or a running application on a DEA node.
- Schemata – Translate messages between new and old components.
- Services - Any type of add-on that can be provisioned alongside your application; for example, a database or an account on a third-party SaaS provider. Cloud Foundry services are implemented with a common architecture that uses gateways and nodes.
- Stacks - Prebuilt file systems, including an operating system, that support running applications with certain characteristics.
- User Account and Authentication (UAA) Server - Identity management service for Cloud Foundry.
- Warden - Provides a simple API for managing isolated, resource-controlled environments.

3.3 - Network and deployment architecture

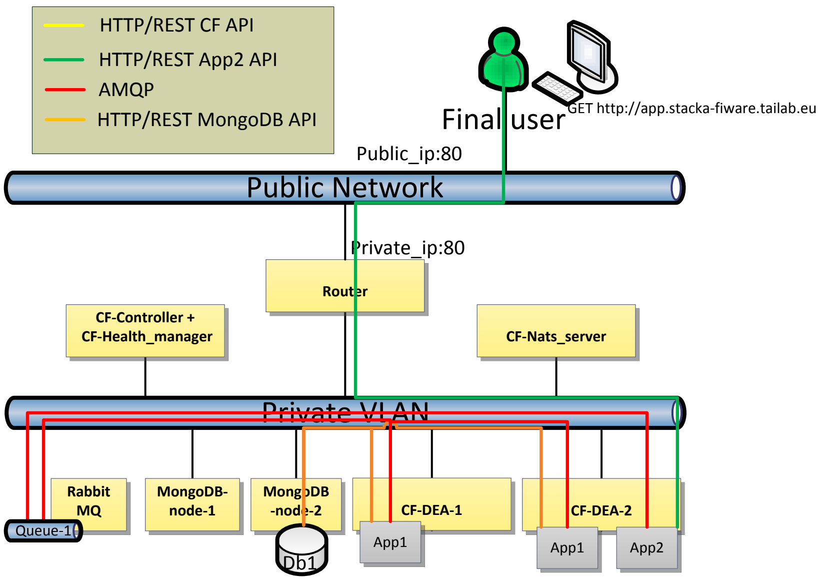


Figure 9 - Networking architecture of the solution

The minimal deployment can be done in one machine, but each of the blocks can be split into several machines. At least 2 are recommended. The current version is hosted in the FI-WARE Cloud testbed and uses 3 different machines.

4 - Initial release of the content PaaS

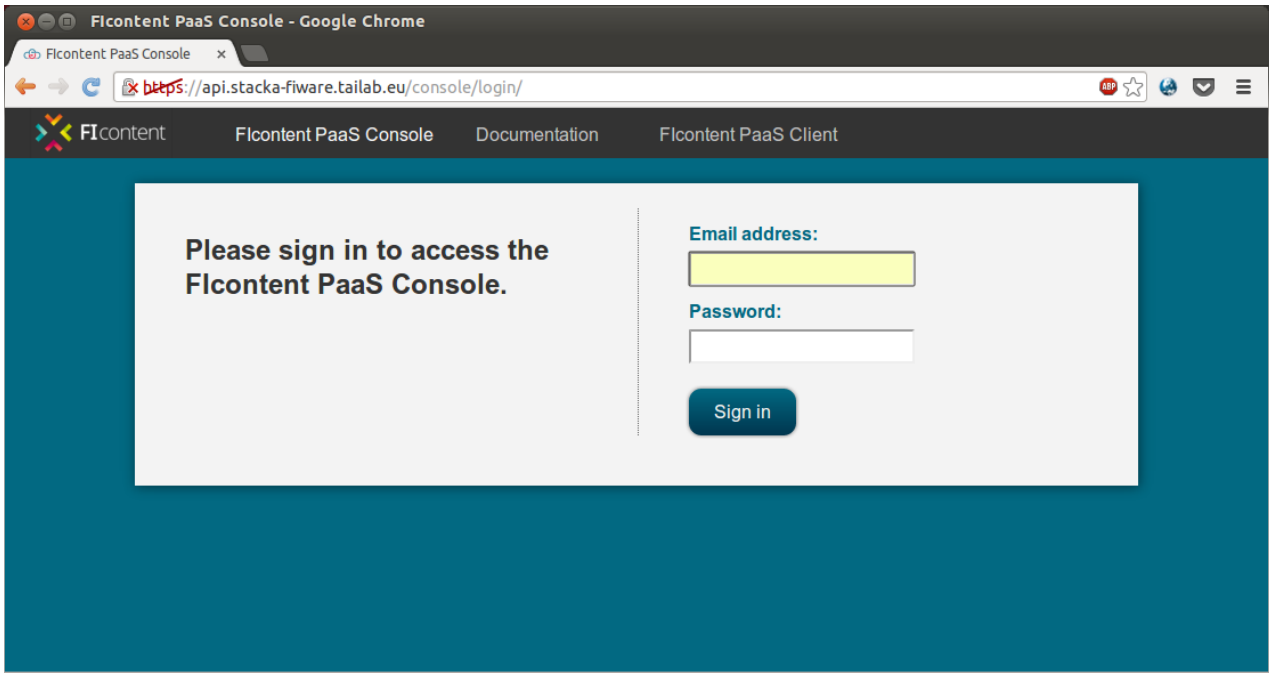
This section shows the running initial release of the prototype, available at https://api.stacka-fiware.tailab.eu/

The initial implementation is based on CloudFoundry v1, packaged by Stackato. It is provided only for testing purposes.

4.1 - Developer (PaaS user) perspective

4.1.1 - Login

The following screenshot shows the initial screen, where users can log in.

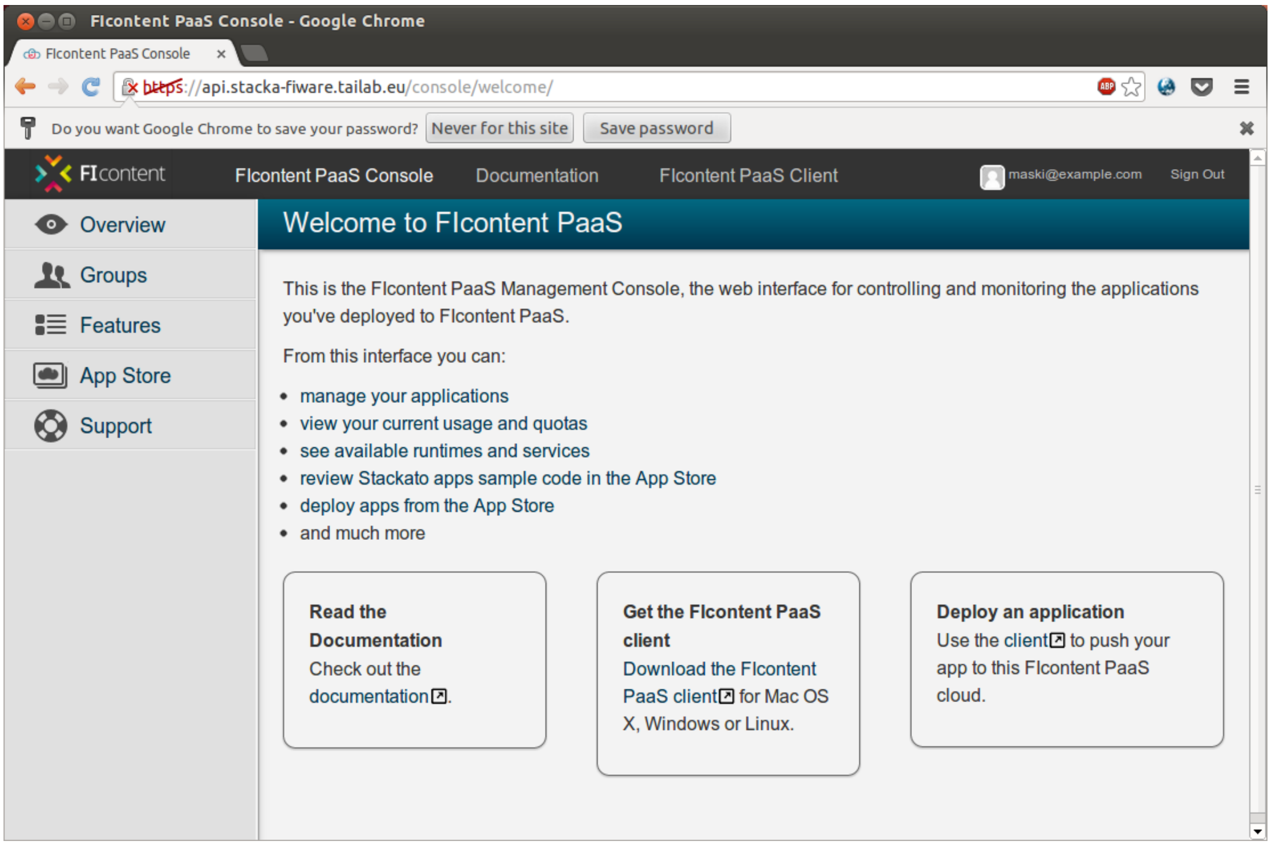


Please sign in to access the FIcontent PaaS Console.

Figure 10 - Login screen

4.1.2 - Welcome screen

The following screenshot shows the welcome page of a developer, where a number of actions are listed:



FIcontent PaaS Console - Google Chrome

https://api.stacka-fiware.tailab.eu/console/welcome/

Do you want Google Chrome to save your password? Never for this site | Save password

Welcome to FIcontent PaaS

This is the FIcontent PaaS Management Console, the web interface for controlling and monitoring the applications you've deployed to FIcontent PaaS.

From this interface you can:

- manage your applications
- view your current usage and quotas
- see available runtimes and services
- review Stackato apps sample code in the App Store
- deploy apps from the App Store
- and much more

Read the Documentation Check out the documentation.

Get the FIcontent PaaS client Download the FIcontent PaaS client for Mac OS X, Windows or Linux.

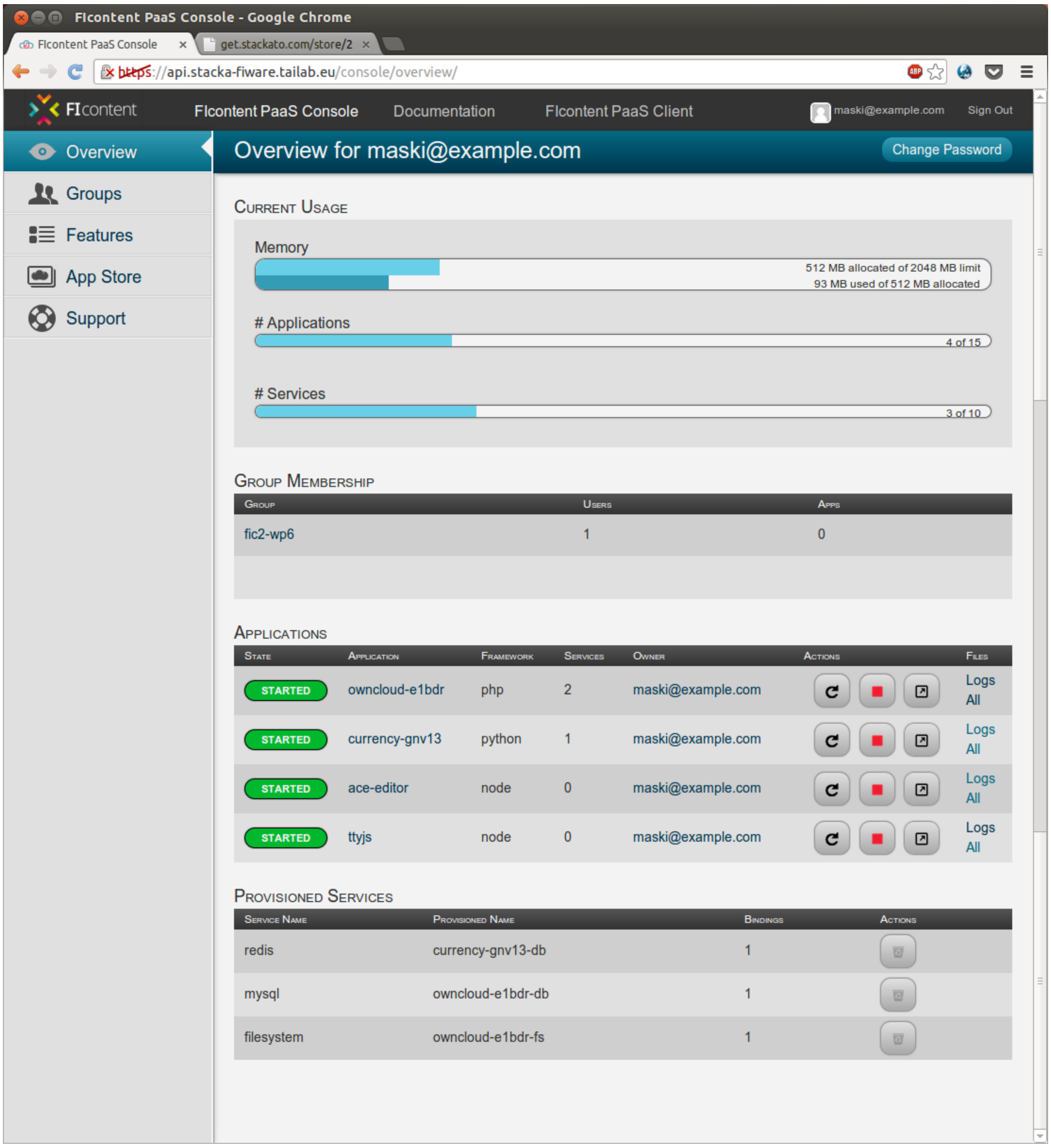
Deploy an application Use the client to push your app to this FIcontent PaaS cloud.

Figure 11 - Welcome screen

4.1.3 - Developer overview

The following screenshot shows the overview of all the applications & services provisioned by a user. This view gives a good understanding of what a PaaS provides:

- different services (redis, mysql, filesystem) can be provisioned on-demand for the deployed applications
- for each application, there is an identified runtime and 0 or more bound services



Overview for maski@example.com

CURRENT USAGE

Memory
512 MB allocated of 2048 MB limit
93 MB used of 512 MB allocated

Applications
4 of 15

Services
3 of 10

GROUP MEMBERSHIP

GROUP	USERS	APPS
fic2-wp6	1	0

APPLICATIONS

STATE	APPLICATION	FRAMEWORK	SERVICES	OWNER	ACTIONS	FILES
STARTED	owncloud-e1bdr	php	2	maski@example.com		Logs All
STARTED	currency-gnv13	python	1	maski@example.com		Logs All
STARTED	ace-editor	node	0	maski@example.com		Logs All
STARTED	ttyjs	node	0	maski@example.com		Logs All

PROVISIONED SERVICES

SERVICE NAME	PROVISIONED NAME	BINDINGS	ACTIONS
redis	currency-gnv13-db	1	
mysql	owncloud-e1bdr-db	1	
filesystem	owncloud-e1bdr-fs	1	

Figure 12 - Developer overview

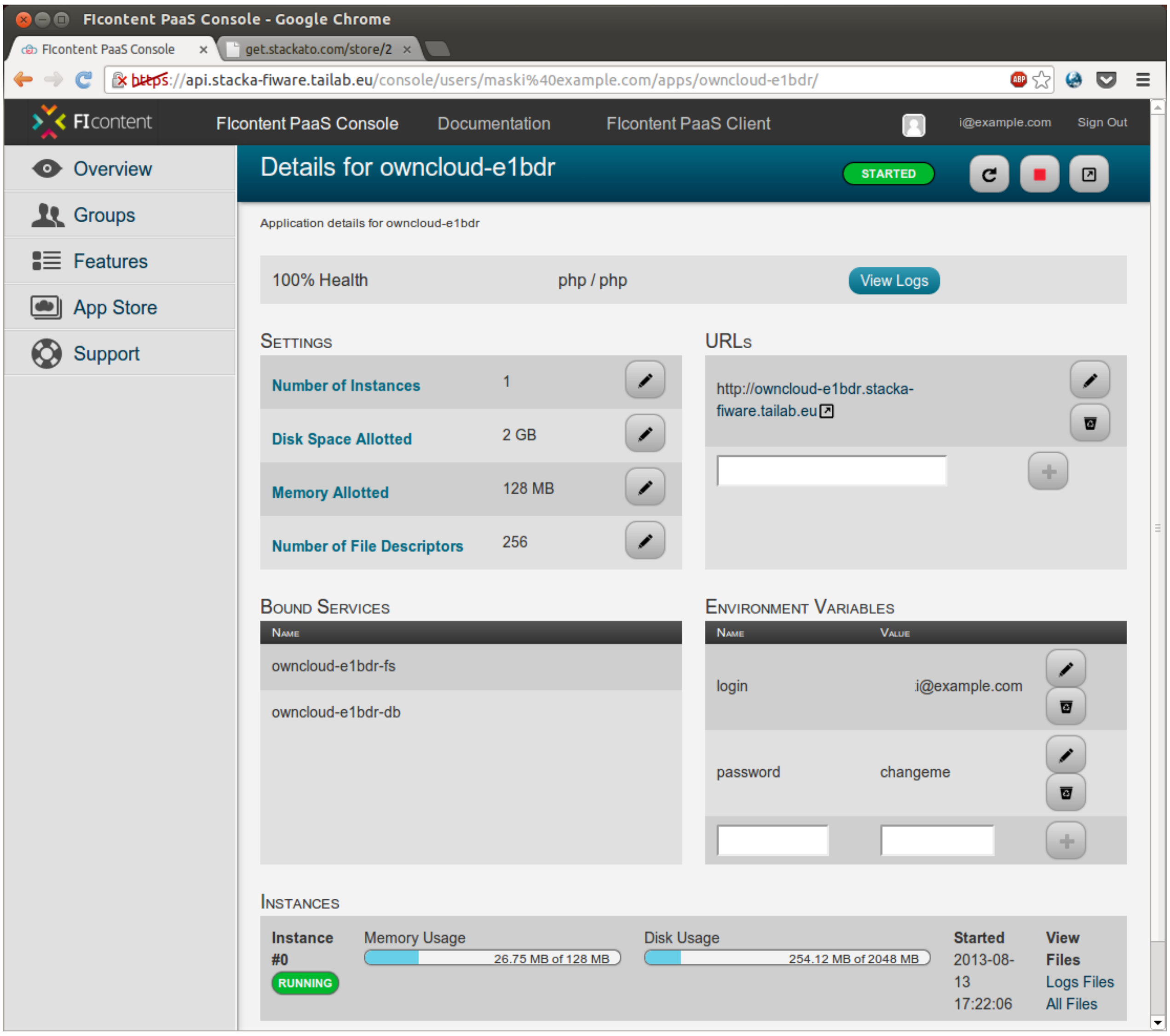
4.1.4 - Application details

The following screenshot shows the details for one particular application. Another important concept to note is that applications are allocated:

- A number of running instances behind a load balancer (which are not VMs but Linux Containers within VMs)
- An amount of RAM memory and disk space in each instance
- A subdomain name

Each application has a set of provisioned services automatically bound to it according to its metadata.

Environment variables can be set and passed to the running instances.



Details for owncloud-e1bdr

Application details for owncloud-e1bdr

100% Health php / php

SETTINGS

Number of Instances	1
Disk Space Allotted	2 GB
Memory Allotted	128 MB
Number of File Descriptors	256

URLS

http://owncloud-e1bdr.stacka-fiware.tailab.eu

BOUND SERVICES

Name
owncloud-e1bdr-fs
owncloud-e1bdr-db

ENVIRONMENT VARIABLES

Name	Value
login	i@example.com
password	changeme

INSTANCES

Instance	Memory Usage	Disk Usage	Started	View Files
#0 RUNNING	26.75 MB of 128 MB	254.12 MB of 2048 MB	2013-08-13 17:22:06	Logs Files All Files

Figure 13 - Screen with the details of a deployed application

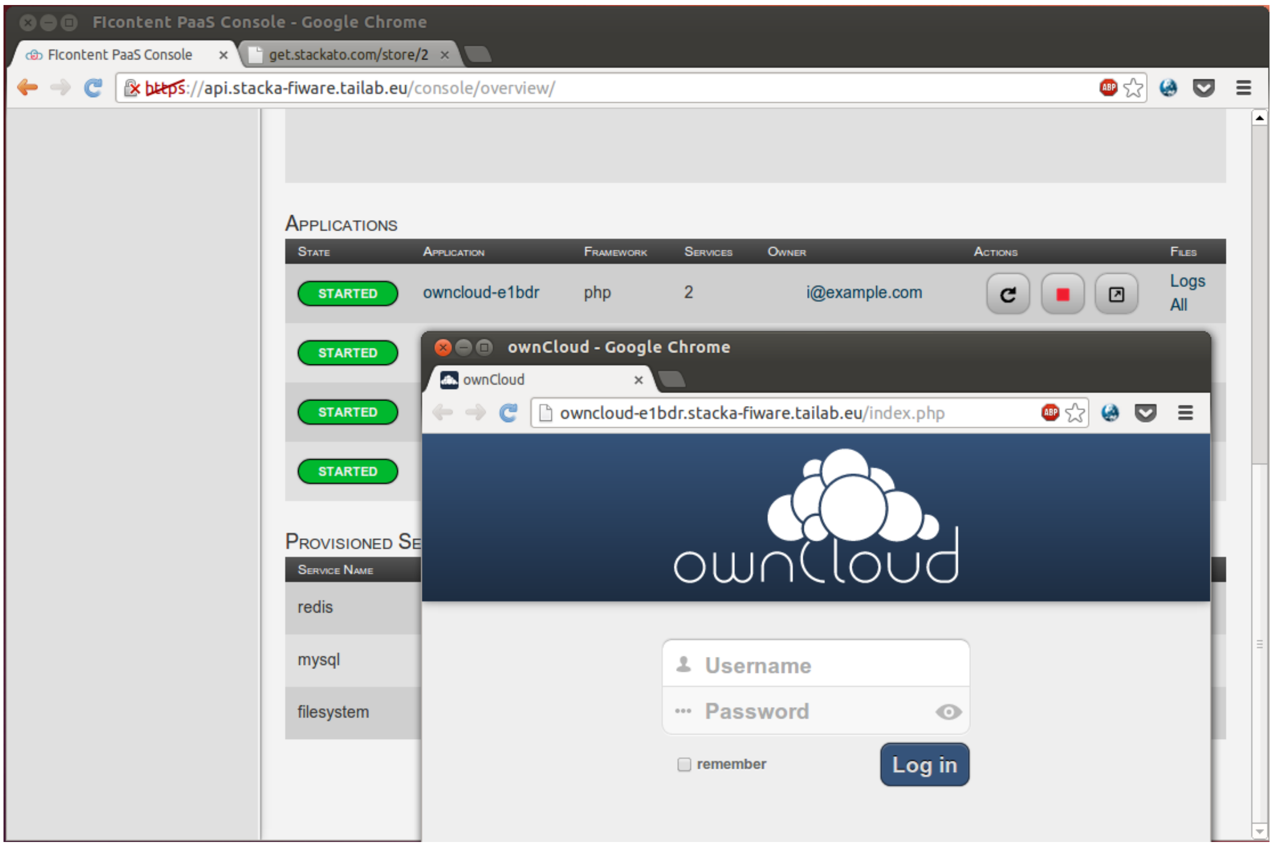
4.1.5 - Pushing a new application

In order to push a new application to CloudFoundry, the vmc tool needs to be installed on the client machine.

The official CloudFoundry documentation can be found on this webpage on GitHub: https://github.com/cloudfoundry/vmc

4.1.6 - Link to hosted application

The following screenshot shows the result of clicking in the subdomain name that was created for the hosted web application (open-source ownCloud file sharing in this example):



State	Application	Framework	Services	Owner	Actions	Files
STARTED	owncloud-e1bdr	php	2	i@example.com		Logs All
STARTED						
STARTED						
STARTED						

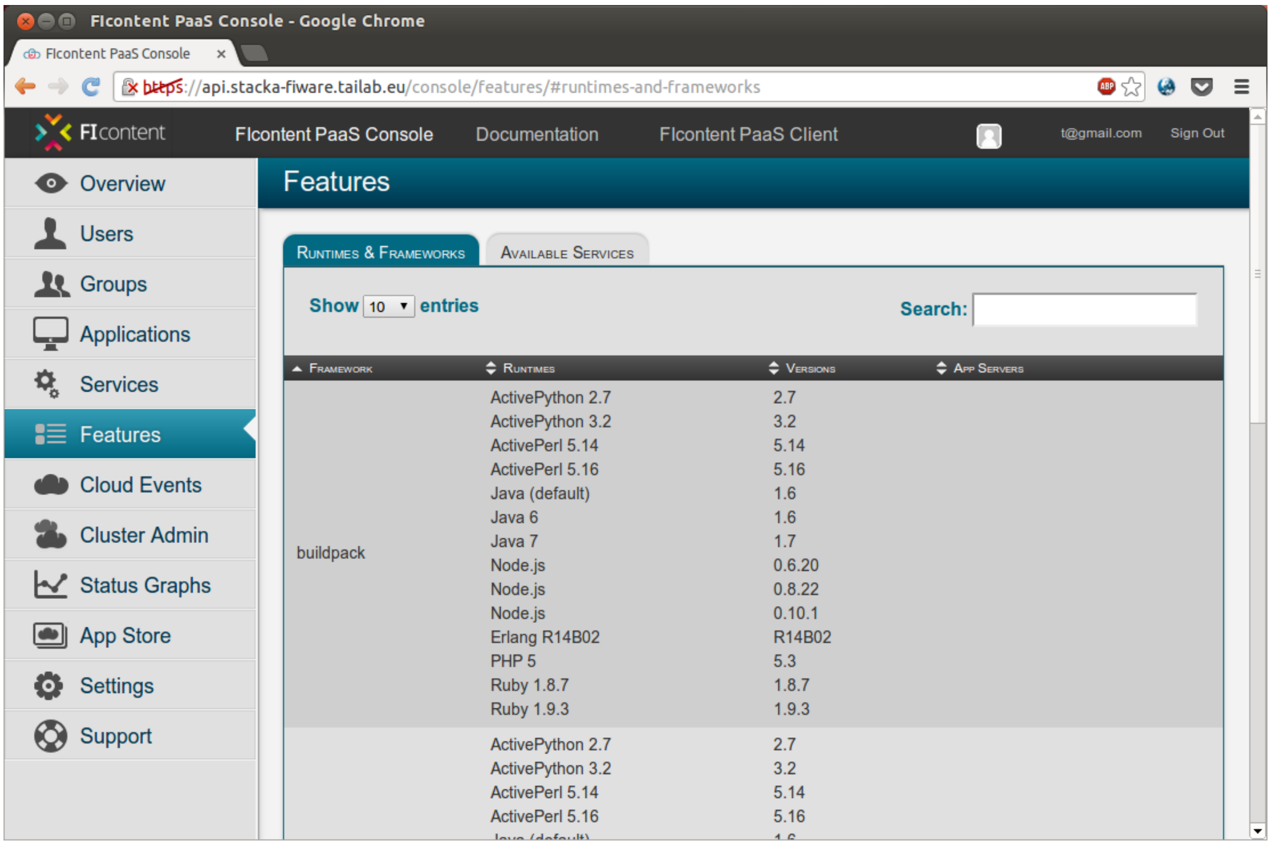
Service Name
redis
mysql
filesystem

Figure 14 - Accessing a running web application from the dashboard

4.2 - Administrator

4.2.1 - Available runtimes

The following screenshot shows the list of runtimes and frameworks available to developers of applications:

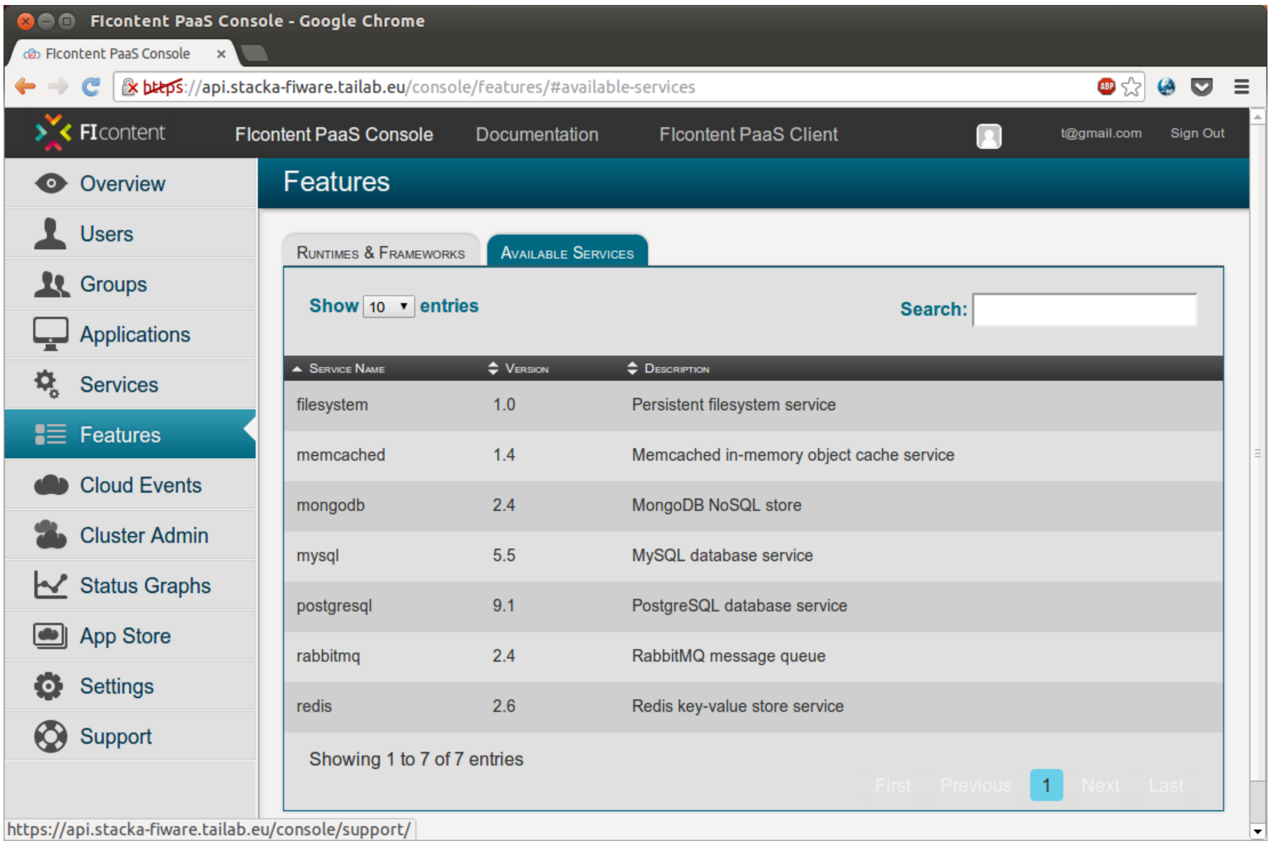


Framework	Runtimes	Versions	App Servers
buildpack	ActivePython 2.7	2.7	
	ActivePython 3.2	3.2	
	ActivePerl 5.14	5.14	
	ActivePerl 5.16	5.16	
	Java (default)	1.6	
	Java 6	1.6	
	Java 7	1.7	
	Node.js	0.6.20	
	Node.js	0.8.22	
	Node.js	0.10.1	
	Erlang R14B02	R14B02	
	PHP 5	5.3	
	Ruby 1.8.7	1.8.7	
	Ruby 1.9.3	1.9.3	
	ActivePython 2.7	2.7	
	ActivePython 3.2	3.2	
	ActivePerl 5.14	5.14	
	ActivePerl 5.16	5.16	

Figure 15 - List of available runtimes and frameworks

4.2.2 - Available services

The following screenshot shows the list of services that can be bound to applications:



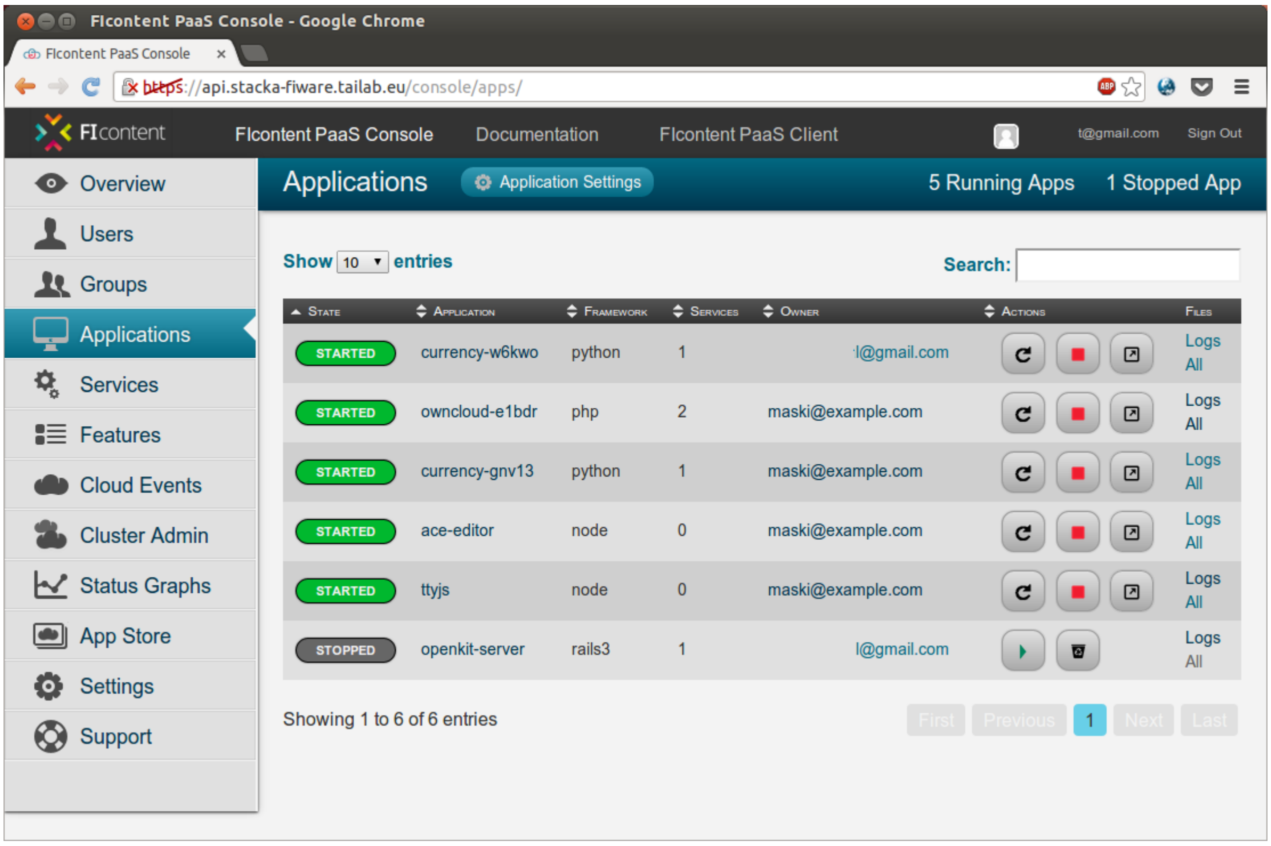
Service Name	Version	Description
filesystem	1.0	Persistent filesystem service
memcached	1.4	Memcached in-memory object cache service
mongodb	2.4	MongoDB NoSQL store
mysql	5.5	MySQL database service
postgresql	9.1	PostgreSQL database service
rabbitmq	2.4	RabbitMQ message queue
redis	2.6	Redis key-value store service

Showing 1 to 7 of 7 entries

Figure 16 - List of available services

4.2.3 - Deployed applications

The following screenshot shows the list of all the applications deployed by all users:



Applications

5 Running Apps 1 Stopped App

Show 10 entries Search:

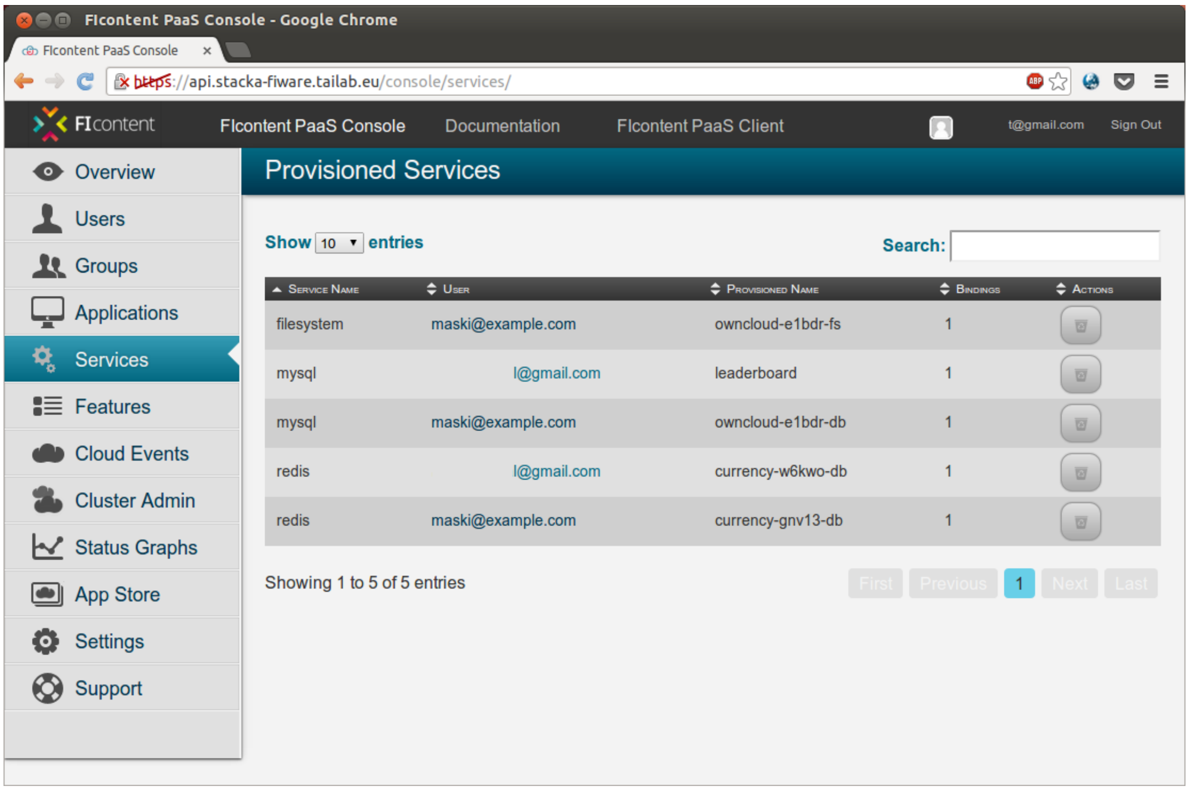
State	Application	Framework	Services	Owner	Actions	Files
STARTED	currency-w6kwo	python	1	l@gmail.com		Logs All
STARTED	owncloud-e1bdr	php	2	maski@example.com		Logs All
STARTED	currency-gnv13	python	1	maski@example.com		Logs All
STARTED	ace-editor	node	0	maski@example.com		Logs All
STARTED	ttyjs	node	0	maski@example.com		Logs All
STOPPED	openkit-server	rails3	1	l@gmail.com		Logs All

Showing 1 to 6 of 6 entries

Figure 17 - List of the applications of all the users

4.2.4 - Provisioned services

The following screenshot shows the list of all services provisioned by all users:


FIcontent PaaS Console - Google Chrome

https://api.stacka-fiware.tailab.eu/console/services/

Provisioned Services

Show 10 entries Search:

Service Name	User	Provisioned Name	Bindings	Actions
filesystem	maski@example.com	owncloud-e1bdr-fs	1	
mysql	l@gmail.com	leaderboard	1	
mysql	maski@example.com	owncloud-e1bdr-db	1	
redis	l@gmail.com	currency-w6kwo-db	1	
redis	maski@example.com	currency-gnv13-db	1	

Showing 1 to 5 of 5 entries

First Previous 1 Next Last

Figure 18 - List of the services provisioned for all the users

5 - Conclusion

This document accompanies the prototype named “Content-dedicated Platform-as-a-Service”: it first presents the notion of Platform-as-a-Service and explains the rationale for the adoption of a Platform-as-a-Service to simplify the development of server-side applications in the FI-CONTENT 2 project. It then shows how the current initial version of the prototype works.