Universidade de Aveiro
2013

Departamento de Eletrónica,
Telecomunicações e Informática

Carlos Eduardo Isidro Gonçalves

Intermediador de Serviços na Núvem

Cloud Service Broker

Universidade de Aveiro
2013

Departamento de Eletrónica,
Telecomunicações e Informática

Carlos Eduardo Isidro Gonçalves

Intermediador de Serviços na Núvem

Cloud Service Broker

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Diogo Gomes, Professor Auxiliar Convidado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor João Paulo Barraca, Professor Assistente Convidado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico esta dissertação aos meus pais, Fátima e Fernando.

o júri / the jury

presidente / president

Prof. Doutor José Luis Oliveira
Professor Associado da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Rui Carlos Mendes Oliveira
Professor Associado do Departamento de Informática da Escola de Engenharia da Universidade do Minho

Prof. Doutor Diogo Nuno Pereira Gomes
Professor Auxiliar Convidado da Universidade de Aveiro (orientador)

agradecimentos /
acknowledgements

Agradeço aos meus orientadores, Professor Doutor Diogo Gomes e Professor Doutor João Paulo Barraca, pela orientação na realização desta dissertação. Agradeço ainda pelo convite para me juntar ao grupo de investigação HNG/ATNoG no Instituto de Telecomunicações no qual estou inserido.

Agradeço à PT Inovação S.A. pela oportunidade dada em realizar este trabalho que se enquadrou num Plano de Inovação em parceria com o Instituto de Telecomunicações. De forma especial, agradeço ao Doutor Engenheiro Pedro Neves pelas várias discussões tidas no decorrer deste trabalho.

Não individualizando, agradeço a todos os meus colegas e amigos do Instituto de Telecomunicações, da Universidade de Aveiro e de longa data. A vossa camaradagem e mui sábias palavras, umas menos ortodoxas que outras para aqui serem eternizadas, fomentaram a vontade de atingir novos objetivos.

Por fim, agradeço à minha família, especialmente aos meus pais, pela motivação e apoio incondicional no decorrer da minha formação pessoal e académica. É a eles a quem dedico este trabalho.

A todos vós, muito obrigado.

Palavras Chave

Computação na Núvem, IaaS, PaaS, SaaS, Intermediação de serviços, SOA, Interoperabilidade na núvem.

Resumo

De acordo com história dos sistemas informáticos, os engenheiros têm vindo a remodelar infraestruturas para melhorar a eficiência das organizações, visando o acesso partilhado a recursos computacionais. O advento da computação em núvem desencadeou um novo paradigma, proporcionando melhorias no alojamento e entrega de serviços através da Internet. Quando comparado com abordagens tradicionais, este apresenta vantajens por disponibilizar acesso ubíquo, escalável e sob demanda, a determinados conjuntos de recursos computacionais partilhados.

Ao longo dos últimos anos, observou-se a entrada de novos operadores que providenciam serviços na núvem, a preços competitivos e diferentes acordos de nível de serviço ("Service Level Agreements"). Com a adoção crescente e sem precedentes da computação em núvem, os fornecedores da área estão se a focar na criação e na disponibilização de novos serviços, com valor acrescentado para os seus clientes. A competitividade do mercado e a existência de inúmeras opções de serviços e de modelos de negócio gerou entropia. Por terem sido criadas diferentes terminologias para conceitos com o mesmo significado e o facto de existir incompatibilidade de Interfaces de Programação Aplicacional ("Application Programming Interface"), deu-se uma restrição de fornecedores de serviços específicos na núvem a utilizadores. A fragmentação na faturação e na cobrança ocorreu quando os serviços na núvem passaram a ser contratualizados com diferentes fornecedores. Posto isto, seria uma mais valia existir uma entidade, que harmonizasse a relação entre os clientes e os múltiplos fornecedores de serviços na núvem, por meio de recomendação e auxílio na intermediação.

Esta dissertação propõe e implementa um Intermediador de Serviços na Núvem focado no auxílio e motivação de programadores para recorrerem às suas aplicações na núvem. Descrevendo as aplicações de modo facilitado, um algoritmo inteligente recomendará várias ofertas de serviços na núvem cumprindo com os requisitos aplicacionais. Desta forma, é prestado aos utilizadores formas de submissão, gestão, monitorização e migração das suas aplicações numa núvem de núvens. A interação decorre a partir de uma única interface de programação que orquestrará todo um processo juntamente com outros gestores de serviços na núvem. Os utilizadores podem ainda interagir com o Intermediador de Serviços na Núvem a partir de um portal Web, uma interface de linha de comandos e bibliotecas cliente.

Keywords

Cloud Computing, IaaS, PaaS, SaaS, service brokering, SOA, Cloud interoperability.

Abstract

Throughout the history of computer systems, experts have been reshaping IT infrastructure for improving the efficiency of organizations by enabling shared access to computational resources. The advent of cloud computing has sparked a new paradigm providing better hosting and service delivery over the Internet. It offers advantages over traditional solutions by providing ubiquitous, scalable and on-demand access to shared pools of computational resources.

Over the course of these last years, we have seen new market players offering cloud services at competitive prices and different Service Level Agreements. With the unprecedented increasing adoption of cloud computing, cloud providers are on the look out for the creation and offering of new and value-added services towards their customers. Market competitiveness, numerous service options and business models led to gradual entropy. Mismatching cloud terminology got introduced and incompatible APIs locked-in users to specific cloud service providers. Billing and charging become fragmented when consuming cloud services from multiple vendors. An entity recommending cloud providers and acting as an intermediary between the cloud consumer and providers would harmonize this interaction.

This dissertation proposes and implements a Cloud Service Broker focusing on assisting and encouraging developers for running their applications on the cloud. Developers can easily describe their applications, where an intelligent algorithm will be able to recommend cloud offerings that better suit application requirements. In this way, users are aided in deploying, managing, monitoring and migrating their applications in a cloud of clouds. A single API is required for orchestrating the whole process in tandem with truly decoupled cloud managers. Users can also interact with the Cloud Service Broker through a Web portal, a command-line interface, and client libraries.

Contents

List of Figures

List of Tables

List of Acronyms

A6	Automated Audit, Assertion, Assessment, and Assurance API
ACM	Application Control Management
ANSI	American National Standards Institute
API	Application Programming Interface
APM	Application Performance Manager
AWS	Amazon Web Services
CAMP	Cloud Application Management for Platforms
CAPEX	Capital Expenditures
CDMI	Cloud Data Management Interface
CDN	Content Delivery Network
CIMI	Cloud Infrastructure Management Interface
CLI	Command Line Interface
CPU	Central Processing Unit
CRM	Customer Relationship Management
CRUD	Create, Read, Update and Delete
CSB	Cloud Service Broker
CSP	Cloud Service Provider
DBaaS	Database as a Service
DMTF	Distributed Management Task Force
DNS	Domain Name System

DSL	Domain-Specific Language
DSTM	Distributed Software Transactional Memory
EC2	Elastic Compute Cloud
ERP	Enterprise Resource Planning
FP7	Seventh Framework Programme
HVAC	Heating, Ventilation and Air Conditioning
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IDCloud	Identity in the Cloud
IETF	Internet Engineering Task Force
INCITS	International Committee for Information Technology Standards
IP	Internet Protocol
iSCSI	Internet Small Computer System Interface
IT	Information Technology
Java EE	Java Platform, Enterprise Edition
JAX-RS	Java API for RESTful Services
JPA	Java Persistence API
JPL	Java-Prolog Library
JVM	Java Virtual Machine
JSON	JavaScript Object Notation
KVM	Kernel-based Virtual Machine
LGPL	GNU Lesser General Public License
MCN	Mobile Cloud Networking
MVC	Model-View-Controller
NaaS	Network as a Service
NAS	Network Attached Storage
NFS	Network File System

NIST	National Institute of Standards and Technology
OASIS	Organization for the Advancement of Structured Information Standards
OCCI	Open Cloud Computing Interface
OMG	Object Management Group
OPEX	Operational Expenditure
ORM	Object-Relational Mapping
OSI	Open Systems Interconnection
OVF	Open Virtualization Framework
PaaS	Platform as a Service
QoS	Quality of Service
RAM	Random Access Memory
ROI	Return On Investment
RoR	Ruby on Rails
RESTful	Representational State Transfer
S3	Simple Storage Service
SaaS	Software as a Service
SAN	Storage Area Network
SDK	Software Development Kit
SMB	Small and Medium Business
SCM	Source Code Management
SSH	Secure Shell
SSO	Single Sign-On
SLA	Service Level Agreement
SOA	Service-Oriented Architecture
SMS	Short Message Service
SNIA	Storage Networking Industry Association
STM	Software Transactional Memory

TCO	Total Cost of Ownership
TLS	Transport Layer Security
TOSCA	Topology and Orchestration Specification for Cloud Applications
URL	Uniform Resource Locator
VCS	Version Control System
VM	Virtual Machine
WAR	Web application ARchive
WWW	World Wide Web
XaaS	Everything as a Service
XML	eXtensible Markup Language
YAML	YAML Ain't Markup Language

CHAPTER 1

Introduction

The increasing use of Information Technologys (ITs) has rapidly intensified people's demand for services publicly available in an anywhere, anytime and anyhow forms. This high demand is motivated by the raise of the latest breakthroughs on services, accessible at user's fingertips, through the use of electronic devices connected to the World Wide Web (WWW), over broadband connections. The debut of easy-to-carry and Internet-capable devices such as laptops, smartphones, tablets and sensors, opened the door for technology companies to research, development and offer more complex services to the market, which many of them never imagined could have been achieved and released at a competitive price.

Considering the amount of devices connected to the Internet, and on top of a consumerist approach where users were only consuming data, new ways of communicating and interacting with people around the world emerged. Users started producing data (e.g., videos, photos, music) and sharing it with others online. The proportion of such volume of data could not be handled easily or efficiently.

Computational systems were assembled to host either a large or limited number of software services. Both approaches impose substantial drawbacks on the quality of service and must be addressed carefully. On the first case, while the quantity of physical resources is reduced, systems were put under a high load of processing into a single node and therefore over provisioned. In the event of a system suffering an unexpected malfunctioning it could put the whole ecosystem at risk and thus compromising the rest of the well-behaved running programs. The latter approach could accommodate a higher number of concurrent requests for a given service as available idled resources were sitting still but were under provisioned. It required a lot more computers and consequently data centers needed additional space to house dozens, possibly hundreds, of machines. Heating, Ventilation and Air Conditioning (HVAC) systems had to be extended, increasing electrical consumption and human resources required to maintain the entire infrastructure. Server consolidation became imperative.

Server consolidation has become a common practice in data centers as it can increase the efficient use of server resources and reduce costs. Applications that traditionally ran on under-

utilized dedicated servers are consolidated to fewer machines sharing resources pools with other applications. Although server consolidation substantially increases resource utilization, it may also introduce complexity in new configurations of data, applications and servers. Virtualization technology has the potential to mitigate this problem. With virtualization, a single physical server can be partitioned into multiple isolated virtual environments (sandboxes) and thus help building secure platforms. It permits multiple execution environments and can increase Quality of Service (QoS) by guaranteeing specified amount of resources per environment. Virtualization can be applied to not only servers but also storage, network, and applications.

Virtualization is paving the way of a new era in IT evolution. It is a key enabling technology for the emerging paradigm of cloud computing. The concept has been around for some years now however it has only recently become captivating from both a consumer and provider perspective. The amount of time and finances invested in managing IT has skyrocketed among government, business, education community, and media. Combining cloud and virtualization technologies, interested parties can deliver optimized resources, on-demand utilization, flexibility and scalability services. Cloud lets IT departments optimize resources capacity based on needs, and paying only for the resources required.

1.1 MOTIVATION AND GOALS

Cloud computing is broken down to three major layers: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Several companies such as Amazon, Rackspace, Google, and Microsoft, expose their cloud solutions publicly in a business-to-client and business-to-business model. With all those companies easily provisioning on-demand cloud services, cloud consumers now face the problem of choosing a provider, with critical decision factors being cost of operation, service availability, and auto-scaling.

This dissertation addresses the aforementioned issue by aiming to develop an intelligent and autonomous Cloud Service Broker (CSB) capable of easing crucial application lifecycle stages (deployment, management, monitor and migrate) from developers orchestrating the whole process in a cloud of clouds. To achieve such a goal, the broker should be aware of user application requirements and transparently deploy to the best rated cloud provider based on given premises, being either and ideally a PaaS or if premises are not fulfilled, falling back to an IaaS solution. Through the proposed CSB, end-user developers will have access to a feature-complete Web portal and a command-line interface where they can auto-deploy, manage and monitor their appliances.

1.2 CONTRIBUTIONS

A paper entitled "Towards a Cloud Service Broker for the Meta-Cloud" [1] was published in the 12th Conferência sobre Redes de Computadores, Aveiro, Portugal.

A Distributed Management Task Force (DMTF) Cloud Infrastructure Management Interface (CIMI) Java model and client library was developed and published under the GNU Lesser General Public License (LGPL) version 3[1].

The author participated in discussions on the DeltaCloud project and contributed on several bug hunting and testing of the cloud software. A code contribution to the CIMI Application Programming Interface (API) frontend was made, using correct CIMI v1.0 namespace in XML[2] in accordance with CIMI v1.0 specification. This contribution enabled automated generation of the CIMI Java models for the developed CIMI Java library.

An OpenStack private cloud was installed and configured on the data center of the Instituto de Telecomunicações ATNoG[3] research group. This setup led to significant contributions to the OpenStack project. Two bugs preventing NFS from being used as a virtualized block storage were found, fixed and submitted to OpenStack:

1. **Add commands used by NFS volume driver to rootwrap:** the NFS volume driver required tools to be executed as root. Those tools needed to be root-wrapped. Bug affected the OpenStack Block Storage (*"Cinder"*) component and was fixed on OpenStack 2013.1 *"grizzly"*[4] and OpenStack 2012.2.3 *"folsom"*[5] series.

2. **No handler for NFS volume:** the NFS volume driver was implemented but not added to the *libvirt* volume driver list. The fix added NFS to that list. Bug affected the OpenStack Compute (*"Nova"*) component and was fixed on OpenStack 2013.1 *"grizzly"* and OpenStack 2012.2.3 *"folsom"*[6] series.

1.3 DOCUMENT OUTLINE

The remainder of this document is organized as follows. Chapter 2 introduces the reader to cloud computing. It compares traditional IT with the cloud and why enterprises should adopt it. The cloud computing stack is described, and interoperability issues and initiatives are discussed. A summary of ongoing cloud research projects is provided. Chapter 3 focuses on the important role of cloud service brokerage platforms and multi-cloud. Commercial and research solutions towards cloud service brokerage services are enumerated. A solution for a

[1] DMTF's CIMI library for Java: `https://github.com/cgoncalves/cimi-java`
[2] https://github.com/apache/deltacloud/commit/be6f8e8f4d5e010367ecc37fefdf84f7c34b3a64
[3] ATNoG: http://atnog.av.it.pt
[4] https://bugs.launchpad.net/cinder/+bug/1087282
[5] https://bugs.launchpad.net/cinder/folsom/+bug/1087282
[6] https://bugs.launchpad.net/nova/+bug/1087252

cloud service broker is proposed in Chapter 4, presenting its specification and architecture. Its implementation details and results are described in Chapter 5. Chapter 6 concludes the work done.

CHAPTER 2

Cloud Computing

Cloud computing is an emerging technology used to deliver on-demand services over the Internet. It is undoubtedly affecting the way business is conducted and is empowering a new generation of products and services. Cloud Computing can be summarized into three keywords: elasticity, on-demand, and (autonomously) fully managed [2]. These three characteristics massively benefit organizations by reducing both Capital Expenditures (CAPEX) and Operational Expenditure (OPEX) while enabling them to channel their efforts to the strategic business sector [3].

The meaning of cloud computing varies from activity to activity, depending whether one is working on upper stack layers or on lower ones. Different authors have tried through the years to find a consensus on a single and wide definition for cloud computing. It is also observed that many scientific, journalistic and academic articles use their description based on their own view of cloud. The U.S. Government's National Institute of Standards and Technology (NIST) defines cloud computing as follows [4]:

> *Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.*

This is by far the most cited and agreed definition shared by most people. The author of this document also shares the same view as NIST and therefore forthcoming interpretations on cloud computing are extracted from such bibliographic reference.

2.1 CHARACTERISTICS AND TERMS

In order to understand what makes a system a cloud solution it is required to clarify its true essence by characterizing key attributes of cloud computing. At a high level, cloud computing are generic servers that one can pay for and use as needed. There are five essential attributes that must exist on an IT infrastructure, and they are: rapid elasticity, measured services, on-demand self-service, resource pooling and ubiquitous network access. These characteristics are generally accepted by independent bodies like NIST [4].

Elasticity is defined as the ability to add and remove computing capacity on demand. To the consumer the cloud appears to be an unlimited source of computational resources and he can purchase as much as needed when he needs. Applications with highly variable workload or unpredictable growth can take advantage of this characteristic. Elasticity in traditional IT systems are handled manually. When the load of services increase one needs to power up more machines and add them to the pool of servers that run the service. When the load decreases, one needs to start removing unnecessary servers from the pool and then powering them off. It's an expensive and time-consuming task to constantly add and remove machines from the pool. Instead, the maximum required number of machines is plugged in all the time to the pool even under low utilization. There might also be the case that resource planning is not done right and the load of a service is so high that even the maximum available computing resources on the servers are not enough to handle the unexpected demand. Errors and dropped requests will be the result of miscalculated necessary resources, with ultimately the provider to receive complains from its customers. In a cloud scenario, capabilities can be elastically provisioned and released within seconds or minutes, in some cases even automatically, and one does not need to speed a great deal of time doing capacity planning.

Measured service is a characteristic controlled, optimized and metered/reported by the cloud provider. Billing, access control, resource optimization and capacity planning are tasks assigned not to the cloud consumer but to the cloud provider.

As a consumer of cloud resources, on-demand self-service means consumers can self-provision and control computing capabilities at any time without any human interaction from the cloud provider. On-demand self-service empowers users to request resources at run time rather than what it was required in the old days. In the past, consumers would have to call hardware vendors, order new servers, wait several days to receive them, and install the operating system. Servers need to be connected to the network, configured with firewall rules, and install and configure software services. This process can take weeks or months to get the hardware ready to use. With cloud computing, this lengthy process is reduced dramatically to a few seconds or minutes by just a matter of logging in to a Cloud Service Provider (CSP) portal and clicking on a button or calling the provider's API to have additional resources available within a short time interval.

Resource pooling is the ability of a cloud to offer a pool of computing resources that can be dynamically assigned to multiple resource consumers via a multi-tenant model. Such dynamic

resource reassignment capability offers much flexibility to providers for managing their own physical resource usage and operating costs. Sharing a large pool of resources among several users reduces infrastructure costs. Common examples of resource pools include data storage services, processing services and networking services.

The last characteristic for a service to be considered a cloud computing solution is ubiquitous network access. Access to the cloud provider's capabilities has to be made available over the network and accessible through standard mechanisms, either workstations, laptops, tablets or mobile phones. Deploying cloud services across geographical regions can even enhance ubiquitous network access. Users located at distant locations would experience lower response times and possibly fewer errors caused by networking glitches.

In order to comprehend terms related to cloud computing used extensively throughout this document it is necessary to describe a few terms.

Service Level Agreement (SLA) is a negotiated agreement between a customer and a service provider. The service contract sets a common understanding about relevant roles and responsibilities for both parties. Relevant responsibilities are security, data encryption, privacy, data retention and deletion, transparency, metrics, availability, auditability, etc. The service level is defined in quantitative terms that the service has to be offered to a customer. To measure it, the service provider must be able to define metrics and be able to measure them on a regular basis.

Multi-tenancy is a principle in software architecture where an instance of software running on a server can be virtually partitioned to serve multiple client organizations called tenants, each working on a virtualized application instance. Tenants may be given the ability to customize parts of an application but cannot customize core software's characteristics. A multi-tenant environment can be economical because software development, resources and maintenance costs are shared across customers. As the same instance is shared, the service provider has only to update the software once.

Service-Oriented Architecture (SOA) is a software and architectural paradigm that exposes software services in a standard form. Services work independently of how or where they are deployed and can be geographically distributed across different machines. Application's business logic is independent of the implementation and therefore due to this nature services are loosely coupled. An application programmed in a language can consume data from a service implemented in a different programming language.

Cloud bursting adds capabilities to enable an organization to scale out their private cloud infrastructure to use resources from an external CSP for certain time intervals; the renting of external resources improves the elasticity of the company's IT infrastructure, and allows them to confront the fluctuations on demand dynamically [5].

Interoperability is the ability of diverse systems and organizations to work together [6]. In the computer world, this property means the exchange of information and its use between multiple systems. This term is recurrently expressed as "no vendor lock-in".

2.2 TRADITIONAL IT AND ECONOMICAL IMPACT

Organizations make tremendous investments towards an IT infrastructure. For small to medium-sized enterprises the total cost of owning and managing hardware and software in-house is enormously expensive. IT departments are incessantly trying to reduce Total Cost of Ownership (TCO) in order to maintain business at a profitable level. A recurring way of reducing TCO has been hiring consultants to help out lower costs of operation, but that also involves costs that may not justify the work [7][8]. Organizations are now turning to cloud computing solutions because of its promising business as being cost effective and technical characteristics. It is then pertinent to scrutinize how cloud compares to traditional computing environments.

The traditional IT infrastructure model is very resource-intensive just to carry daily IT operations. For instance, an IT organization has to cycle hardware and software licensing with agreements ranging from months to years, in-house and contracted technicians are needed to install and maintain the computational environment and spend large amount of money to upgrade hardware. Enterprises of assorted sizes and complexities benefit from a cloud computing solution. Small corporations and start-ups will notice cost benefits immediately.

There are noteworthy key differences between traditional IT and cloud computing that leads to choosing the latter approach [9]. Provisioning time takes minutes/hours on the cloud and days/weeks on traditional. There are upfront costs on a traditional while a pay-as-you-go on cloud; the play-as-you-go model reduces or eliminates upfront costs in procuring hardware and software resources. When analyzing economies of scale, the cloud provides cost advantages in procurement of hardware and software, and provides advantages from improved productivity. Multi-tenancy is typically found in cloud to host multiple consumers effectively across shared resources. On the scalability factor, cloud resources can be scaled up or down automatically whereas on traditional IT environments human intervention may be required to add hardware and/or software. Virtualization is generally used on cloud solutions, whereas traditional environments use physical environments only or a mix of physical and virtualized.

There are, however, two great advantages of in-house systems: security and control over data. Having data stored on third party servers is subject to cloud providers to keep data shield not only from outside intruders, but from tenants having access to shared resources. Having applications and data in a in-house infrastructure gives enterprises greater control. However, cloud providers are addressing both concerns by letting customers audit for security vulnerabilities and mitigating control of data by adopting open cloud standards.

When it comes to the economical impact that cloud computing represents, many studies show values ranging considerably but all concur that cloud has indeed changed worlds industry for the better. Forecasts are very positive on how cloud will continue gathering and driving more businesses.

The 3rd Annual Future of Cloud Computing Survey [10], a joint work by North Bridge Venture Partners and GigaOM Research, is the industry's broadest survey on providing a

glance into cloud computing adoption trends, inhibitors and drivers of growth. In the third survey 855 respondents participated, including vendors (57%), business user and IT decision makers (43%). One-third of respondents were C-Suite[1]: 36% C-Level, 31% IT, 21% business and middle management and 21% others. Fifty-seven collaborators, including corporations such as Citrix, Microsoft, RedHat and Rackspace, supported the survey. North Bridge Venture Partners reports that investments by venture capitalists has totaled $1.6 thousand million in 2010 and increased to $2.4 thousand millions in 2011. Even though in 2012 investments slowed down to $1.8 thousand million, cloud adoption continued to rise in 2013 with 75% of those surveyed making use of some sort of cloud platform, representing a 67% growth from the year before. According to forecasts reported, a 126.5% increase from 2011 on the total worldwide addressable market for cloud computing is expected reaching $158.8 thousand millions.

2.3 MOVING TOWARDS CLOUD COMPUTING

A realistic understanding of the concerns regarding moving towards cloud is essential. Cloud computing provides increased efficiency, numerous options for improving business processes, applications and services. While cloud computing becomes the leading delivery mechanism for computing, organizations should weight up first moving their applications towards the cloud and prioritize their move. Applications that would benefit and should be the first ones being taken out from on-premise infrastructures to the cloud are [11]:

Pilot projects: cloud pilot projects are a good way for enterprises to assess how reliable, cost beneficial, and useful deploying to the cloud can be. Pilot testing should be done through non-critical applications, have limited scope and Return On Investment (ROI).

Variable workloads: many applications don't require high computational resources all the time, but only at short and sometimes predictable periods. Prior to cloud computing, organizations would have to invest on machinery to handle maximum workload, even though most of the time it would be idle. Moving that workload to the cloud eases organizations to concentrate on buying only the resources to handle its normal requirements. Under workload peak situations, cloud computing can provision the resources needed by scaling up resources and releasing those when the workload returns to normal.

Non-essential tasks: low risk applications and data that may be processed on virtual machines should be moved to automatically free up resources for the rest of the organization. Core applications and critical data can keep on running on organization's own IT setup.

Data mining: data mining is the process by which accurate and previously unknown information is extracted from large volumes of data [12]. It typically requires great

[1]C-Suite refers to a group of corporation's most important senior executives

computational effort to process massive amounts of data. Equipment must be bought, maintained, powered and cooled. By leveraging cloud computing characteristics, this processing task can be moved off in-house premises. Once processed, virtual instances can be terminated.

Development and test: developers need to have the same level of development tools on their machine as on production environments when performing development and testing on in-house systems. Moving development tools into the cloud allows organizations to have a centralized place where all developers have a common testing environment. When software updates are needed, it is only required to update in one place. Hence a single server of virtual machines for required making it possible to shut them down when tests are complete.

Three key areas need to be looked upon before organizations shift to the cloud: technical, and external and internal business considerations [13][14]. Security and legal compliance, and privacy are the main concerns of enterprises assessing the feasibility of adopting the cloud [15][14]. According to a survey [15], 76% of the respondents consider security issues to be their top concern, followed by a 51% on legal issues, 50% on privacy and compliance issues. The less concerning areas of risks are the immaturity of technology (10 %), lack of functionalities (15%) and lack of performance (20 %). Remarkably 63% of participants on the security issue agree that security concerns are what is blocking them from moving to the cloud. The survey report speculates that enterprises are not worried predominantly about the lack of security measures in themselves but about the lack of transparency on the side of CSPs. On another report [16] it is also identified key impediments on private cloud, public cloud and compliance aspects. IT professionals are most concerned with lack of control in the private cloud to enable them to limit access to data and services to authorized users only. Another concern still on private clouds is the lack of visibility into abstracted resources because such resources are made virtual and shared across multiple lines of business or customers. Other top concerns are the adequate firewalling, traffic analysis and Virtual Machines (VMs) disrupting other VMs, multi-tenancy and the potential of mixing VMs, and different trust levels of VMs. In public clouds, concerns are regarded to measurement of security capabilities and with control. IT departments are unable to measure the security services being offered by their CSP, which erodes their confidence in their provider's overall security capabilities. Lack of transparency and ability to perform audits, and physical boundary controls represent other greatest security concerns on public clouds.

2.4 CLOUD COMPUTING STACK

Cloud Computing is a broad term that describes an extensive range of services. To truly understand how the cloud can be of value, it is first important to understand its different components. This section outlines the delivery models in Section 2.4.1 and deployment models in Section 2.4.2.

2.4.1 DELIVERY MODELS

Defining what comprises cloud computing is complex because it comprises of many things. Classification scheme for cloud computing generally agreed consists on three delivery models: infrastructure as a service, platform as a service and software as a service.

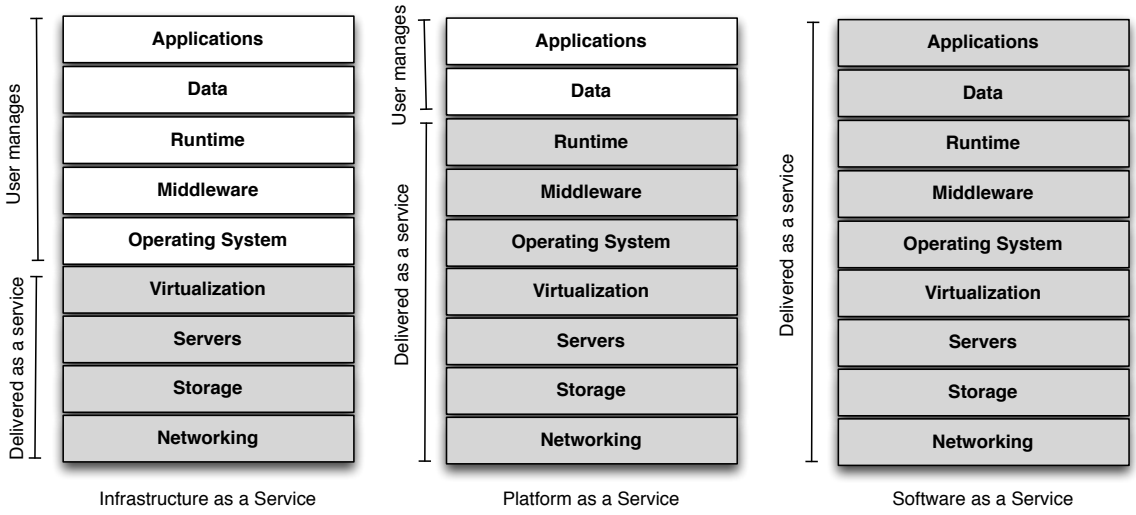

Figure 2.1: Cloud Computing delivery models

IaaS is at the foundation level of the other two cloud computing delivery models, PaaS and SaaS, that provides access to computing resources in a virtualized environment. Capabilities provided to the consumer are processing, storage and network. Consumers do not manage or control the underlying physical infrastructure, but everything from the operating system layer above.

PaaS consists on paying for hardware, network, storage, operating system and a platform capable of running customer's applications as a single service. The customer pays to host and run applications over the Internet. PaaS has several advantages for developers. Operating system can be changed and upgraded frequently without the customer noticing. Geographically distributed development teams can jointly work on the same software projects. Services can be obtained from diverse sources across the globe. Decreasing the number of hardware on-premises, software licenses, and employees, reduces CAPEX and OPEX.

SaaS is a software delivery model that provides access to software remotely. SaaS is becoming an increasingly ubiquitous delivery model as underlying technologies that make use of Web services and SOA. The use of a SaaS model is convenient because it eases administration, software updates and patch management, and collaboration and compatibility since all users will have the same version of software.

End users will typically use SaaS services, software developers PaaS and IT groups whose responsibility lies under the infrastructure will use IaaS. Other than hereby insight models, Network as a Service (NaaS), Database as a Service (DBaaS), and Everything as a Service (XaaS) are many other delivered models but considered a subset of one of the three main delivery models.

INFRASTRUCTURE AS A SERVICE

IaaS is a provision model in which a service provider outsources compute resources, storage and networking capabilities used to support operations. The provider owns the equipment and is responsible for housing, running and maintaining them. The service is offered to customers on-demand and typically paid on a per-use basis. As depicted in Figure 2.1, the provider runs and maintains services such as networking, storage, servers and services in the Open Systems Interconnection (OSI) model [17] from Layer 4 to Layer 7 while the user manages higher levels of the stack including the operating system, data and applications. Servers can be ordinary computers, blade servers, rack servers or others. Servers can be just bare-metal servers or virtualized servers where there is a virtualization layer of software between the hardware and the operating system. Storage can be Storage Area Networks (SANs) or Network Attached Storages (NASs). That includes both locally attached storage within a server or in most cases SAN or NAS environments. Layer 4 to Layer 7 services are comprised of load balancers, firewalls, intrusion detection and all things that are above the network layers of Layer 3. Those servers are going to help protecting services, speed-up performance and making sure there are not being attacked.

IaaS provides a layer of virtualized hardware that can serve as a foundation for SaaS and PaaS delivered by the computing power and data centers required for applications to run. Like other offerings transversal to the cloud stack, elasticity and flexibility characteristics are present in this cloud model. Cloud adopters of IaaS are allowed to choose when, how and what computing resources to consume and scale. Changes are processed on-demand and can drastically reduce time to market. Outsourcing the task of building, running and maintaining infrastructure to a service provider releases enterprises from spending, or at least reducing, capital expenditures on hardware and software. Enterprises are given the possibility with a pay-as-you-go pricing model, paying for just what they use.

The most frequent used type of servers is virtualized servers. A layer of server virtualization can be provided by hypervisors such as VMWare ESX, KVM or Citrix Xen, open source technologies or proprietary ones. These server virtualization technologies allow having multiple

virtual server instances running on top of the same physical server. The ultimate goal of IaaS is to build and deliver this infrastructure under the line of a VM. The application may just need one VM or a cluster of VMs due to the need of large computing resources. On this infrastructure consumers can specify how many Central Processing Units (CPUs) and memory needs to be allocated to the VMs, request to add bandwidth, subnets or a number of Internet Protocol (IP) addresses that are required to make the application work properly. Load balancing servers and firewall servers are other services that can be requested to the IaaS cloud provider.

PLATFORM AS A SERVICE

PaaS is a category of cloud computing that abstracts and integrates in a cloud-based computing environment a platform to allow developers to build, run and manage applications and services over the Internet. PaaS allows users to create software application components that may exist in a cloud environment or may integrate with applications managed in private clouds. PaaS services consist of pre-configured features that users can subscribe to. In a PaaS, developers don't have to be concerned with low-level details. The infrastructure and applications are managed for them, accelerating and simplifying development, delivery and management of complex applications while lowering costs and risks associated with development and operations. Services integrated in a PaaS environment include operating systems, middlewares, database management systems, storage, network access, hosting and tools for design and develop services.

In these cloud-based environments, developers are enabled to leverage key middleware services as they no longer have to provision servers, install web servers, deploy database software and establish secure networks. As a result, developers can push modern web applications to highly available, centralized servers and within seconds applications are deployed. They don't have to invest in physical infrastructure, but are able to pay for virtual infrastructure. Load balancing, rapid scalability, automatic updates, security in an isolated environment, including data security, backups and recovery, and more are all features provided by the cloud platform to customers.

With respect to automatic updates, customers have to make a decision. As new updates and configurations become available, PaaS CSPs can immediately push them to the platform and possibly replacing older versions. Developers have to ponder whether to use a PaaS environment and depend upon on the platform's software update policies or maintain their own stack of software. By maintaining the software stack, one as to set up, configure, maintain and administer the whole software ecosystem. Alternatively, the vendor can be the one to provide these maintainability services. Customers may be forcibly required to update their applications as new software dependencies versions may break compatibility. Another point to be taken under consideration is the risk of vendor lock-in if offerings require proprietary service interfaces or development languages.

SOFTWARE AS A SERVICE

A SaaS is any cloud service where consumers are able to access software applications on a cloud infrastructure over the Internet. The applications are accessible from various client devices such as laptops, desktops or smartphones through user interfaces. Examples of SaaS applications are web-based email, office suite including a word processor, a spreadsheet and a presentation program, Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), and human resource management platforms, media and storage services, and social applications. With traditional software applications, users would have to buy the software upfront and install it onto their computers. In a SaaS model, however, users subscribe to the software instead of purchasing. Subscription is usually on a monthly basis and applications are used online with files saved in the cloud in contrast to saving them locally on users computers. The outcome of online storage is the ubiquity of data on any Internet enabled device connected to the network.

In a SaaS cloud platform, applications are hosted on a remote data center. The provider takes care of all software development, maintenance and updates. Ensuring customers that their data and configurations are isolated and secured from other customers are of the uppermost priorities. Establishing and implementing security measures, both virtually and physically, are other examples of responsibilities of the cloud vendor.

2.4.2 DEPLOYMENT MODELS

Clouds are not all equal. They can be delivered through different delivery models and deployment models. The former was presented in Section 2.4.1 while the latter will be detailed in this section. There are four types of cloud deployment: public, private, community and hybrid.

A public cloud is a cloud computing model in which services, such as applications and storage, are available for general use over the Internet by a service provider who hosts the cloud infrastructure. A private cloud is technically similar to a public cloud infrastructure but owned by a particular organization with IT infrastructure resource pools, and privately owned and usually managed. A community cloud is a combination of a public cloud and a private cloud in the sense that resources are not dedicated to a single organization and yet it's not available to anyone. Hybrid cloud is a combination of a public cloud, with dedicated in-house servers, servers hosted at a service provider, or public or community cloud-based servers.

PUBLIC

In a public cloud, services are run off-site over the Internet and are potentially accessible to the general public. This deployment model offers the greatest level of efficiency in resources as it shares a large pool of physical resources. The infrastructure is already setup and ready

to be deployed. Customers choose what they want, when they want. They can choose how much they want and scale up or down as needed. Metering and reporting is available so that customers know what they are getting at anytime. Public cloud customers benefit from economies of scale because infrastructure costs are spread across all users, allowing each client to operate on a low-cost, "pay-as-you-go" model.

The public model offers several features and benefits to business organizations. The use of public cloud services shifts responsibilities of managing complex IT to third-party providers. Eliminates CAPEX and reduces OPEX because the maintenance and human resource costs associated with creating and managing the infrastructure is offloaded to a provider. With pay-per-use models, organizations are charged by the hour reducing a company's resource needs to the situation of a temporary project or unanticipated spikes in usage. Self-service provisioning leads to lower costs and better agility because human intervention is minimized. These factors avoid the need to build out internal infrastructure and ending in ensuring cost efficiency. Public clouds provide scalability and ability to elastically resize cloud resources based on the organization needs. Access to resources is available through APIs, helping applications scale automatically.

Organization's IT departments are concerned about public cloud security [18][19]. Cloud providers have to be as transparent as possible on security measures, and cloud consumers are required to trust to their provider the data they have on the cloud. Due to the multi-tenant nature of cloud computing, unauthorized access of data by other tenants is a security risk added by public clouds. Equally on a multi-tenant environment, there are resource contention issues. A tenant may intentionally or accidentally consume unreasonable amount of shared resources, affecting the overall performance of other tenants using the same hardware. If data in the cloud is unencrypted then data privacy is an important security consideration that shouldn't be tread lightly. Either an employee on the cloud provider side or an intruder may gain access to the data. Countries have different regulatory requirements on data privacy and cloud providers may not offer information regarding the physical location of the data [20]. In some jurisdictions, cloud providers may be held liable for illegal data they may be hosting.

PRIVATE

A private cloud is a new trend of cloud computing that involves a secure cloud based environment in which only specific users can operate. As with a public environment, private clouds also provide the same technological characteristics. However organizations are reluctant to move their infrastructure to the public domain [19]. One of the reasons is the afraid of losing control over the infrastructure with security and privacy concerns being the greatest fear of running business outside their premises. Private clouds offer a higher level of security and control.

Private clouds are cloud infrastructures powered by a virtualized environment using an underlying pool of physical computing resources owned by and under the control of a private

organization, and used internally to offer services on-demand. A private cloud can be deployed using either automated and managed self provisioning capabilities on top of a virtualized infrastructure or using one of the available free and open source or proprietary software systems that control large pools of compute, storage and networking resources.

The additional security offered by a private cloud comes at the expense of CAPEX costs to build the infrastructure. OPEX costs should also be higher than a public cloud because the infrastructure must be managed and maintained either by the enterprise itself through internal expertise or outsourcing to third party cloud enterprises. Scalability can be a minus point when compared to public clouds as resources are usually limited to the physical perimeter. Nonetheless, organizations get assistance on expanding their cloud industrial park by employing cloud bursting to public clouds. This enables the private sector to offload non-sensitive functions or respond to spikes in demand to a public cloud.

Despite the fact that these clouds overall seem to not to welfare from cloud economics as much as public clouds, there are advantages for businesses in the financial sector, banking, health-care and others. Sensitive data are stored within business premises with tighter security measures.

COMMUNITY

Community cloud computing is a good cloud strategy when several organizations share similar needs like security, location, and compliance considerations. Like in a public cloud, a community cloud shares software, storage and computational resources with multiple organizations. In parallel, in a private cloud, organizations are responsible for the operation of their own IT infrastructure. This model can greatly help organizations saving costs through cooperatively work together with other organizations sharing common requirements.

The conceptualization of community clouds is that organizations join efforts towards a collaborative cloud infrastructure. The ownership and control of the cloud belongs to community as a whole and not to a single organization.

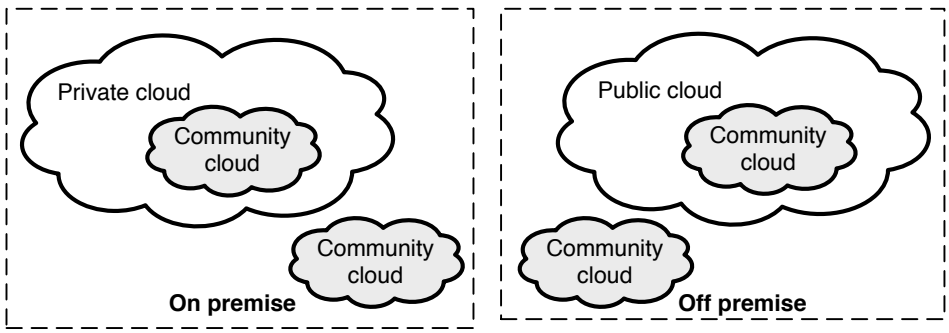

Figure 2.2: Community cloud computing

A community cloud can be hosted either on premises on a private cloud or off promises in a public cloud. Depending on the security required and trusting level on the cloud provider, it can also be hosted on dedicated hosting completely isolated from a cloud computing environment.

An example of a case scenario where one can take the strengths of community clouds are vertical organizations comprising sub-holding companies or governments with their government agencies and institutions. These entities have and share common services and data for specific business they are conducting.

HYBRID

Hybrid cloud combines several clouds and traditional IT models to create a customized cloud. A hybrid cloud can be any combination of using a public cloud, private cloud, community cloud or dedicated in-house servers or servers hosted at a service provider. This model presents opportunities for enterprises to spread workloads across each of these different environments. For instance, sensitive data can be hosted on premises on a private cloud and tasks that are less of a concern, off premises on a public cloud. Requirements for managing a hybrid cloud becomes much more complex. IT departments have now not only to manage one infrastructure but all together while federating capabilities into one bigger cloud environment.

In a hybrid cloud configuration, organization businesses are enabled to continue leverage legacy applications while developing new product solutions for the cloud. Migrate legacy applications to the cloud may be appealing, but it's easier said than done. What a hybrid cloud environment can offer is the possibility to slowly migrate parts of architecture components from a product to the cloud while maintaining non-ready cloud components on non-cloud servers. Other scenario where hybrid clouds can benefit organizations is by offloading compute workloads into the cloud while still maintaining high performance connectivity on in-house systems. Furthermore, companies that have invested large amounts of capital may deploy a hybrid cloud solution to continue use their own IT infrastructure while procuring a third party cloud service provider for augmenting their computing capacity.

2.5 INTEROPERABILITY ON THE CLOUD

There is currently a lot of discussion regarding the role of standards in the cloud. While some parties advocate the cloud is a whole new paradigm with its own new technology and therefore requires new set of standards, other parties say cloud is a technology based on existing technologies that already have standards [21]. The author of this document agrees with a mix of these two sides: cloud is based on existing standardized technology but yet needs its own standards so that it can interoperate with concurrent clouds.

In the cloud community *interoperability* is a term used to refer the ability to easily move

resources from one provider to another and between same or different deployment models.

An obstacle in the adoption of cloud computing is the heterogeneity of solutions. The lack of heterogeneity limits the possibility of customers to move their business to the cloud and moving their data and applications from one cloud provider to another. Most existing cloud solutions have not been built with interoperability in mind.

Cloud computing interoperability difficulties arise when different cloud providers try exchanging raw data, VM or applications. Different providers use different data models, interfaces, authentication and authorization mechanisms, and so on. These incompatibilities prevent providers from communicating in such a way they cannot understand each other and cooperate among them. Consequently they usually lock customers into a single cloud infrastructure, platform or service preventing the portability of data and software to elsewhere. Big vendors like Amazon, Google, SalesForce, VMWare and Rackspace battle for dominance by promoting their own standards instead of agreeing on a widely accepted standard. This is a decisive point for customers to efficiently use the cloud as a leased infrastructure, which is not under their control. If cloud has relieved customers from initial capital expenditure in hardware, installation and managing infrastructure, most of the current solutions bind customers to a cloud technology.

2.5.1 STANDARDIZATION INITIATIVES

Open standards are the main upholders of interoperability. Many non-profit organizations, academies, governments and vendors are committed on advancing cloud computing interoperability standards in various fields such as workloads, authentication and data access.

CloudAudit[2], also known as Automated Audit, Assertion, Assessment, and Assurance API (A6), focus on open, extensible, and secure interface, namespace, and methodology for cloud computing providers and their authorized consumers to automate the audit, assertion, assessment, and assurance of their environments. As of October 2010, CloudAudit is part of the Cloud Security Alliance.

The Cloud Computing Interoperability Forum dedicates to common, agreed-on framework/ontology for cloud platforms to exchange information in a unified manner. Sponsors of the Unified Cloud Interface Project to create an open and standardized cloud interface for the unification of various cloud APIs.

Cloud Security Alliance[3] recommends practices for cloud computing security. Working on Version 3 of the Security Guidance for Critical Areas of Focus in Cloud Computing. Nonprofit organization includes Google, Microsoft, Rackspace, Terremark, and others.

Cloud Standards Customer Council[4] works on standards, security, and interoperability issues related to migration to the cloud. It is an end-user advocacy group sponsored by the

[2]CloudAudit: `http://www.cloudaudit.org`
[3]Cloud Security Alliance: `https://cloudsecurityalliance.org`
[4]Cloud Standards Customer Council: `http://www.cloud-council.org`

Object Management Group (OMG) and creator of the Open Cloud Manifesto.

Cloud Storage Initiative[5] centers its attention on the adoption of cloud storage as a new delivery model (Data-Storage-as-a-Service). Initiative sponsored by the Storage Networking Industry Association (SNIA), the creator and promoter of the Cloud Data Management Interface (CDMI). SNIA includes members from NetApp, Oracle, and EMC.

DMTF[6] emphasis on interoperability management for cloud systems. It is the author of the Open Virtualization Framework (OVF) and runs the Open Cloud Standards Incubator.

IEEE P2301[7], Guide for Cloud Portability and Interoperability Profiles acts on standards-based options for application interfaces, portability interfaces, management interfaces, interoperability interfaces, file formats, and operation conventions.

Organization for the Advancement of Structured Information Standards (OASIS) Identity in the Cloud (IDCloud)[8] concentrates on profiles of open standards for identity deployment, provisioning, and management in cloud computing. It performs risk and threat analyses on collected use cases and produces guidelines for mitigating vulnerabilities.

Open Cloud Consortium[9] studies frameworks for interoperating between clouds and operation of the Open Cloud Testbed.

Open Data Center Alliance[10] unifies customer vision for long-term data-center requirements. It is an independent IT consortium developing usage models for cloud vendors.

Amazon Web Services (AWS)[11] delivers a scalable cloud computing platform. The most well-known of these services are Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3). Even though the platform being proprietary, these services APIs are *de facto standard* APIs currently available. Many of the existing cloud platforms provide an Amazon EC2 and S3-compliant APIs. Examples of cloud platform are OpenStack[12], Eucalyptus[13], and Apache CloudStack[14].

Standards Acceleration to Jumpstart Adoption of Cloud Computing[15] drives the creation of cloud computing standards by providing key use cases that can be supported on cloud systems that implement a set of documented and public cloud-system specifications. Sponsored by NIST.

The Open Group Cloud Work Group[16] works with other cloud standards organizations to show enterprises how to best incorporate cloud computing into their organizations.

[5]Cloud Storage Initiative: `http://www.snia.org/forums/csi`

[6]DMTF: `http://dmtf.org`

[7]IEEE P2301: `http://standards.ieee.org/develop/project/2301.html`

[8]IDCloud: `https://www.oasis-open.org/committees/id-cloud`

[9]Open Cloud Consortium: `http://opencloudconsortium.org`

[10]Open Data Center Alliance: `http://www.opendatacenteralliance.org`

[11]Amazon Web Services: `http://aws.amazon.com`

[12]OpenStack: `http://www.openstack.org`

[13]Eucalyptus: `http://www.eucalyptus.com`

[14]CloudStack: `http://cloudstack.apache.org`

[15]SAJACC: `http://collaborate.nist.gov/twiki-cloud-computing/bin/view/CloudComputing/SAJACC`

[16]Open Group Cloud Work Group: `http://www.opengroup.org/getinvolved/workgroups/cloudcomputing`

TM Forum Cloud Services Initiative[17] reflects on common approaches to increase cloud computing adoption such as common terminology, transparent movement among cloud providers, security issues, and benchmarking.

These standardization bodies do help build standards in different cloud fields like infrastructure management, storage, networking, and application.

OPEN VIRTUALIZATION FRAMEWORK

OVF is a packaging standard designed to address the portability and deployment of virtualization appliances [22]. The format standard formed by DMTF, an industry working group comprised by dozens of member companies and organizations, is a package structure that consists of a number of files: descriptor file, optional manifest and certificate files, optional disk images and resource files. In 2010 the DMTF's OVF standard was adopted as an American National Standards Institute (ANSI) International Committee for Information Technology Standards (INCITS) standard [23]. OVF makes it easier to move VMs around in the data center or from one cloud to another without losing virtual networking characteristics. Major virtualization vendors already support this standard, as is the case of VMWare ESX Server, Amazon Web Services EC2, Microsoft Hyper-V, Citrix Xen Server and Red Hat Kernel-based Virtual Machine (KVM). Most vendors just support OVF on importing and exporting features. Although users are nevertheless capable to convert from a proprietary virtualizaion format (e.g., Amazon Machine Image) to another format by exporting first to the OVF package followed by importing to the final format supported by the hypervisor.

CLOUD DATA MANAGEMENT INTERFACE

CDMI is a SNIA standard that defines the interface that applications can use to access cloud storage and manage the data stored therein by performing Create, Read, Update and Delete (CRUD) operations [24]. Through this interface, clients will be able to discover the capabilities of the cloud storage offering and manage containers and data. The CDMI specification describes REST Hypertext Transfer Protocol (HTTP) interface for accessing the capabilities of the cloud storage system, allocating and accessing containers and objects, as well as managing users and groups, implementing access control policies, attaching meta-data, moving data between cloud systems, exporting data via other protocols such as Internet Small Computer System Interface (iSCSI) and Network File System (NFS) [25]. Transport security is obtained thanks to the cryptographic protocol Transport Layer Security (TLS). There are several cloud storage interfaces under the change control of single vendors. As other storage cloud vendors do not want to depend on its competitors for interface changes, each one tends to create and disseminate their own storage interface. This leads to multiple interfaces that

[17] TM Forum Cloud Services Initiative: `http://www.tmforum.org/EnablingCloudServices/8006/home.html`

cloud consumers have to deal with if they need to consume cloud services from various storage cloud providers or locking into a specific service. CDMI on the other side is under change control of a standards body that does it best to accommodate requirements from different storage cloud vendors.

OPEN CLOUD COMPUTING INTERFACE

The Open Cloud Computing Interface (OCCI) is a set of open community-lead specifications from the Open Grid Forum. It's a RESTful protocol and API for all kinds of management tasks [26]. The specification was originally initiated to create a management API for IaaS model based services, allowing development of interoperable tools for common tasks including deployment, scaling and monitoring [27]. Later the group extended the focus to include other layers in the cloud stack [28]. OCCI began in early 2009 and initially lead by SUN Microsystems[18], RabbitMQ[19] and the Universidad Complutense de Madrid[20]. As of now, it counts with over 250 members including individuals, industry and academic parties. OpenStack, Apache CloudStack, Eucalyptus, OpenNebula[21], and RESERVOIR[22] are examples of projects implementing the OCCI API.

CLOUD INFRASTRUCTURE MANAGEMENT INTERFACE

Similar to OCCI, CIMI is a standardization effort within the DMTF consortium that specifies interactions between cloud environments although addressing infrastructure management [29]. Basic resources of IaaS (machines, storage and networks) are modeled with the goal of providing consumer management access to IaaS implementations and facilitating portability between cloud implementations [30]. It implements a REST interface over HTTP and defines an API supporting both eXtensible Markup Language (XML) and JavaScript Object Notation (JSON) data-interchange formats. Unlike the OCCI specification discussed earlier, the CIMI specification does not describe a formal structure. Instead, it describes parts of the process of a consumer choosing a configuration they want and deploy it.

[18]SUN Microsystems: `http://www.oracle.com/sun`
[19]RabbitMQ: `http://www.rabbitmq.com`
[20]Universidad Complutense de Madrid: `http://www.ucm.es`
[21]OpenNebula: `http://opennebula.org`
[22]RESERVOIR: `http://www.reservoir-fp7.eu`

TOPOLOGY AND ORCHESTRATION SPECIFICATION FOR CLOUD APPLICATIONS

The OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) Technical Committee[23] aspires to enhance the portability of cloud applications and services. The specification provides a language to describe service components and their relationships using a service topology and orchestration, invocation and management behavior, of IT services [31]. The combination of topology and orchestration enables preservation across deployments in different environments, empowering cloud interoperable deployment of cloud services and their management lifecycle (e.g., scaling, patching and monitoring) [32].

TOSCA is meta-model oriented language for defining Service Templates. A Service Template comprehends a Topology Template, Node Types, Relationship Types and Plans (Figure 2.3).

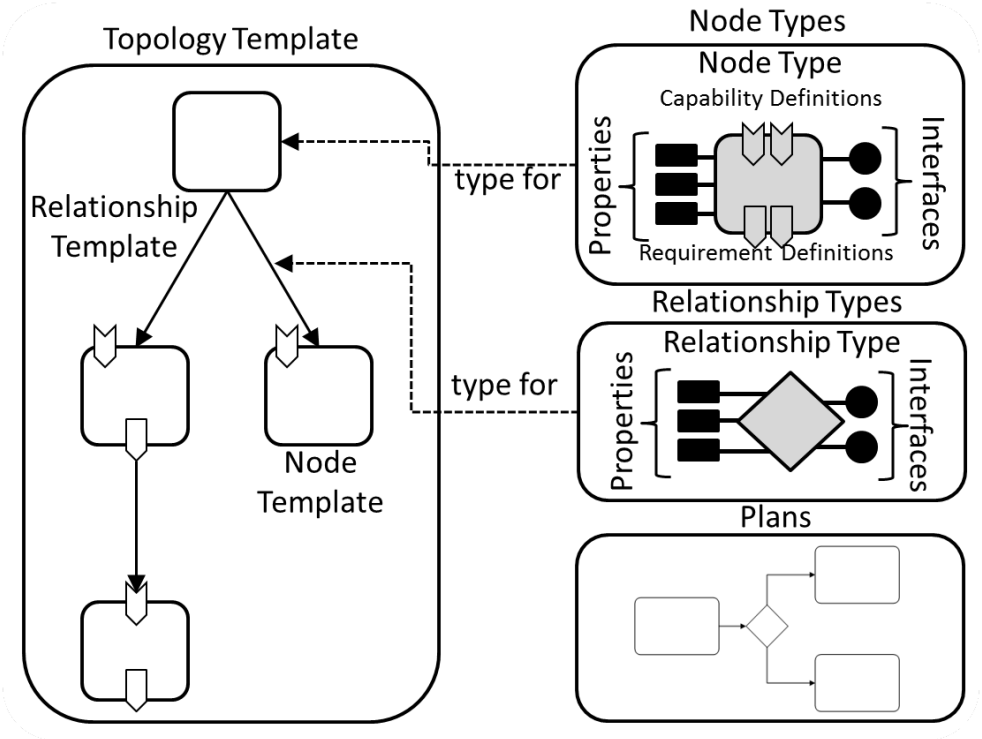

Figure 2.3: Structural Elements of a Service Template and their Relations in TOSCA

A Topology Template defines the structure of a service and consists of a set of Node Templates and Relationship Templates. A Node Template designates the occurrence of a Node Type that defines the properties of a component and operations of a service. Node Types are defined separately so that they can be reused. Plans are defined to orchestrate models that are used to create and terminate a service, and manage it during its whole lifetime.

[23]https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca

CLOUD APPLICATION MANAGEMENT FOR PLATFORMS

Cloud Application Management for Platforms (CAMP)[24] is a joint cloud specification by Oracle, CloudBees, Cloudsoft, Huawei, Rackspace, Red Hat, and Software AG submitted to the OASIS standards organization. CAMP is a standard self-service interface specification to be offered by PaaS CSP to manage the building, running, administration, monitoring and patching of applications in the cloud. It defines a model, a REST-based API for manipulating that model and a format for getting applications into and out of PaaS clouds [33]. It is targeted at application developers and deployers for self-service management of their applications.

2.5.2 SUMMARY

Cloud computing is the new trend in technology with industry starting to offer a panoply of cloud services. These new services are available through proprietary interfaces that interfere with customers' capabilities to easily move from one cloud to another. Standardization bodies aware of the problem started working on cloud interface specifications aimed at removing obstacles and improving communication over the Internet.

In a cloud management stack, OVF, TOSCA and OCCI or CIMI, and CAMP can all work together. TOSCA could be used to formalize a TOSCA service template centering on business requirements for a service, making sure its implementation is viable. Next an engineer decomposes the designed service and creates a virtual machine configured according to computing, network and software requirements packaging it in the OVF format. The OVF package is then deployed and begins working on a cloud via a OCCI or CIMI implementation. Remote management of SaaS services living in the virtual machine can be made possible through the use of the CAMP interface.

2.6 GROUNDBREAKING RESEARCH

In [34] it is identified indicators that the potential of cloud computing have not yet been sufficiently explored: federation and interoperability; better user support by specializing platforms as opposed to current generic platforms; ease of use by introducing modern programming models to the cloud; trusted clouds (centered on legislation, policy, security and privacy); and better manageability and efficiency mechanisms.

This section summarizes some state-of-the-art cloud computing research projects. The herein selected projects diverse in cloud computing fields from mobile cloud networking, cloud storage, cloud transactional memory and next-generation operating systems and how they can contribute to the advancement of cloud computing. Each of the following subsections

[24]CAMP: `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=camp`

introduces the needs for the inception of the project, and analyzes and discusses the proposed solution. Noteworthy these projects seize on at least one of the five indicators just mentioned.

Section 3.4.2 further extends innovative research projects on the field of cloud service brokerage.

2.6.1 MOBILE CLOUD NETWORKING

With the surface of numerous mobile devices, many cloud activities are being extended to cope with the trending mobile computing model [35]. Today's Telco industry is facing massive mobile data rates and consequently it needs to expand its infrastructure to react to customers demand and accommodate them. Although, scaling up their infrastructures can be troublesome as data centers are confined due to physical area constraints. Mobile Cloud Networking (MCN)[25] tries to tackle the aforementioned problem by leveraging cloud computing as infrastructure for future mobile network deployments and operations. Also by exploiting cloud computing, the project endeavors to reduce CAPEX and OPEX for mobile service providers as a mean to sustain average revenue per user [36].

The MCN project is working on extending the concept of cloud computing beyond data centers. It will create an atomic and fully cloud-based mobile cloud networking platform (Figure 2.4). Such platform will feature a mobile network, decentralized computing and storage offered as one service and following cloud computing principles (on-demand, elasticity and pay-as-you-go) [36]. MCN is structured in several segments. First it will take advantage of the OpenStack cloud platform as foundation for the infrastructural resources delivered as fully virtualized end-to-end MCN services. On top of this framework, extensions to the mobile network are envisaged to conceptualize wireless clouds and mobile core network clouds. A mobile platform for end-to-end mobile service deployments will be designed and developed. The platform will center attention on business needs, namely SLA management, authentication, authorization, accounting, charging, billing and content distribution services [36].

Following this architecture and deploying this platform, Mobile Telco providers' infrastructure will not anymore be restricted physically to a geographic area as different services can be provisioned, accessed and cooperate between them remotely. Providers would be now able to rent cloud computing resources from third party data centers and deploy their infrastructure on top of them, instead of building, managing and owning the whole infrastructure layers (e.g., networking, storage, bare-metal servers) below their real needs.

The MCN consortium is formed by major institutions in the cloud and telecommunications research and development industry such as SAP, Orange, Telecom Italia, British Telecommunications, Portugal Telecom Inovação, NEC and Intel.

[25]Mobile Cloud Networking: `http://www.mobile-cloud-networking.eu`

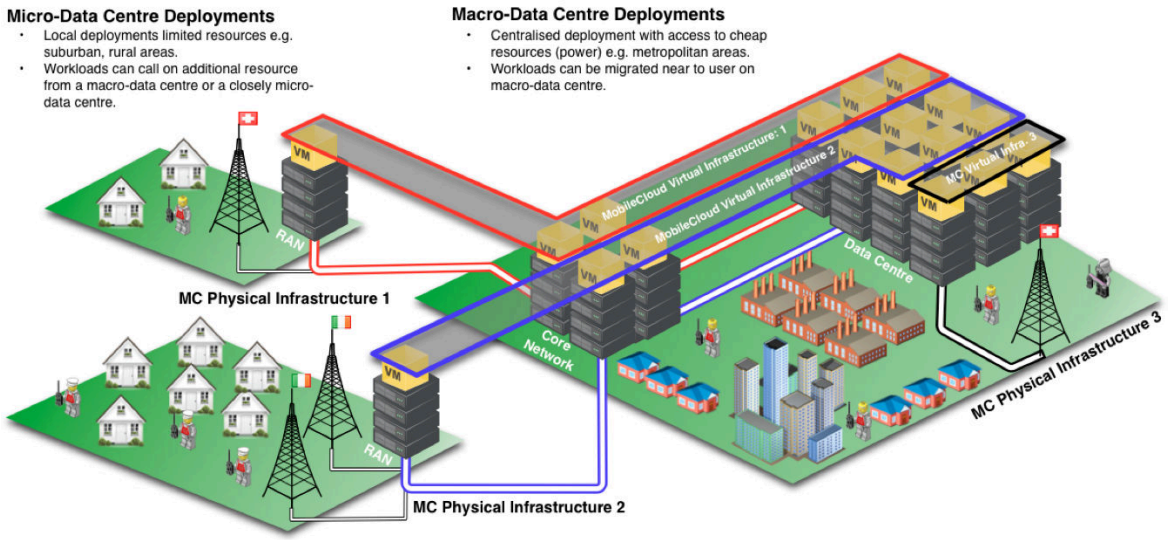

Figure 2.4: Example of a cloud-based mobile cloud networking platform

2.6.2 VISION CLOUD

The amount of digital data being generated by end-users and organizations is undoubtedly flooding today's networks. Data generation is at such an acceleration that users are now storing their data into the cloud. Cloud storage is not only being used for offloading data from users' computers but because of data accessibility anywhere provided by cloud computing. Storage capacity in the cloud is elastic so that individuals can rent storage depending on their needs over time. The same reasons apply to businesses. With the rise of large data centers and computational resources, organizations are storing data on the cloud, and offloading data-intensive computation to CSPs. This way organizations are given the chance to cut on large capital investments needed to build and provision large-scale computer installations [37].

VISION Cloud[26] fosters to architect a scalable and flexible cloud environment to address data-intensive storage cloud services [38], [39]. The VISION Cloud platform provides efficient support by taking a content-centric view of storage services. Further, it relies on standards to ensure better interoperability, namely SNIA CDMI (see Section 2.5.1), DMTF OVF (see Section 2.5.1) and Cloud Security Alliance (refer to Section 2.5.1) [40].

The project is driven by the following areas of innovation: a) raise the abstraction level of storage by managing content into objects with user-defined and system-defined attributes in a meta-data model; b) extend capabilities on data migration in current infrastructure to migrate and federate data across administrative domains to avoid data lock-in; c) bring computing closer to storage in order to enable secure execution of computational tasks near their data; d) enable efficient retrieval of objects by content and their relationships; and e) advance the state of the art of cloud-based storage by creating a secure platform for running data-intensive services.

An added advantage of VISION Cloud when compared to most current infrastructure

[26]VISION Cloud: `http://www.visioncloud.eu`

platforms is that it covers the need for users to define geographical constraints to where data should be hosted. Data can thus be in compliance with legal requirements for the country or region it is in. As per key innovation c) pointed above, VISION Cloud assumes that for securing data, computational resources must be physically close to storage. This leads to a coupling of storage with computation.

2.6.3 CLOUD-TM

One of the indicators stated in Section 2.6 as a need on and that could be exploited to take maximum advantage of clouds was modern programming models oriented for cloud computing. Moreover, tools that simplify the design and implementation of applications for the cloud environment are of the most importance. Presently there are several relevant programming models for tackling large-scale processing data analysis (e.g., MapReduce and Parallel DBMSs) [41], and Distributed Software Transactional Memory (DSTM) and parallel applications (e.g., D2STM [42], HyFlow [43]).

Specifically, the DSTM programming model pushes Software Transactional Memory (STM) model paradigm further. STM deals with aspects such as non-distributed, cache coherent and shared-memory systems on multi-core and symmetric multiprocessing architectures [44]. Threads can take a snapshot of memory, run isolated, and commit atomically. DSTM enhances STM by enabling this model for threads in different processes. DSTM is by itself, however, not capable of taking full benefit from cloud computing platforms' infrastructures, as very unlikely it will be able to handle scalability challenges on large cloud computing environments [44]. Cloud-TM[27] is a new paradigm that uses the STM programming model in the cloud computing paradigm [45].

Cloud-TM is a middleware platform that, jointly with the Fénix Framework, tries to simplify the development of services in cloud platforms. At the same time other goals include minimizing operational costs by automating provisioning of resources and maximizing scalability and efficiency of services through unmanned resources and workload fluctuation controllers.

[27] Cloud-TM: `http://www.cloudtm.eu`

CHAPTER 3

Brokering Cloud Services

With cloud computing in its early years, there are many researching initiatives undergoing. Realizing the potential cloud has to offer, industry has rapidly started doing business on cloud services. Currently most cloud service offers do not fully provide in a single platform cloud consumers all the features they need. Determining the most suitable way to procure, implement and manage cloud technologies can be challenging to users. Some users would like CSPs to enable them to manage and monitor their service as they would on their own data centers, while other users prefer data to be truly managed by them. Others demand that cloud contributes to their top business priorities (growth, costs, markets and geographies) [46], as comfortableness with cloud service increases.

Organizations will want to combine multiple cloud services from multiple providers to address those three business priorities, in a cloud-centric business process. Interconnecting on-premises solutions, outsourced applications and processes with applications and processes owned by partners or suppliers isn't really new as organizations have been integrating solutions for quite some time already. However, integrating and intermediating cloud-centric solutions goes beyond what they are offered with. This need has led to the rise of cloud service brokers.

This chapter explains what cloud service brokers consist of and how they are IT and business changers in Section 3.1. Afterwards three CSB topology options are illustrated and a comparison between them is made (Section 3.2). The significance of multi-cloud software tools are covered in Section 3.3 as well as described software libraries for development in several different programming languages. Section 3.4 gives an overview over currently available industry CSB products and looks forward to the future of cloud service brokerage platforms.

3.1 WHAT IS CLOUD SERVICE BROKER

Leading technology analyst Gartner[1] defines a CSB as an entity that manages the use, performance and delivery of cloud services and negotiates relationships between cloud providers and cloud consumers [47]. It also identifies the following characteristics for a platform to be considered a CSB [48]:

- **Must broker cloud services:** a CSB does not have to deliver cloud services but must broker at least one cloud service
- **Must have a direct contractual relationship with the service consumers:** from a business model point of view, a CSB has a direct relationship with the consumer of the provisioning of a cloud services-based expertise, technology or business outcome.
- **May or may not have a direct contractual relationship with service providers:** a CSB may aggregate a mix of free and fee-based services. For a free service a contractual relationship may not exist.
- **May be any legal entity:** such as an independent CSPs, government agencies, joint ventures or internal IT organization of a company
- **Must intermediate on behalf of one customer to many services or many customers to one service or both**
- **Can support design, build or run phases of the cloud-based solution:** the cloud broker adds non-trivial value on top of the original CSP.

Advantages of using a brokering entity may include a) help users determine the best framework based on a number of factors such as provisioning assistance and budget guidance, and identify and integrate disparate services across multiple CSP; b) simplified interface including Single Sign-On (SSO) and interoperability benefits, hiding the complexity inherent in working with multiple cloud providers; and c) cost-effective resources.

A CSB may assume one or all three primary CSB roles: intermediation, aggregation and arbitrage [49]. In regard to intermediation, a CSB provides value added services on top of existing cloud platforms, such as identity, access management, performance reporting or enhanced security capabilities. Aggregation is about providing the data integration, process integrity or intermediation needed to bring multiple services together to cloud organizations or individuals. Data is modeled across all component services and integrated as well as ensuring the movement and security of data between the service consumer and multiple providers [49]. Cloud service arbitrage provides flexibility and opportunistic choices by offering multiple similar services to select from. Arbitrage is a complementary function of aggregation in the

[1]Gartner: `http://www.gartner.com`

sense that the broker moves cloud workloads from one cloud service to another based on cost and other factors.

There is a high chance that organizations will need to have services with multiple CSPs. The need to provide central governance as a result is critical. A CSB streamlines the provisioning and management of these services and allows consumers to negotiate terms and configure integration upfront, minimizing the need for that type of interaction on a per-request basis. Another plus side of a CSB is the elimination of vendor lock-in as the ability to move infrastructure or platform from one CSP to another becomes a push button action. Prior to CSBs, this wasn't a lightweight task to the IT department. Steps to migrate would involve snapshotting the service, get the snapshot off the provider, moving it to a new location, secure information, and move metric reporting. All these steps can be accomplished with broker capabilities, saving time, money and IT resources.

CSBs allow for the rapid deployment of new services, which optimizes the business impact of both internal enterprise users and external users. The CSB can integrate, consume and extend cloud services, providing more services from providers. The accessibility of a CSB centralizes billing and empowers IT groups to stay focused on their technical job. Another advantage of brokers to organizations is the ability to supply visibility across all the services it provides. Such capability truly enables the governance of external cloud usage. The centralization of all these components facilitates an integrated billing, reporting and management of services. Before cloud service brokers, management of systems required the use of multiple interfaces with no cohesive single interface. Billing could change from department to department depending on how each negotiated prices. A single destination provides business with the information they need across all implementations and gives IT information it needs to verify that SLAs are being met.

3.2 CLOUD SERVICE BROKER TOPOLOGY OPTIONS

As organizations evolve to a "as a service" model embracing various public clouds and private cloud services, they need a standard framework for delivering and managing their distributed services across organizations. This ensures consistent collaborative governance and compliance across disparate services. Aggregating services increases buying power thus reducing costs and ensuring optimum license allocation across users.

There are three types of CSB topology: internal "IT as a Broker", cloud based and hybrid. In an Internal "IT as a Broker" topology, an in-house CSB platform is installed internally in an enterprise. Cloud consumers are enforced to use the internal CSB to consume from or deploy to the cloud (Figure 3.1). This topology model is preferred when organizations desire to have full control in-house. Apart from having to maintain the platform and its interoperability CSPs, internally they are only dependent on themselves to communicate externally. This means communication between cloud consumers and CSPs will only break if they decide to

disrupt the internal CSB API and the former API gets obsoleted and shutdown.

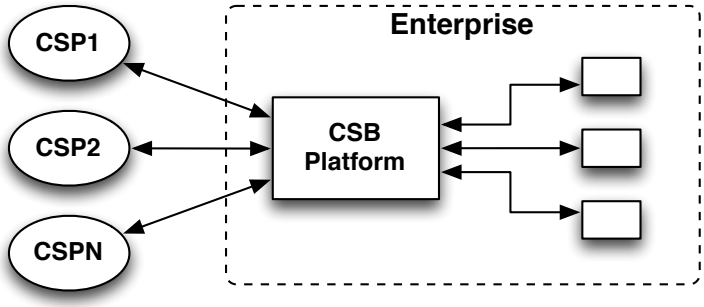

Figure 3.1: Internal "IT as a Broker" CSB topology

The difference between a cloud based topology and the internal "IT as a Broker" topology is more than an formal change where the CSB now is out of organization premises. The CSB platform is independent from the organization and because of that it affects how business and IT operations work. The most perceptible impact is that being an external platform to the company, the IT department can focus on working on customer products rather than on internal tools and platforms to accelerate and ease development tasks. Cloud services brokering is delegated to a CSB platform maintained by external entities. Depending on various factors such as size of the company and dependency on the cloud, outsourcing this function to another company may be cost beneficial. Even though API compatibility from the CSB to the cloud consumer should be safeguarded for a long period of time, there is no guarantee that at some point in time it won't be broken. The cloud consumer will have to adapt their services to the new API.

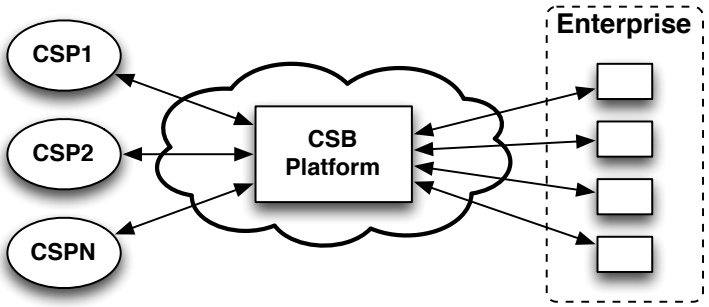

Figure 3.2: Cloud based CSB topology

A hybrid topology combines the two previous topology models described above. Enterprises own an internal CSB platform to be used by their cloud services. The inner CSB cooperates with an outer CSB to broke services to CSPs. When comparing this solution to the internal "IT as a Broker" topology, the enterprise only depends on one external API and is still able to deploy to multiple CSPs. The enterprise's broker may be substantially technically simpler

because demanding operations can be leveraged to the external broker (eg. orchestration, auto-scaling, reporting, billing, authentication and authorization, and policy enforcement). Nevertheless, depending exclusively on a single third-party service is adverse. The brokering service may experience network outages or run out of business. It is then an additional point of failure and latency. Figure 3.3 illustrates a hybrid CSB topology scenario.

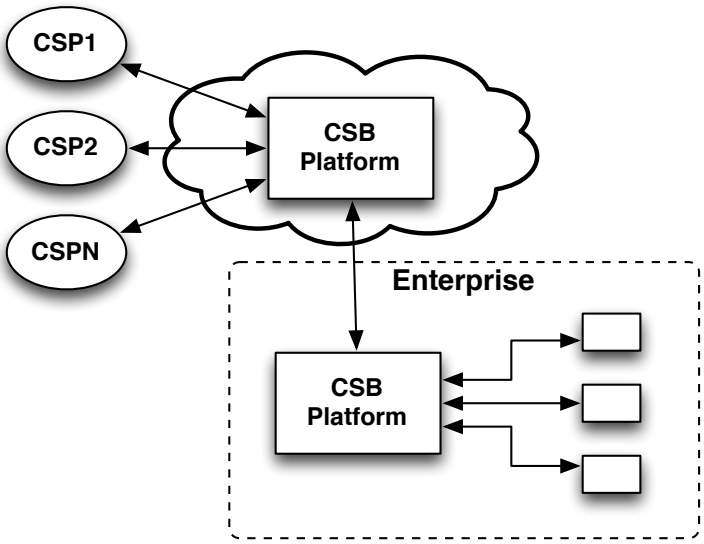

Figure 3.3: Hybrid CSB topology

3.3 MULTI-CLOUD TOOLS

Interaction with numerous CSP of the same cloud layer can be chaotic to developers. For example, many use diverse vocabularies for the same cloud term, incompatible APIs and object models, and authentication mechanisms. These factors lead to inconsistencies from one cloud platform to the next, and thus constraining cloud interoperability issues. Multi-cloud tools work as agnostic entities of abstraction bringing homogeneity across different clouds to users. CSB implementations can use these tools to rapidly and easily consume services from CSPs.

JClouds[2] is a cloud agnostic library for the Java Virtual Machine (JVM) that enables developers to access a variety of supported cloud providers using one API. The library supports over 30 cloud providers and cloud software stacks including Amazon, Azure, Rackspace, GoGrid, HP and OpenStack. JClouds offers compute API and blobstore API abstractions as Java and Clojure[3] libraries. JClouds is an Apache Incubator[4] and open-source project.

[2]JClouds: `http://jclouds.incubator.apache.org`
[3]Clojure: `http://clojure.org`
[4]Apache Incubator: `http://incubator.apache.org`

Deltacloud[5] is a Ruby library for cloud services. It provides the API server and drivers necessary for connecting to cloud providers. It enables the management of cloud resources in different clouds through one of three supported APIs: Deltacloud API, DMTF CIMI API or Amazon EC2 API. The software is a REST-based cloud abstraction API that claims to maintain long-term stability and backward compatibility across different versions. Deltacloud provides functionality for computational and storage services over 20 cloud services. The projected was founded by RedHat and later incorporated into the Apache Software Foundation.

Fog[6] is yet another Ruby cloud services library. Fog provides an accessible entry point and facilities cross service compatibility to compute, Domain Name System (DNS), storage and Content Delivery Network (CDN) cloud services. Fog has become very popular lately, and serves as the backbone for Chef's cloud computing functionality and many other projects.

Libcloud[7] is a Python cloud computing library that abstracts away differences among multiple CSP APIs. It is capable of managing four different cloud resources: servers, storage, load balancers and DNS. More than 30 cloud platforms are supported. Libcloud supports Python versions from 2.5 to version 3. It is also part of the Apache Software Foundation ecosystem.

Dasein[8] is a cloud abstraction layer for Java. The abstraction model enables programmers to build applications for IaaS and PaaS services. A meta-data layer enables applications to dynamically discover the capabilities of the cloud provider and thus to create conditional logic based exclusively on the Dasein model. Dasein is an open-source project under the Apache Software License and is sponsored by Dell.

Pkgcloud[9] is a standard library for node.js[10] that allows interaction with multiple cloud service providers. It supports compute and storage services from Joyent, Microsoft, Rackspace, several database providers like MongoHQ and RedisToGo, and DNS services. The API follows the asynchronous philosophy of node.js. CDN services and more service providers are on the roadmap of the project.

Gophercloud[11] is a multi-cloud language binding for Go[12]. Being developed by Rackspace it naturally provides an Openstack-compatible Software Development Kit (SDK). An official release has not yet been released although Gophercloud is being actively developed since June 2013.

These projects are visible efforts focusing on proposing thin layers of uniform APIs for certain groups of cloud providers. These solutions have been partially adopted to IaaS level but are not completely solving the problem of porting from one cloud to another at the PaaS

[5]Deltacloud: `http://deltacloud.apache.org`
[6]Fog: `http://fog.io`
[7]Libcloud: `http://libcloud.apache.org`
[8]Dasein: `http://dasein.org`
[9]Pkgcloud: `https://github.com/nodejitsu/pkgcloud`
[10]Node.js: `http://nodejs.org`
[11]Gophercloud: `https://github.com/rackspace/gophercloud`
[12]Go: `http://golang.org`

level. Currently, application portability is only possible between a small number of PaaS providers sharing the same PaaS platform.

3.4 CLOUD SERVICE BROKER SOLUTIONS

CSBs are emergent cloud approaches responsible for delivering cloud interoperability. Cloud service brokerage is a business model in which companies add value to one or more cloud services on behalf of one or more cloud providers to cloud consumers. Existing cloud services are multiplying and expanding faster than the ability of cloud consumers to manage them. CSBs help organizations select, manage and coordinate the various services they require. While the concept of cloud computing is simple, concluding how cloud best fits an organization's needs can be incredibly intricate. There are many different cloud computing solutions and it can be quite intimidating to figure out what exactly one needs.

This section overviews diverse CSB solutions. Section 3.4.1 presents currently available industry solutions on discrete cloud service brokerage branches. While listing leading CSBs industry products, a summary of what each one as to offer and differentiator features is made. Section 3.4.2 presents some research projects towards novel cloud service brokerage solutions.

3.4.1 INDUSTRY SOLUTIONS

Industry companies have already started making business out of cloud service broker solutions. These brokerage products do not necessarily operate on the same cloud service branch, but range from brokering third-party cloud services on a marketplace, cloud service brokers aggregating SaaS applications to cloud service brokerage enablers for multi-tiered cloud services.

AppDirect[13] is a cloud broker offering marketplace and a management platform that enables companies to distribute web-based services, both for well-known service providers and for businesses that are creating their own brokerages of cloud applications. It has more that 150 applications represented in its catalog and offers centralized billing, helping business keep better track of what they are spending on cloud applications and services. Its integration support for Microsoft Office 365 is the company's core differentiator.

Cloud Compare[14] is a cloud services brokerage that helps organizations run business applications. The service helps introducing cloud through planning, research, independent expert advice and project management. Their Cloud Adoption Framework for SMEs framework assists businesses moving data, applications and business processes to the cloud, by helping

[13]AppDirect: `http://www.appdirect.com`
[14]Cloud Compare: `http://cloudcompare.ie`

customers understand operational processes and making recommendations based on the knowledge of how an enterprise works.

Cloud Sherpas[15] helps organizations adopt cloud solutions in the cloud in different ways. Examples are messaging, collaboration and CRM, designing of cloud strategies, streamline account management of multiple clouds and integrate existing cloud solutions with other cloud implementations services and business systems.

HP Aggregation Platform for SaaS[16] is a platform targeted to serve as the single point of access for all SaaS applications. Operators are enabled to create a SaaS marketplace portal to expedite their delivery to Small and Medium Business (SMB) customers. As a result, customers can discover SaaS products and bundles, run trials, subscribe to services and consume them. The platform eases the integration of IT SaaS and hosted services into the communication service provider environment. The portal is integrated in a cloud governance layer that provides IaaS services with access to software, network, storage and processing templates.

Nephos Technologies[17] is an independent CSB platform, acting as independent advisers, to help manage, automate and orchestrate infrastructure deployments in the cloud. It provides consultancy to customers on developing cloud strategies, helping deployments to be PCI[18] compliant, moving to object based storage, big data analytics, cloud backup and archive.

CLOUDSolv[19] is yet another comprehensive tool for selling SaaS solutions in one marketplace for all hosted solutions services. CLOUDSolv is a CSB aggregator platform for Managed Service Providers and Value Added Resellers and enables automated provisioning and billing.

Appsecute[20] is a management portal that enables IT operators, developers and management to monitor and maintain applications and services across Cloud Foundry[21] based PaaS clouds. A real-time status is combined with a Facebook-style timeline to allow teams to see everything that's happening with their software components and applications. In June 4, 2013, Appsecute was acquired by ActiveState[22] [50].

To help companies become CSBs there is Jamcracker[23] with their Jamcracker Services Delivery Network platform. Businesses can get help from Jamcracker on unifying the delivery of public, private or hybrid cloud services to their customers and collaborators. It provides a multi-tiered and multi-tenant cloud delivery and life-cycle management platform that offers a way for purchasing and managing cloud services available as a catalog with cloud services with provisioning, billing, security, user administration and support frameworks. Modular APIs

[15]Cloud Sherpas: `http://www.cloudsherpas.com`

[16]HP Aggregation Platform for SaaS: `http://h20229.www2.hp.com/partner/ngsd/HPAP4SaaS.html`

[17]Nephos Technologies: `http://www.nephostechnologies.com`

[18]PCI: `http://www.pcicomplianceguide.org`

[19]CLOUDSolv: `http://www.synnex.com/cloudsolv`

[20]Appsecute: `http://appsecute.com`

[21]Cloud Foundry: `http://www.cloudfoundry.com`

[22]ActiveState: `http://www.activestate.com`

[23]Jacmcracker: `http://www.jamcracker.com`

can be used to develop cloud adapters for content, web services and applications. Jamcracker enables the platform to be provisioned whether internally on companies' premises or externally.

3.4.2 RESEARCH TOWARDS NOVEL SOLUTIONS

Research groups have been exploring new ways towards cloud service brokerage to advance cloud computing. There are several research projects working on multi-cloud and multi-tier cloud services, cloud interoperability, full provisioning and management automation of services on the cloud.

MOSAIC

The mOSAIC project[24] is an open-source API and platform for multiple clouds that enables application developers to select cloud services according to their application needs. Developers don't need to choose a cloud provider at design time and develop applications based on that decision [51]. Instead, mOSAIC enables developers to postpone their decision about which cloud provider best fits their needs from design time until run-time. The mOSAIC platform can automatically provision applications across a set of clouds, without the user's direct involvement. The automatic provisioning and controlling of the application brokering is only possible through the predefined SLA. The SLA must contain indicators at component and application level.

A component called Resource Broker (RB) is the component in mOSAIC for application brokering. It is responsible for all mediation between clients and cloud providers. RB encloses an entity responsible for discovery and negotiation with cloud providers called Cloud Agency (CA), and a Client Interface (CI) in charge of requesting additional resources from the Application Executor (AE) component. The AE component manages the deployment, execution and monitoring of applications [52]. The Cloud Agency (CA) is the component in mOSAIC's architecture promoting pricing awareness by taking into consideration and negotiates on the pricing policies of each CSP.

mOSAIC has some constraints on the architecture of the deployed applications. It assumes that the application is divided into components with explicit dependencies in communication and data between them. The application architecture must be SOA and must only use the mOSAIC API for inter-component communication. Other types of communication are forbidden (e.g., direct sockets) [53].

[24]mOSAIC: `http://www.mosaic-cloud.eu`

STRATOS

STRATOS[25] is a cloud service broker which promotes the deployment and runtime management of cloud application topologies using cloud elements and services [54]. Provisioning is sourced from multiple cloud providers based on specified requirements. Cloud consumers describe the application topology and requirements in a Topology Descriptor File (TDF). The TDF includes structural information, monitoring directives, management directives and the deployer's set of constraints, requirements and preferences. STRATOS is forced to honor the terms from the TDF and because of that the TDF can be considered as an SLA document between the two entities, client and broker. The client submits the TDF to the CloudManager as illustrated in Figure 3.4.

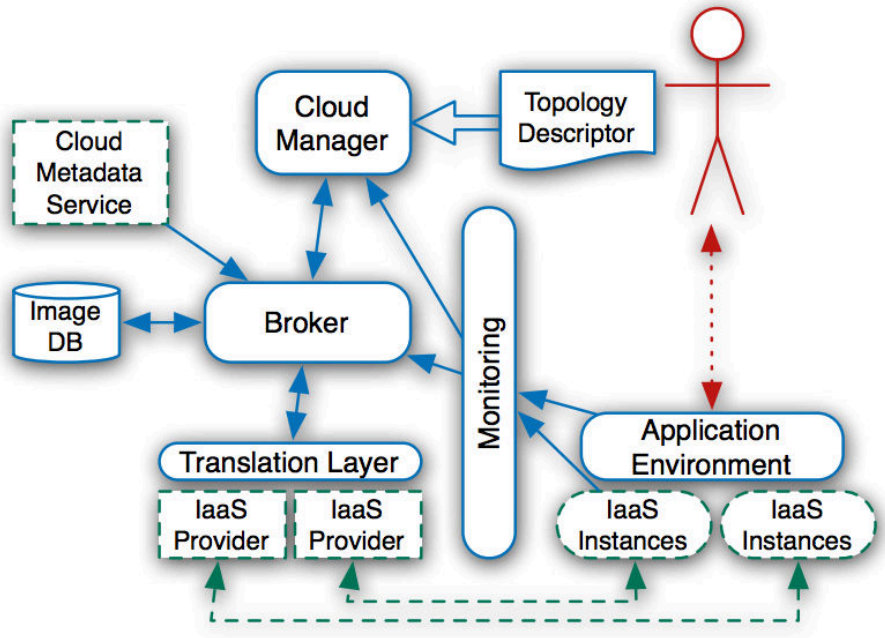

Figure 3.4: STRATOS cloud management framework

The CloudManager contacts the Broker component, which initiates the topology and calculates the most efficient allocation of resources across the providers. Such calculation is based on the TDF specifications. The topology can be built on a single cloud provider or partially sourced from multiple cloud providers. Monitoring information is used by the CloudManager to make on the fly elastic decisions and by the Broker to assist in the decision process. The Broker uses the Translation Layer to access assorted clouds through an API that abstracts the different cloud APIs.

The current implementation has the disadvantage of requiring that the client create accounts with each provider and include the access credentials at deployment time [54]. It also doesn't mention geo-location nor policy awareness that could be translated to the TDF as constraints.

[25]STRATOS: `http://wso2.com/cloud/stratos`

COMPATIBLEONE

CompatibleOne[26] is an open source and collaborative research project funded by French institutions. It offers an interface for allowing the description of user cloud needs and provisioning on the most appropriate CSP. The scope of CompatibleOne addresses all three layers of cloud computing stack. It provides organizations unrestricted access to cloud technologies and removes constraints of vendor/data lock-in.

CompatibleOne comprises a model and an execution platform. The model, CompatibleOne Resource Description System (CORDS), is an object oriented language for the description of cloud applications, services and resources. Advanced Capabilities for CORDS (ACCORDS) is a cloud application provisioning and deployment control system for federation of different cloud systems.

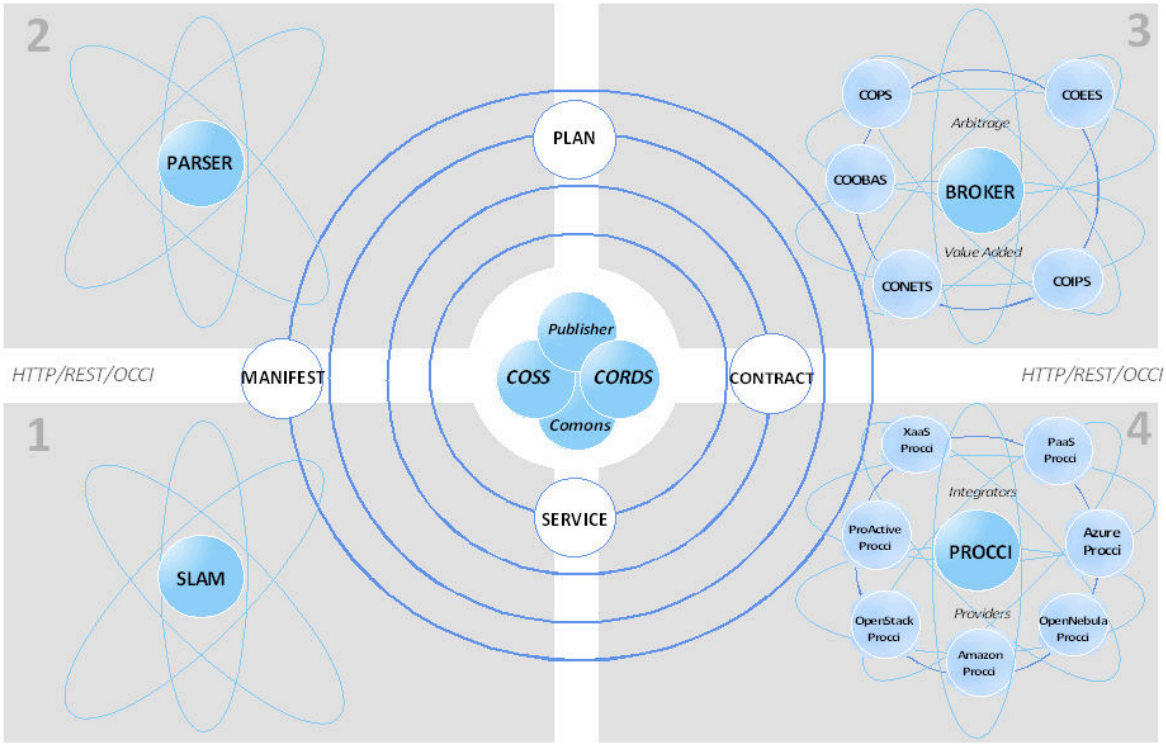

Figure 3.5: CompatibleOne ACCORDS platform architecture

CompatibleOne workflow encompasses four steps (Figure 3.5). First consumers describe their requirements in a form of a manifest file. There it is described the required technical and economical criteria. The structure of the manifest file corresponds to a model based on OCCI. In the second step, the corresponding provisioning plan is built and validated given the manifest. The manifest is received by the engine. The engine gathers required data for accounting and billing, and then passes the manifest to the ACCORDS Parser. The Parser transforms the XML of the manifest file into provisioning plan. The parser runs a syntactic verification. The provisioning plan is transmitted to the Broker of which is held charged for

[26]CompatibleOne: `http://www.compatibleone.org`

executing the provisioning plan. The provisioning plan is transformed into contracts with selected providers. Fourth and final step, the Broker coordinates the activity of the PROCCI for the selection of the specific CSPs. The PROCCI provisions the delivery of the services required by the consumer.

BROKER@CLOUD

Broker@Cloud[27] is a Seventh Framework Programme (FP7) project intended to deliver a brokerage framework. The envisaged framework will allow cloud intermediaries to provision their platform with a set of methods and mechanisms for enabling continuous quality assurance and optimization of software-based cloud services.

The broker is formed by four core components. The service governance and quality control component is responsible for the lifecycle management of the applications, dependency tracking, policy compliance, SLA monitoring and certification testing. The failure prevention and recovery component enables the framework with features such as event monitoring, reactive and proactive failure detection, adaptation analysis and recommendation. The third is a service optimization component for continuous optimization of cloud services. This includes optimization opportunity detection and analysis based on cost and quality. The fourth component implements framework interfaces for an abstracted platform description of cloud services. The description contemplates technical, operational and business aspects, and static and dynamic views.

Broker@Cloud will open source most of its software and the validation of the project will result by integrating two different cloud service delivery platforms (CAS Open[28] and SingularLogic Galaxy[29]). Project consortium includes, but is not limited to, CAS Software[30], SingularLogic[31], SAP AG[32], and The University of Sheffield[33].

[27]Broker@Cloud: `http://www.broker-cloud.eu`
[28]CAS Open: `http://www.cas-crm.com/products/cas-open/summary.html`
[29]SingularLogic Galaxy: `http://www.slgalaxy.eu`
[30]CAS Software: `http://www.cas.de`
[31]SingularLogic: `http://portal.singularlogic.eu`
[32]SAP AG: `http://www.sap.com`
[33]The University of Sheffield: `http://www.shef.ac.uk`

OPTIMIS

OPTIMIS[34] is yet another FP7 project. The project is partnered by 14 institutions including SAP, British Telecom[35], Fraunhofer SCAI[36], University of Leeds[37], and Universitaet Stuttgart-HLRS[38] . OPTIMIS is a toolkit that enables organizations to automatically externalize services and applications to trustworthy CSPs in the hybrid model. The toolkit consists on a set of components: the Service Builder, the Basic Toolkit, the Admission Controller, the Deployment Engine, the Service Optimizer, and the Cloud Optimizer [55]. The Service Builder enables developed services to be delivered as SaaS and the Basic Toolkit provides functionalities common to components. The Admission Controller is the component that formulates and solves optimization problems for finding the optimum allocation of the services to be deployed. The Deployment Engine is responsible for the discovery and negotiation with cloud providers for provisioning services. The Deployment Engine is the one that initiates the deployment of the service across selected clouds by using the appropriate infrastructure optimized and allocated by the Cloud Optimizer component. The Service Optimizer component is then responsible for monitoring the service parameters for SLA violations.

OPTIMIS is a peer-to-peer federated inter-cloud project that deploys its tools in data centers and complement cloud management and orchestration platforms. In a peer-to-peer federation, cloud providers communicate directly with each other transparently. It also supports independent multi-clouds if clients use the Deployment Engine and the Service Optimizer directly to run services within multiple CSPs. The handicap here is that OPTIMIS agents have to be deployed in cloud providers' infrastructure and that may be something they do not wish to participate in.

[34]OPTIMUS: `http://optimis-project.eu`

[35]British Telecom: `http://http://bt.com`

[36]Fraunhofer SCAI: `http://scai.fraunhofer.de`

[37]University of Leeds: `http://leeds.ac.uk/`

[38]High Performance Computing Center Stuttgart (HLRS) of the University of Stuttgart: http://hlrs.de

CHAPTER 4

Solution for a Cloud Service Broker

This chapter presents a proposed architecture for a CSB platform aiming at recommendation and execution of services on the cloud. The platform suggests users which PaaS provider most suits their application needs through a recommendation process, deploys, manages, migrates and monitors applications on different cloud service providers. The entire application life cycle orchestration is exposed to users via a public and abstracted API or, optionally, via a Web portal or command line interface.

Section 4.1 specifies conceptual features that the CSB should offer to its users, while Section 4.2 proposes an architecture for implementing the CSB platform.

4.1 SPECIFICATION

Apart from user interfaces on where the interaction between users and the platform occurs, otherwise the platform would be externally unaccessible, three key requirements for the proposed CSB solution were identified: a) recommendation, b) application provisioning and management, and c) extensibility on CSP offerings.

The CSB should assist users on choosing the best PaaS to run users' applications by recommending them one or more PaaS CSPs based on application requirements. For such assessment, the platform compute the application requirements against each and every supported PaaS offering. Section 4.1.1 further details on recommendation where recommendation algorithms are enumerated. After recommendation, application provisioning and management takes place (Section 4.1.2). The CSB should supply users the means to push their applications to the cloud and manage them by providing an interface. The interface should be preferably a machine-readable one that supports powerful machine-level access and interoperability while

remaining user-friendly. Later on, human-readable interfaces can be developed on top of the initial interface and made available to users and third-party developers. Last but not least key requirement identified is the extensibility on CSP. The CSB architecture should neither be designed nor implemented in such a way that confine PaaS offerings to the set offered on its first release. This means the CSB platform operator should be able to extend existing offerings in a way that does not involve redesigning the architecture to add up new PaaS solutions. Rather, the architecture and implementation should be modular and flexible enough to comply with this specification (Section 4.1.3)

4.1.1 RECOMMENDATION ALGORITHM

The recommendation algorithm is a key property of a CSB. It comprehends the capability to recommend users with a set of cloud providers that best match users and applications requirements, either on a technical, functional and geographic level. The CSB should support and prioritize dynamically multiple criteria requested by users and allow the platform operator to add, remove or modify policy recommendations.

Next sub-sections unveil various criteria to assist cloud consumers choosing a cloud provider of their best interest. With exception of the first criteria, all remained criteria are sorted with no particular order.

PAAS AND APPLICATION TECHNICAL CHARACTERISTICS

Depending on the application type it is first mandatory to check its technical requirements. Technical requirements imply that without a all-inclusive cloud offering match the provider can be rejected in advance with no further consideration. Technical requirements are defined as runtime (Java, Ruby, Python, ASP.NET, etc), framework (Django, Ruby on Rails Grails, etc) and services (databases such as MySQL, PostgreSQL, Microsoft SQL, Redis and MongoDB; periodical task execution, logging, etc). These are unavoidable requisites that must be supported.

MANAGING PERFORMANCE OF PLATFORM AS A SERVICE PROVIDERS

Monitoring applications is essential for identifying and measuring performance issues. Also of equal importance is monitoring the cloud provider where each application is hosted. Application Performance Manager (APM) are good options to monitor and report. Typical cases of metric monitoring are CPU load, memory and traffic.

ACCOUNTING, BILLING AND BUSINESS MODELS

Business models vary significantly among PaaS cloud providers. A more or less exhaustive dissection of their models according to the application to be provisioned can bring up notable monthly savings to consumers. Examples of different business models are how providers account resource usage; some bill for number of CPU cycles, time of execution or memory. An example of such influence on the monthly cost of running applications on the cloud is an application that by itself does not require intensive computational resources but is a 24/7 running instance. In this situation, when comparing two or more cloud providers that are somehow tight in scoring, the final decision is definitely on the cost saving factor.

NEW PLATFORM AS A SERVICE PROVIDERS

Due to the trend of migrating information systems to the cloud, it is expected that new cloud providers also start emerging new added value services. It is clear the need of continuous monitoring of recent offerings so that consumers can both ensure they are running their applications on the cheapest and most reliable cloud provider. With that in mind, we extract a new recommendation criteria. Whenever a new PaaS provider is available and supported, the CSB should automatically and autonomously re-evaluate each user and application needs. In case a new provider offering matches or surpasses the current provider, users should be notified and let to decide whether it is of their interest to migrate to the new one or not.

UPDATED PLATFORM AS A SERVICE PROVIDER FEATURES

Similar to 4.1.1 where it is discussed new business opportunities, it is equally perceived the necessity to re-evaluate updated offerings of existent cloud providers. Added and updated technologies, accounting and business model changes, or any other worthy aspect that affects cloud consumers and the operator of the platform should be taken into consideration. Again the recommendation algorithm should be executed and inform consumers of updated solutions if necessary.

USER PROFILE

The CSB should differentiate between regular and gold users. While gold users have access to exclusive PaaS offerings, regular users will be limited to a restricted set only. It should be possible to hide PaaS providers with low confidence to business customers or, on the other hand, hide PaaS providers with high reputation and quality of service to end customers.

GEOGRAPHICAL LOCATION

Knowing the geographical location of where applications are going be provisioned is of the utmost importance. Physical and logical topology issues (distances between nodes, physical interconnections, transmission rates, and/or signal types) have impact on network latency. Social, economic and political issues are also to be taken into account; having user's data bounded to a particular geographic area due to territorial jurisdiction may be an imposed requirement.

4.1.2 APPLICATION MANAGEMENT

Application management should include actions such as creation, deployment, state switching (start, stop, restart and delete), elasticity, services, monitoring and logging.

CREATION AND DEPLOYMENT

Provisioning an application to the cloud involves first creating its environment in the broker, upload the code into it and then send it to the chosen PaaS provider. The application environment is a place to store the code, preferably in a Version Control System (VCS).

Any time users want to deploy a new version to the cloud, all that should be done is to update the code in the VCS and from there the broker should trigger a process to redeploy it on the PaaS provider.

STATES

There were identified six states that applications can be at. The application life cycle illustrating these states and actions is shown in Figure 4.1.

Created: the application environment was created. Users can upload code to the CSB;

Deploying: transactional state representing the period of which the CSB is deploying the application to a PaaS provider;

Running: application is running and accessible from the Web;

Stopped: application has been stopped. Could have been forcefully stopped due to some error or ordered by user;

Restarting: state in which the application is restarting in the current PaaS provider deployed;

Deleting: instructions were given to delete the application; application will be first undeployed and deleted on the PaaS provider and then on the CSB.

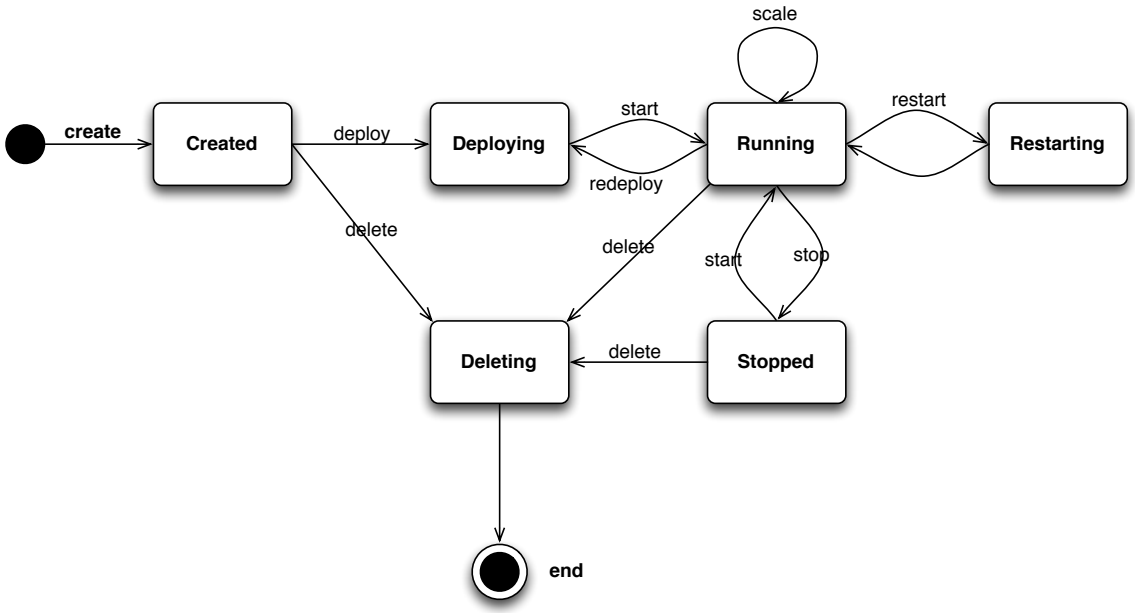

Figure 4.1: Application lifecycle

ELASTICITY

Application performance throughout its lifetime is subject to load from user's requests. Time to process a request and respond widely diverge depending on internal and external factors. Internal factors are those controlled by the developer and have ramifications due to, for instance, a bad architecture or implementation design resulting in heavy resource usage; external factors can be associated to network congestioning or high server loading caused by over-provisioning or peak-hours.

Providing users with the ability to quickly ramp up additional resources or instances of the application is imperative and the CSB should offer this to users by intermediating with PaaS providers.

SERVICES

The Cloud Service Broker must be able to offer users the ability to request some, if not all, services available from PaaS providers and attach them to their applications. Services are additional software resources such as databases, job schedulers, DNS, mobile, search, logging, email and Short Message Service (SMS), workers and queuing, analytics, caching, monitoring, media, utilities, and payments services.

MONITORING AND LOGGING

Users should be able to query the broker for the status of their application. Monitoring and logging features must be available, since almost every PaaS provider gives users this

functionality. Additionally, the CSB should notify users in case of unexpected events (e.g., application stopped responding or PaaS provider outage).

4.1.3 EXTENSIBILITY ON PLATFORM AS A SERVICE OFFERINGS

Even when considering an initial implementation of the CSB supporting as many possible PaaS providers, the platform must be extensible enough in order to easily later add new PaaS providers, either with existing and compatible APIs or new ones. This should look like a plug-and-play driver approach.

Also of particular interest is to consider the possibility of not covering all users' requirements from the set of provider offerings. Examples of this can be geographical considerations related to the data centers that underlie the clouds: physical location, available resources, and jurisdiction that data is under. For this reason it is important to extend cloud services with regard to a platform to host and run Web applications to those singular occasions by bootstrapping PaaS platforms on top of IaaS cloud solutions and nevertheless still abstracting CSB users of the internal configurations to setup a platform able to cope with their needs.

4.2 PROPOSED ARCHITECTURE

In this section it is proposed an architecture for a cloud service broker in accordance, but not limited, to the specification in Section 4.1.

The CSB framework, and depicted in Figure 4.2, consists on four layers. The PaaS and IaaS managers support multi-provider and multi-cloud environments using each a single API, orchestrated by the intelligent CSB. Users have access to a Web portal and command-line interfaces where they can perform arbitrated operations on resources.

Combining all service-oriented components of multiple cloud systems into one single piece of software, provides an efficient and practical platform. This approach is suitable not only for helping users decide which PaaS provider better fits current or future requirements in terms of deployment, execution and monitoring, but also for the case that no single provider fulfills these requirements. Intelligent handling of application specification enables fallback to different PaaS providers or smooth deployment to be built directly in to an IaaS cloud solution.

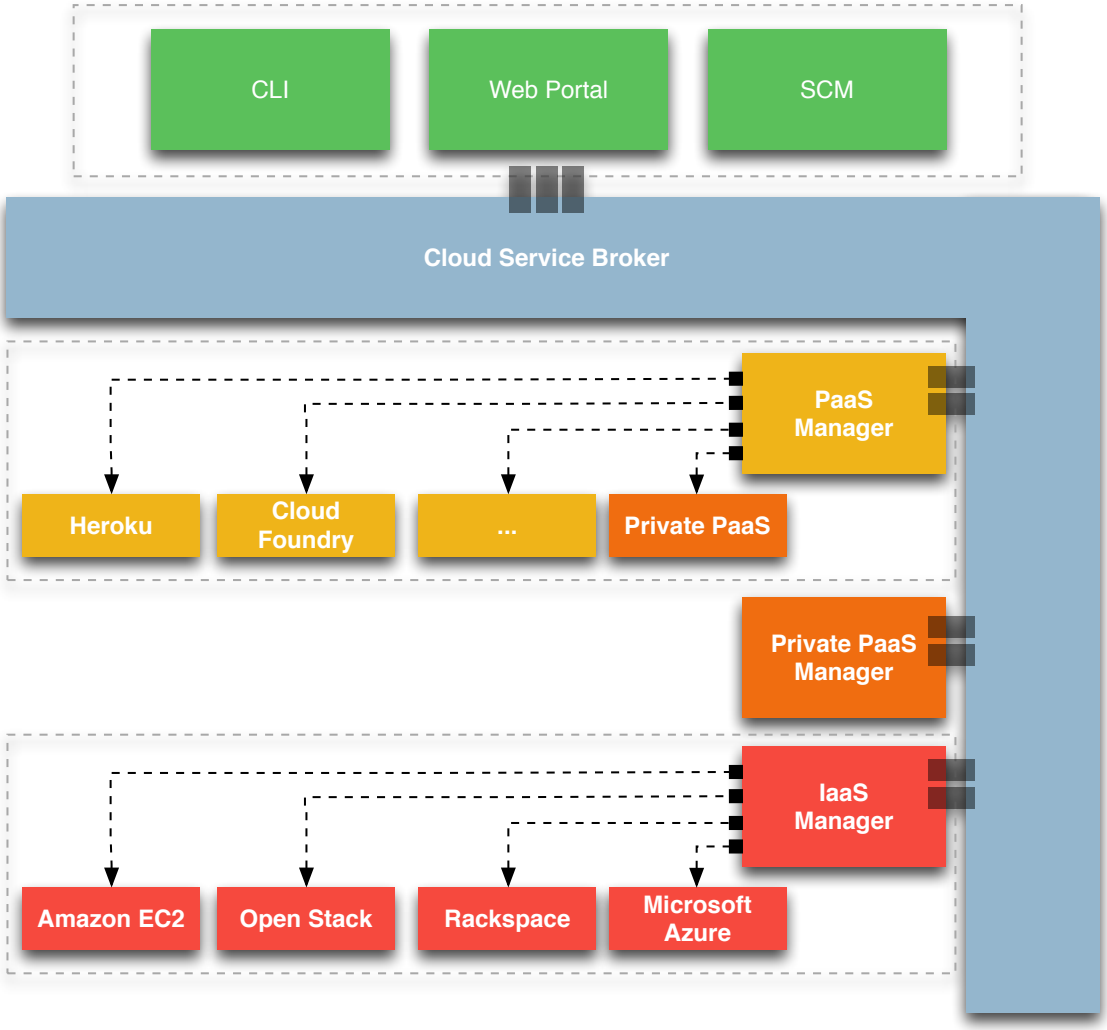

Figure 4.2: Architecture overview

4.2.1 PAAS MANAGER

The PaaS Manager is a framework developed by Portugal Telecom Inovação[1] and Centro ALGORITMI[2] which aggregates several PaaS public offerings based on shared similarities [1][56]. It is intended to provide fundamental features for developers such as creation, management, monitoring, and logging regarding applications and databases. In terms of monitoring, each vendor provides distinct metrics and paradigms. The PaaS Manager also tries to uniform metric name identifiers for equal metrics given by different providers so that users don't have to learn every provider's vocabulary.

The PaaS Manager architecture is entirely modular so each vendor API can be implemented by different entities that abstract the background processes through a Representational State Transfer (RESTful) interface. Figure 4.3 depicts the framework architecture and the four

[1]Portugal Telecom Inovação: `http://www.ptinovacao.pt`
[2]Centro ALGORITMI: `http://algoritmi.uminho.pt`

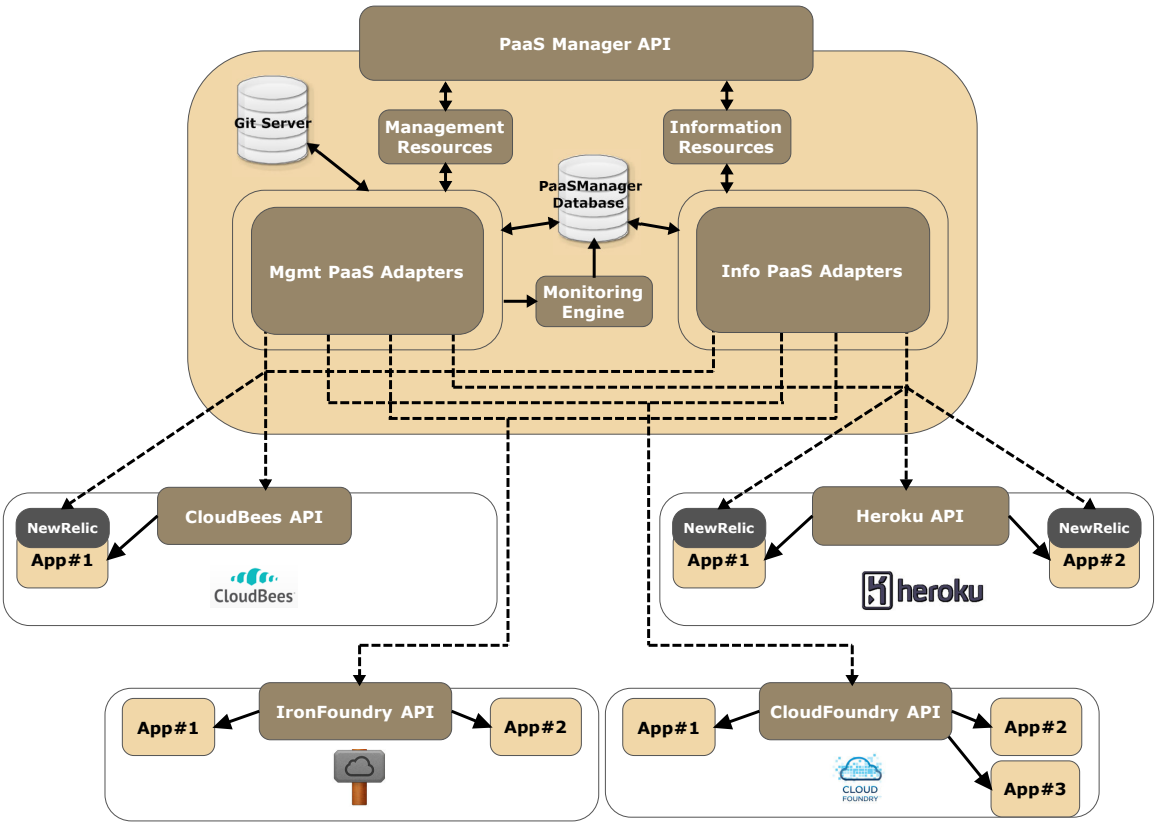

Figure 4.3: PaaS Manager architecture

operational modules:

API: lightweight and RESTful interface supporting all aforementioned operations. The interface can be accessed by authenticated external systems, by including an API key in all the requests. The PaaS Manager on its side mediates the access between users and providers through a unique account. As a result users don't need to register in each vendor for having access to their services

Management Resources: decision module responsible to interact with each PaaS adapter for management tasks (create, deploy, start, stop, migrate, scale, etc). Adapters implement operations related to management features and exposed by the vendors APIs. After they process the acquired information returning unified responses in JSON or XML representations. Other relevant elements are integrated in the Management Resources module and PaaS adapters. A central database keeps state of created applications and a Git server maintains all the applications source code repositories.

Information Resources: module that implements methods related to acquisition of information concerning applications and databases. Here the adapters are responsible for acquiring and processing information related to status, logs and monitoring tasks.

Monitoring Engine: collects real-time metrics disclosed by each provider. A background job is launched and kept alive from the moment the application has been deployed until it is stopped or removed. This process is defined by a synchronous sampling performed every minute. The collected information is stored in the central database and can be queried by users through the PaaS Manager API.

4.2.2 IAAS MANAGER

As mentioned early, relying entirely on the success of the PaaS Manager doesn't cover all cloud customers' needs. A failed attempt to migrate an application to a cloud, fallbacks to the lowest cloud layer and results in the need for carrying out an endless low-level set of system administration tasks: provision the virtual machine, install the operating system, install and tweak user's application software dependencies, install security hardening configurations, add monitoring tools, etc. In the proposed solution, an external entity could help distressing the workload by brokering the relationship between the two worlds. The herein IaaS Manager acts such an external entity (see Figure 4.4).

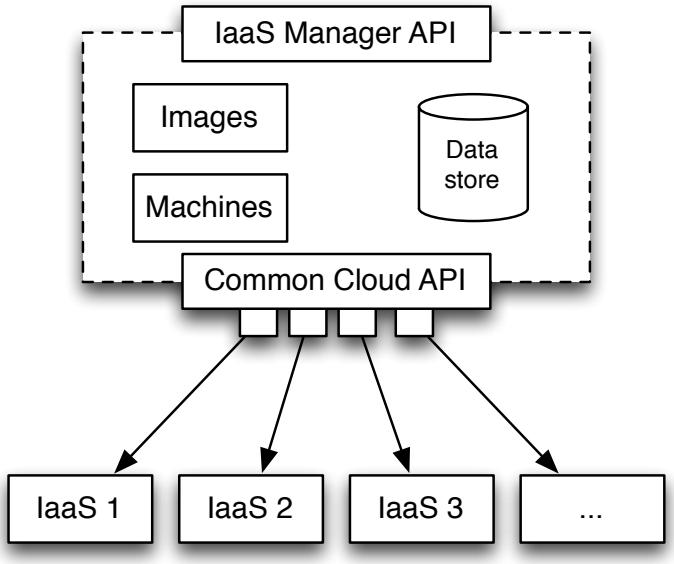

Figure 4.4: IaaS Manager architecture

The IaaS Manager is a cloud virtual machine management software. It leverages multiple clouds to provide a single solution via an abstraction layer between users and different IaaS cloud service providers which greatly improves seamless interaction and interoperability. The platform is designed to support both private and public IaaS clouds. The platform architecture follows an adapter-driven approach. Developing a new driver compliant with a defined interface is what's only needed to offer a new vendor. Internally the adapter speaks the IaaS API

language albeit abstracted to the IaaS Manager by the *Common Cloud API*.

The public API is a RESTful Web API using HTTP and REST principles. *Image* and *Machine* are the minimum required set of resources that should be available. The *Image* resource should provide CRUD operations on virtual images, while the *Machine* resource should provide CRUD and state operations (e.g., start, stop, reboot) on virtual instances. With a virtual image pushed to the IaaS Manager, virtual instances can be launched and started when desired. A database exists to store meta-data on the provisioned machines and uploaded virtual images. The IaaS Manager can be further extended to support advanced networking, storage volumes and storage snapshots features, as well as firewalls, load balancers services.

4.2.3 PRIVATE PAAS MANAGER

Notwithstanding the fact that the IaaS Manager provisions virtual machines to where PaaS will be run it does not carry out any PaaS bootstrap or setup, it simply only manages virtual machines and their computational and network resources. The Private PaaS Manager serves as the piece that administrates such activities. The Private PaaS Manager is a decoupled component of the CSB ecosystem that can be accessed via a RESTful API. The CSB is no more than a user of the Private PaaS Manager. The sequence of which steps are executed when a private PaaS deployment is requested is shown in Figure 4.5 and can be read as:

1. User pushes his application code to the CSB;
2. The CSB requests the IaaS Manager to provision a new virtual machine on a given IaaS provider;
   2.1. The IaaS Manager provisions a virtual machine;
3. A command is sent by the CSB to the Private PaaS Manager to bootstrap a PaaS instance on the just created machine;
   3.1. PaaS configurations are carried on the virtual machine as instructed by the the Private PaaS Manager;
4. The CSB registers the newly created PaaS on the PaaS Manager;
5. The CSB sends all pending applications for deployment and instructs the PaaS Manager to deploy applications to the private PaaS;
   5.1. The PaaS Manager deploys each application on the private PaaS.

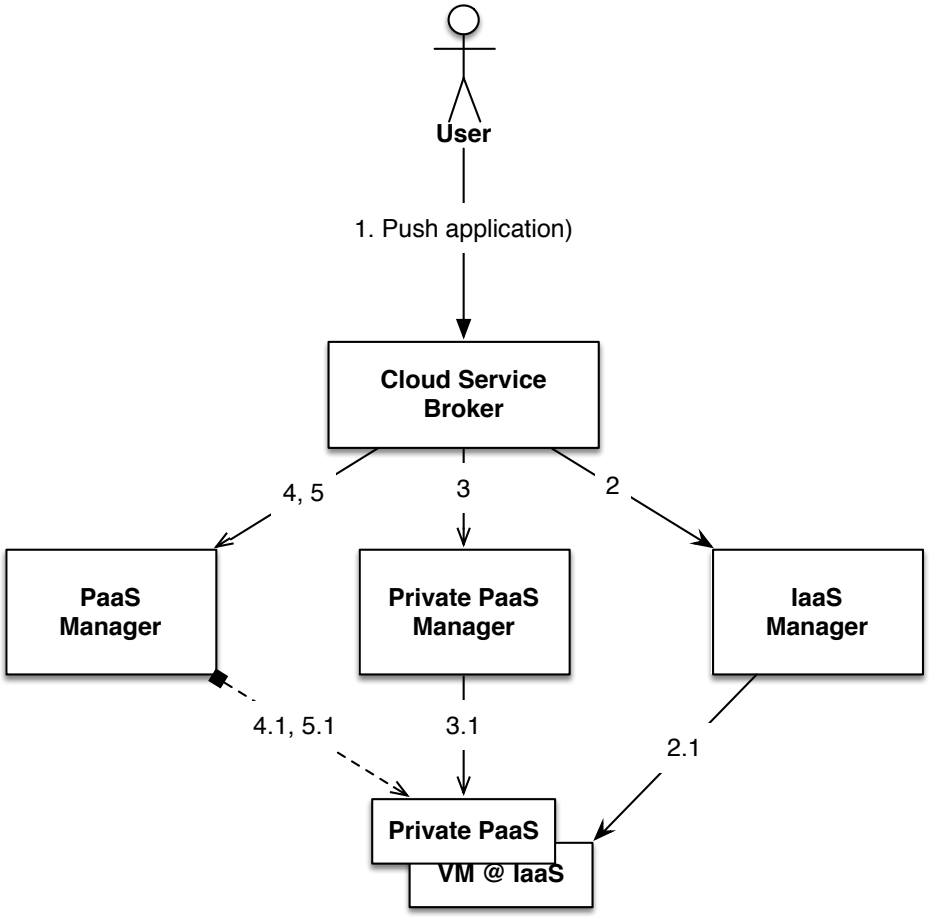

Figure 4.5: Deploying a private PaaS

4.2.4 CLOUD SERVICE BROKER

The CSB is the major component of this architecture. It connects the PaaS Manager, IaaS Manager, Private PaaS Manager and user interfaces through a service bus. While the PaaS Manager, IaaS Manager and the Private PaaS Manager are self-contained, CSB orchestrates, further augments, and eases the process of deploying and running an application in the cloud from the ground up. It also advises users which vendor better matches their application dependencies.

Through the use of the CSB users are qualified to effortlessly migrate their applications from on-premises to the cloud or between cloud providers. Upon user request the CSB scales computational resources on the platform where the application is hosted in order to accommodate high demand periods in a resilient manner. After peak hour or when demand simply drops, applications can be migrated back to its normal state.

Supported functionality can differ significantly between cloud providers. There are PaaS services that do not yield monitoring or logging information. Others while committed to assure no cloud confinement exists it is only possible to interoperate with another provider if

they share the same API which at present implies sharing the exact same platform. The CSB has a database containing data of provider capabilities, including functional, technical and location data. This knowledge is highly important for discerning which characteristics some providers support but others do not and provide means to better advise users with a filtered set of providers matching their application requirements.

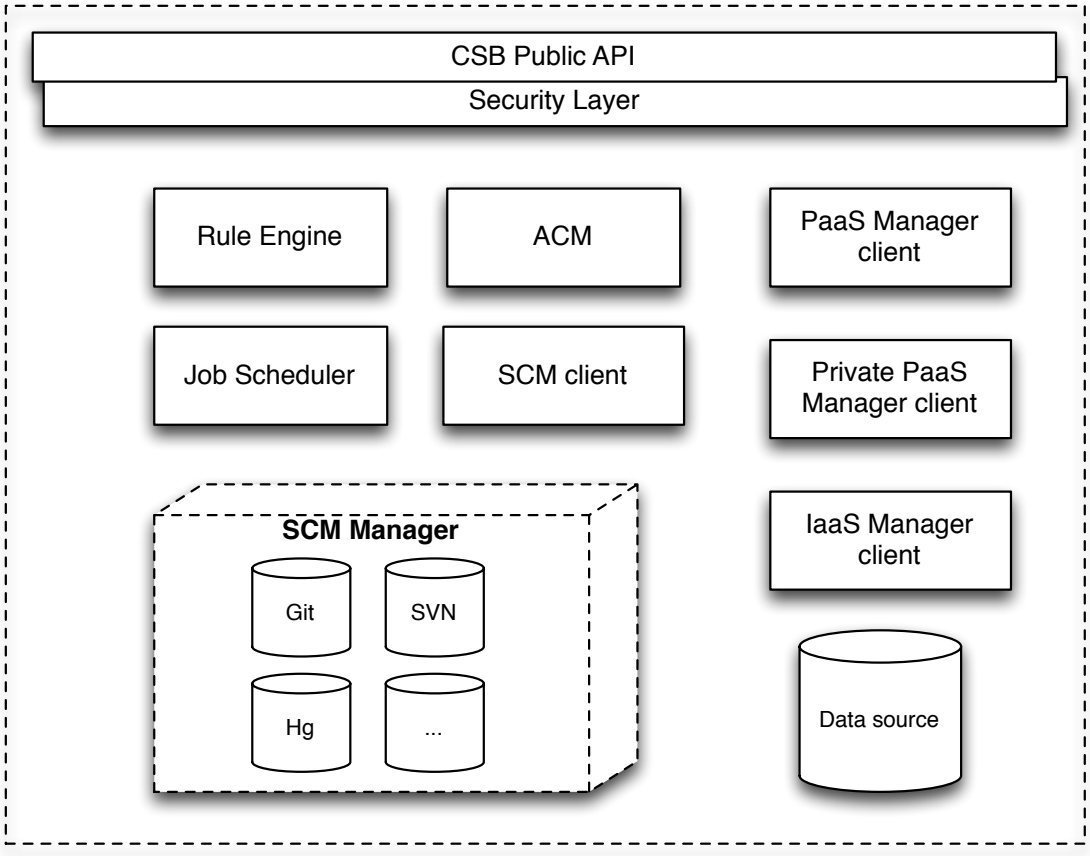

Figure 4.6: Cloud Service Broker components

Figure 4.6 presents several components inherent to the CSB:

CSB Public API: the public API is the users' entry point. The API exposes various services through a RESTful interface;

Security Layer: most web services require user authentication and authorization. This layer ensures user access to API call requests is valid;

Rule Engine: the recommendation of cloud providers to users is handled by a rule engine. A rule engine in this case is a system that produces a list of providers that are in agreement with user's application requirements;

PaaS Manager client: a HTTP client implementation to interact with the PaaS Manager;

Private PaaS Manager client: a HTTP client implementation to communicate with the Private PaaS Manager;

IaaS Manager client: a HTTP client implementation to interact with the IaaS Manager;

Job Scheduler: while bootstrapping a private PaaS instance, users that meanwhile requested application deployments on that same PaaS will be put on hold until the bootstrapping process terminates, in which case subsequently the pending deployments will be triggered;

Application Control Management (ACM): component that controls every aspect related to applications, e.g: state and transition operations, monitoring and logging querying, source code updates, etc;

SCM Manager: manages Source Code Management (SCM) repositories (Git, SVN, Mercurial and others);

SCM client: a client implementation of the SCM Manager RESTful Web Service API;

Data source: stores applications, users, access permissions, and cloud providers information.

CHAPTER 5

Implementation and Results

An implementation of the CSB complying with the specifications and architecture introduced in Chapter 4 was developed. Section 5.1 presents implementation details while on Section 5.2 results from the implemented platform are extracted and validated.

5.1 IMPLEMENTATION

Several different programming languages and frameworks were used on the implementation of the CSB . Such was due to convenience depending on each architectural component presented in Section 4.2. This is, there were tools and libraries written in a specific programming language that were more suitable to be used for implementing a given component than others.

The core of the CSB is a Java Platform, Enterprise Edition (Java EE) application, and depends on frameworks and general libraries such as Hibernate[1], Java API for RESTful Services (JAX-RS)[2], RESTEasy[3], SLF4J[4], Quartz[5], CIMI, and Apache and Google Java libraries. The Rule Engine (see Section 5.1.3) is Prolog and talks with the CSB through the bidirectional Java-Prolog Library (JPL)[6] interface. Prolog code is also wrapped in Java in the CSB thanks to JPL. Likewise, the IaaS Manager is a Java EE application that shares common Java libraries (e.g., RESTEasy, Hibernate). The Private PaaS Manager is on the other side a Sinatra[7] Web application library and domain-specific language written in Ruby[8]. Ruby was

[1]Hibernate: `http://hibernate.org`
[2]Java API for RESTful Web Services: `https://jax-rs-spec.java.net`
[3]RESTEasy: `http://www.jboss.org/resteasy`
[4]SLF4J, a simple logging facade for Java: `http://www.slf4j.org`
[5]Quartz, an open-source job scheduling library: `http://quartz-scheduler.org`
[6]Java-Prolog Library: `http://www.swi-prolog.org/packages/jpl/`
[7]Sinatra: `http://www.sinatrarb.com`
[8]Ruby: `https://www.ruby-lang.org`

also selected as the programming language to implement the Web portal (see Section 5.1.4), using the Ruby on Rails (RoR)[9] framework.

This section is organized as follows. Implementation characteristics of the IaaS Manager and Private PaaS Manager are specified in Section 5.1.1 and Section 5.1.2, respectively. Section 5.1.3 exposes on the implementation of the CSB. Current section ends with Section 5.1.4 discussing the developed user interfaces. The disposition is arranged in a reverse order of dependencies on the entire CSB ecosystem. That is, both IaaS Manager and Private PaaS Manager systems are loosely coupled to the CSB. In fact, they do not depend on the CSB to work as they can be used by third-party systems. On the other hand, the CSB depends upon on these two systems, and on the PaaS Manager, and user interfaces depending on the CSB to work.

5.1.1 IAAS MANAGER

The IaaS Manager is a Java EE Web application wrapped in a Maven[10] project for easy build and dependency management, and distribution. As a Web application, interaction by users with the system is made through a public API that implements the CIMI interface (refer to Section 2.5.1). A standardized cloud interface such as CIMI amplifies the chances of interoperability with current and future third-party entities and tools. Users can perform CRUD and other operations on images and machines as designed in Section 4.2.2 and documented on Section 6.1. Available IaaS clouds include, but are not limited to, Amazon EC2, OpenStack, Eucalyptus, OpenNebula, Rackspace, and GoGrid.

The *Common Cloud API* is leveraged by the multi-cloud Deltacloud tool (Section 3.3). It helps the IaaS Manager cut through the complexity and work with a wide range of IaaS clouds, either private or public ones. Deltacloud offers three APIs (Deltacloud API, DMTF CIMI API and Amazon EC2 API) for communication. Although extensively implemented by several tools, the Amazon EC2 API is proprietary. Deltacloud API is not much used by the community as of this time. Hence, the one chosen for communicating with was CIMI. As mentioned before (Section 2.5.1), CIMI is a standardized API. Having the IaaS Manager speaking CIMI with a multi-cloud tool as Deltacloud in this case, enables future change of the used multi-cloud tool to another CIMI-enabled with no code refactoring needed.

Both public CIMI API provided to consumers and *Common Cloud API* are empowered by the created and open-sourced CIMI Java models library (see Section 1.2). The CIMI Java models are generated in a astute way. Instead of manually creating a Java class for representing each CIMI model, collections and relationships, the library generates them automatically. The process of translating defined CIMI models to Java classes is done through a Maven plugin (*jaxb2-maven-plugin*), that uses JAXB2[11] to generate Java classes from XML schemas. To do

[9]Ruby On Rails: `http://rubyonrails.org`
[10]Maven: `http://maven.apache.org`
[11]Java Architecture for XML Binding: `https://jaxb.java.net`

so, the CIMI XML Schema file is used and included in the project file structure. Although the automatism offered by the plugin, a few tweaks prior to the generation process of the outputted Java classes are required. These tweaks are mostly with respect to renaming CIMI attributes that overlap with Java methods and classes names. All classes were forced to implement the `Serializable` Java class in order to have instantiations of those serialized over the network. This technique enormously reduces the amount of programming and time necessary to get a representation of all CIMI models as Java classes. Succeeding updates on CIMI models can be promptly mapped to new Java classes, as a rebuild is all it takes.

Additionally to the CIMI Java models library, a CIMI client library was developed. The CIMI client library was implemented in Java and, once more, depends on the CIMI Java models. Such client allows third-party Java applications to consume IaaS resources from the IaaS Manager through the CIMI interface. The same client is also used for the communication between the IaaS Manager and Deltacloud. As we will see later on Section 5.1.3, the CSB is another example of an application depending on the CIMI client library.

IaaS Manager RESTful Web Services and the RESTful CIMI Java client library are built using RESTEasy, a JBoss project that is a fully certified JAX-RS implementation. JAX-RS is a specification that provides a Java API for RESTful Web Services over the HTTP protocol. RESTEasy offers tighter integration with the JBoss Application Server, which is used to deploy the IaaS Manager.

5.1.2 PRIVATE PAAS MANAGER

As explained in Section 4.2.3, the Private PaaS Manager is the unit in the CSB ecosystem that bootstraps, configures and manages a PaaS cloud platform on top of a machine, either a physical machine or a virtual machine. Cloud Foundry[12] is the PaaS platform adopted for deploying PaaS instances. Cloud Foundry is an open source PaaS that provides choice of clouds, frameworks and application services. The choice of clouds is expressed by the feasibility of deployment across different deployment models (e.g., public, private and hybrid clouds), and many IaaS clouds (e.g., OpenStack, Amazon AWS, vSphere). Supported frameworks include Spring, Ruby on Rails, Sinatra, Node.js, Grails, Scala, PHP and Python, while dozens of application services (e.g., MySQL, PostgreSQL, Redis, MongoDB, RabbitMQ) are equally supported. Cloud Foundry is backed by a great number of corporations actively contributing to the project, and having deployed and built on top of Cloud Foundry commercial services. To name a few, ActiveState, AppFog, eBay, IBM, Intel, Tier 3 and CloudSoft, are examples of corporations contributing to the open source platform.

An implementation of the designed Private PaaS Manager was written in the Ruby language and is RESTful. Ruby was elected as the programming language to use due to the existence of dependency libraries also written in Ruby. The Private PaaS Manager platform

[12]Cloud Foundry: `http://cloudfoundry.com`

takes advantage of Sinatra, a Web application library and Domain-Specific Language (DSL) also written in Ruby. Sinatra is a small, light and fast framework that reduces the dependency footprint to just a couple of software libraries at the cost of not following a Model-View-Controller (MVC) pattern, nor having support for a object-relational mapping database wrapper or scaffolding like other frameworks (e.g., RoR). Rather, it is best to be used on simple Web applications and, in the opinion of the author, on REST applications as it extensively uses the Route pattern. Routes are HTTP methods paired with a Uniform Resource Locator (URL)-matching pattern and following the principle of REST as a Web API design model. API authentication is implemented through HTTP Basic Access Authentication [57] to allow only authorized consumers. This access authentication scheme is not considered a secure method of user authentication as user name and password fly over the network unencrypted, unless used in conjunction with RFC 6176 [58].

Various Web services are implemented and documentation is available in Section 6.1. The Private PaaS Manager can perform actions such as setup of a new PaaS instance, start, stop and restart, discover and inspect the health the deployed components that constitute a running PaaS environment, and retrieve a collection of instantiated PaaSs. Documentation of REST API is available in Section 6.1. Setup, start, stop and restart Web services are actions that take some time to be completed in the context of a single HTTP request. Because of that, when invoking those services a `HTTP 202 Accepted` status is returned to the client. The `HTTP 202 Accepted` means that the server has received and accepted the response for processing, but the processing has not yet been completed. The Private PaaS Manager creates and schedules a job service that is completed on background. Each PaaS has its own job queue so that if multiple jobs are requested before a previous one gets completed, they will be queued and processed one at a time in the same order they were received. An example that demonstrates the need for such feature is a setup request followed by a start request. In this case, starting the PaaS will only be executed after finishing processing the setup job. Resque[13], a Redis[14]-backed Ruby library for creating background jobs, backs the job service system. The setup job has the particularity from the other jobs of accepting and storing a previous passed in URL endpoint when the request was received. The Private PaaS Manager invokes a `HTTP POST` request on the URL endpoint that can be used to signal the callee that the setup has just been completed. Furthermore as we shall see in Section 5.1.3, the CSB makes use of the callback to be informed that the PaaS setup has finished and therefore the new PaaS instance can be surfaced to upper layers such as the Web portal or other user interfaces.

At the core of the management component lays the *cloudfoundry-manager* RubyGem, a Ruby client library developed for bootstrapping and managing CloudFoundry clouds. This library is responsible for performing operations as the ones solicited by the Private PaaS Manager Web services on Cloud Foundry instances. During the bootstrapping stage, the

[13]Resque: `https://github.com/resque/resque`

[14]Redis, an open-source and in-memory key-value data store: `http://redis.io`

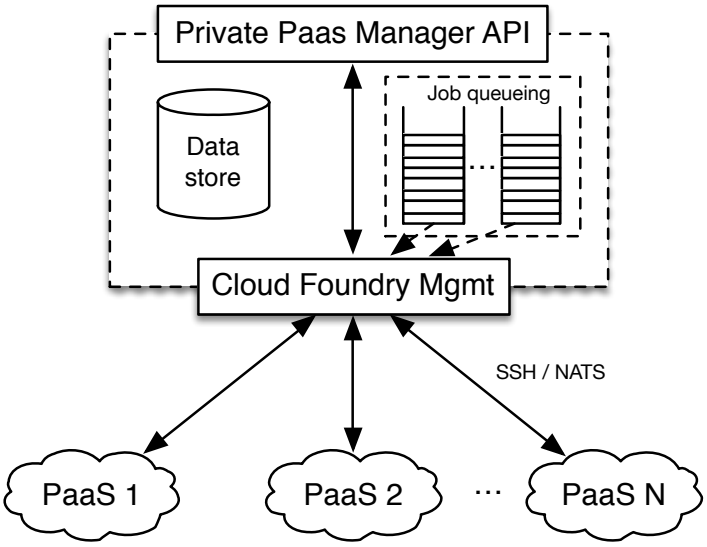

Figure 5.1: Structural elements of the Private PaaS Manager

cloudfoundry-manager establishes a Secure Shell (SSH) connection to the targeted machine (IP address, user name and password are passed in). The client library configures a Cloud Foundry instance according to the domain name previously passed in as parameter, too. Information of the just instantiated platform is persisted in a data store, a YAML Ain't Markup Language (YAML) file. An SSH connection is also used to start, stop and restart PaaS instances. Any other communications to the instances are via NATS[15]. NATS is intensively used by the Cloud Foundry platform for internal components communication. *Cloudfoundry-manager* depends on NATS Ruby client library for relaying requests such as discovery of Cloud Foundry components and retrieving their information.

Along with the Private PaaS Manager server component, a client library was developed. The implemented REST client is a Java library wrapped as a Maven project. The interrelationship between the Private PaaS Manager and the CSB, or any other Java consumer application, can easily be handled using this client library.

5.1.3 CLOUD SERVICE BROKER

The implemented CSB is a SOA application mostly written in Java EE. The CSB is formed by various components that make the whole application. These components when depending on external components communicate through a REST API, and thereupon REST client libraries are used and, when needed, were developed from scratch. This section elaborates on the implementation details of the CSB. A formalization of the Manifest structure is

[15]NATS, a lightweight publish-subscribe and distributed queueing messaging system: `https://github.com/derekcollison/nats`

presented as well a Rule Engine for assessment of the Manifest. The Web API and its security measures to allow only authentication and authorized users to consume REST services are explained, and details on the implemented PaaS services are also expounded. This section ends reporting on how the CSB is prepared and encompasses the provisioning a private PaaS on multi-infrastructure cloud computing platforms.

WEB API

The CSB is a SOA application through which users can access cloud services through a REST API. Data on HTTP requests can be serialized as JSON or XML and should be specified on the HTTP `Content-Type` header. Response serialization format can also be chosen by users, either appending the HTTP `Accept` header (`Accept: application/json` or `Accept: application/xml`) or suffixing the URL with `.json` or `.xml`. The API is a fully compliant JAX-RS REST API built using the RESTEasy framework.

The REST API is safeguarded by a security layer illustrated in Figure 4.6. This layer intercepts API service calls over the Web API for authentication and authorization access rights beforehand processing the requested and protected services. The CSB enforces two authentication mechanisms, HTTP Basic Authentication [57] and OAuth 1.0a [59]. Both authentication mechanisms were implemented for different reasons, and users can opt to use one of the two on each request. When using HTTP Basic authentication, users provide their user name and password. This is the simplest technique, though credentials are encoded in BASE64 and are insecurely transmitted if over a non-secure HTTP; a HTTPS connection can be established to secure all information flowing in the channel. In the other available authentication mechanism, OAuth, users grant access to their protected resources without sharing their credentials with the consumer. OAuth was the protocol arbitrated to be implemented also because it is an open and Internet Engineering Task Force (IETF) standardized protocol, provides both authentication and authorization security checks.

Figure 5.2 shows the interactions between a user, consumer and the service, both to authenticate the consumer as an entity who as access to user's data, and to authorize it to consume data from the service on behalf of the user. In a OAuth authentication process the Consumer obtains an unauthorized request token from the Service. The Service entity is the CSB platform. The User is redirected to an authentication page in the CSB and gives authorization on the request token. The Consumer exchanges the request token, now authorized by the User, for an access token. After successfully interchanging the access token and token secret, the Consumer is now able to access protected resources on behalf of the User.

Authorization to protected resources is empowered by a role policy mechanism. Roles were introduced to differentiate users so that for a set of users belonging to a given role in the CSB could allow or forbid access to Web service resources. For instance, the Web portal (Section 5.1.4) has a user account, which belongs to the Administrator role. Only users

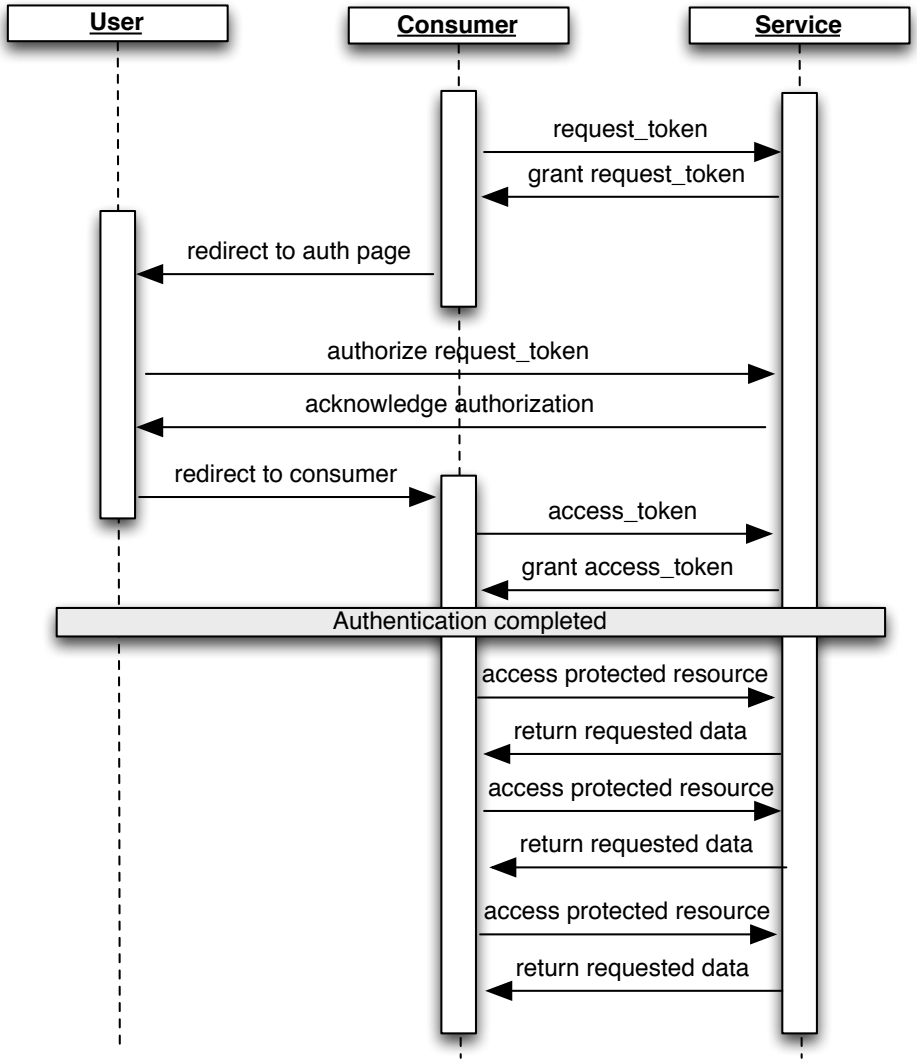

Figure 5.2: OAuth messaging exchange

enrolled in the Administrator role have access to invoke more restricted resources, such as user sign up directly through the REST API. Users in the default User role are unable to create user accounts straightforwardly. This role information data is stored in a database. The physical data model of the database is shown in Figure 5.3 (`PK` stands for Primary Key, `AK` for Alternative Key and `FK` for Foreign Key).

Tables `consumers`, `request_tokens` and `access_tokens` relate to OAuth. Table `consumers` contains registered applications information. A registered OAuth application is assigned a unique key identifier and secret. The secret should not be shared. Table `request_tokens` stores yet unauthorized access grants. The `callback` is optional and, if provided, users will be redirect to the containing URL. The redirect URL's host and port must exactly match the `connect_uri` in the `consumers` table. Once the User gives Consumer permission, the requested record is deleted. Granted accesses are stored in the `access_tokens` table. Users information is saved in the `users` table. User passwords are first hashed with salt

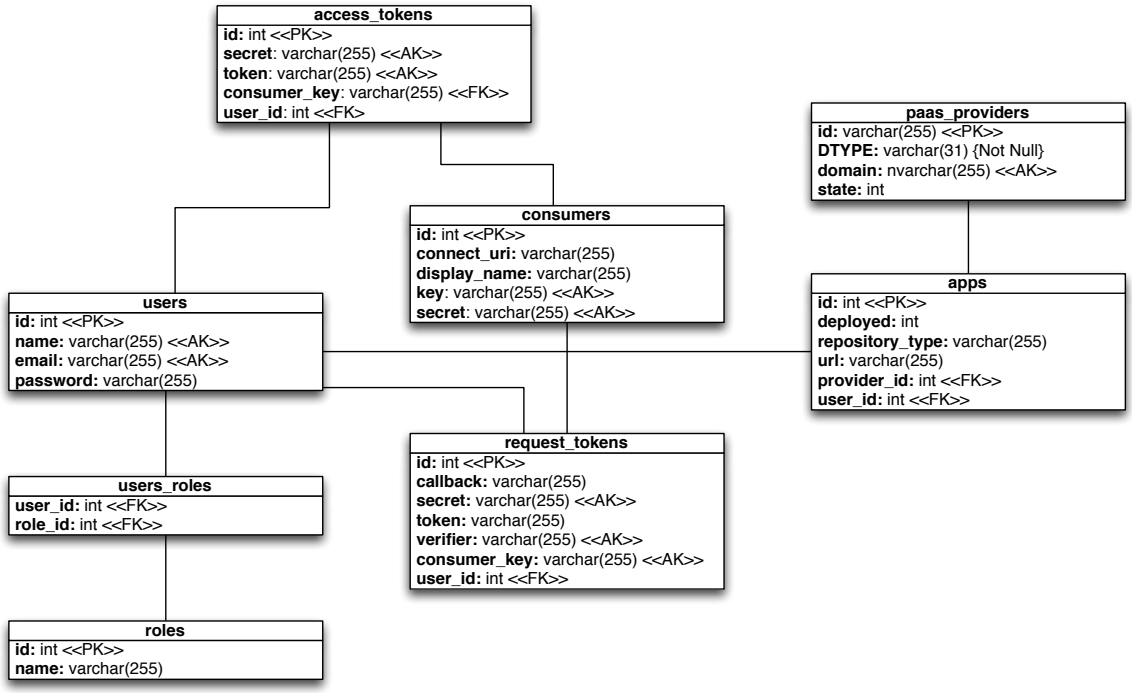

Figure 5.3: Database model

using SHA-256 [60] and before storing in the database. Table `apps` and table `paas_providers` are related to user applications and PaaS providers, respectively. Later in this document they will be explained. The CSB depends on Hibernate[16], an Object-Relational Mapping (ORM) library for Java that implements the Java Persistence API (JPA) interface[17], for mapping Java objects to a relational database, MySQL in this case.

MANIFEST AND RULE ENGINE

The manifest can either be a JSON or XML formatted file document. It comprises two base elements: `rules` and `provider`. `Rules` is a collection of a variable number of `rule` elements. Each rule is composed by its identifier and parameters, if any. The `provider` element is an optional element. It can be specified in case the user knows beforehand in which PaaS platform the application should be deployed to. Else, it should be left unspecified. Listing 9 formally describes the elements of the Manifest document in a XML schema document. This schema is an abstract representation of the Manifest's elements and relationships.

Recommendation of CSPs to users in compliance with the manifest document is handled by the Rule Engine. The Rule Engine in this case is a system that produces a list of providers that are in agreement with user's application requirements and thus is well matched to be deployed on any of those clouds. The developed solution to implement a rule engine was backed

[16]Hibernate: `http://www.hibernate.org`

[17]JSR 317: JavaPersistence API, Version 2.0: `http://jcp.org/aboutJava/communityprocess/final/jsr317/index.html`

by Prolog, a declarative programming language associated with artificial intelligence and computational linguistics. Prolog programs are instructions that can almost always be read as logical statements. Results of a computation of a Prolog program is a logical consequence of the axioms in it. A Prolog program consists on facts and rules which serve to define relations on sets of values. Facts are predicate expressions that make a declarative statement about the problem domain; rules are predicate expressions that use logical implication to describe a relationship among facts.

The Rule Engine uses the SWI-Prolog[18], an open source implementation of Prolog. SWI-Prolog was chosen over other Prolog implementations (e.g., BProlog, GNU Prolog, Open Prolog and Visual Prolog) for several reasons. It's open source under the LGPL license, runs on multiple operating systems (e.g., Unix, Linux, Windows and Mac OS X) and is compliance with both ISO-Prolog and Edinburgh Prolog dialects. Another reason for choosing SWI-Prolog was that it's under active development and offers a Java interface, called JPL. JPL is a bi-directional Java Prolog bridge available in SWI-Prolog that allows Java and Prolog to call each other. It is a dependency of the CSB because the CSB needs to inject Java objects unmarshalled from XML or JSON formatted Manifest's rules to Prolog. JPL hence translates Java objects to Prolog data types, process the rules and returns back results reversing from Prolog data types to Java objects.

Facts in the CSB context represent data information including runtimes, frameworks, services, metrics, PaaS providers and what they have to offer. Those facts can be extracted automatically autonomously from a list of all PaaS offerings and are stored in a knowledge database file. The Rule Engine loads and consults the knowledge database when processing rules. Rules are expressions that using facts produce a list of matching PaaS offerings. Rules take at least two parameters: (1) a list of input providers to evaluate and (2) a list of providers as output that are logically in accordance with the rule. More parameters can be passed in if the rule accepts optional parameters. For example, to check if a PaaS cloud supports a specific runtime it is necessary to also pass in the identifier of the runtime.

Currently the Rule Engine supports four kinds of rules: `rule_runtime`, `rule_framework`, `rule_service` and `rule_metric`. With the exception of the `rule_metric` rule, the three other kind of rules have two variants. One variant only assesses if the runtime, framework or service is supported disregarding any version information. The second variant is more flexible as it permits specifying a comparison operator and version. These two extra parameters allow users to stipulate a minimum, equal or maximum of a given version of a runtime, framework or service compatible with the user's application. The available operators are: `less`, `less-equal`, `equal`, `greater` and `greater-equal`.

Invoking `POST /manifest` Web service in the Manifest REST API (refer to Section 6.1) and submitting a Manifest document in either one of the two supported file formats, XML and JSON, will trigger the Rule Engine. The Rule Engine runs the rule chain until the complete chain ends or until a rule producing an empty PaaS providers list is hit. The outputted

[18]SWI-Prolog: `http://www.swi-prolog.org`

list of a rule serves as input list for the next rule. If a rule produces an empty list of PaaS providers, the list of the previous ran rule is returned. The order by which rules are evaluated is irrelevant, following a everything or nothing at all strategy. Providers not fully complying with the Manifest should be discarded. It is up to users to decide if they accept any of the recommended clouds and which one they are more interested in. Refer to Section 6.1 for an example of a Manifest document (see Listing 10) and the outputted list of PaaS providers returned by the Rule Engine (see Listing 11) in XML.

The operator of the platform can easily add new rules to the Rule Engine. News rules are added to the Prolog file containing existing rules. The CSB platform will automatically make available those changes to the entire CSB ecosystem with no further necessary changes.

PAAS SERVICES

The CSB brokers PaaS services between users PaaS CSPs, and adds value-added services on top of the intermediation service layer. The implemented CSB endows cloud consumers with a ample domain of application services. Consumers can create applications, deploy them on a supported PaaS platform, perform state switching operations such as start, stop and restart. Consumers can also scale instances, migrate applications from one provider to another, create and associate services to their applications, and retrieve logs and monitoring data. All these services are more are accessible via the CSB REST API. A full list and documentation of these services can be found in Section 6.1. Inner to the REST API lays an infrastructure for cooperation between various entities internal to the CSB ecosystem and external as is the case of PaaS CSPs of which users are abstracted from. Internal entities include the ACM module, the SCM Manager, and the PaaS Manager. A PaaS Manager models and client library (illustrated in Figure 5.4 as *PM client*) had to be developed so that the CSB could communicate with the PaaS Manager platform.

The SCM Manager[19] is a SOA application for managing Git, Subversion, Mercurial and other repositories. It features a RESTful Web service API, a central user, group and permission management, and a plugin framework accompanied by a plugin API. Thanks to the SCM Manager, users are given the option to choose which SCM system they want to use. The CSB is prescind from the intrinsic differences of the miscellaneous SCM system types. The *CSB deploy* plugin was developed and added to the SCM Manager plugin system. This plugin extends the `PostReceiveRepositoryHook` repository hook that is an extension point of the plugin framework, and it is called every time a changeset is received and carries out programmed procedures. In the context of the CSB, the *CSB deploy* plugin is programmed to signal the CSB platform that changes to the application were received and it should redeploy the application. The plugin invokes the `PUT /apps/:appId/deploy` REST service (see Section 6.1).

The ACM module wraps all SCM procedures from the application point of view, controlling

[19]SCM-Manager: `http://www.scm-manager.org`

and further extending operations provided by the SCM Manager. For instance, the ACM module retrieves application's source code from the SCM Manager and compresses it in a single file as the PaaS Manager requires applications to be deployed in a ZIP archive file format. The ACM module depends on the SCM client library to consume resources from the SCM Manager.

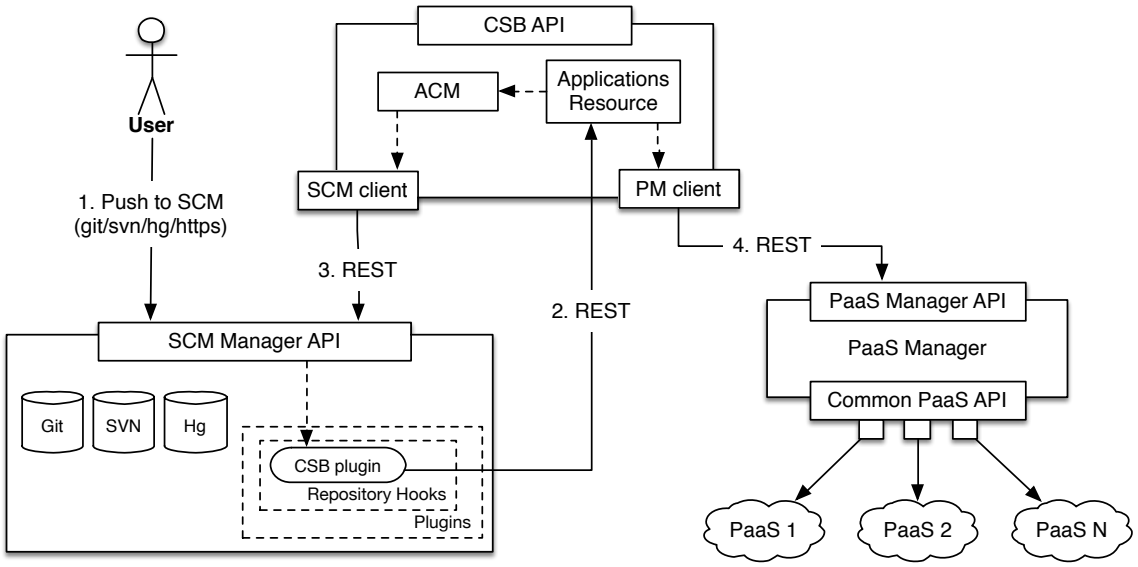

Figure 5.4: Activity diagram showing the flow of an application deployment

Users interact with the CSB via either the CSB API itself or the SCM system when pushing code to the their application's repository. The latter will automatically trigger a new deployment of the application on the targeted PaaS cloud. Figure 5.4 shows an activity diagram representing the flow of an application deployment. For simplicity of understanding the flow, only the essential modules and entities involved were depicted. The flow comprises four steps between the entities presented in the diagram (User, SCM Manager, CSB and PaaS Manager). User pushes his code to the remote application repository provided by the CSB when it first created the application on the platform. In step 1, the SCM Manager receives the code and stores it in the corresponding SCM repository. The SCM Manager fires off the *CSB deploy* plugin of which will inform the CSB that a redeployment should occur (step 2). The CSB receives the request, ask the ACM module to retrieve the newest application source code and create a ZIP archive file from blob data received from the SCM Manager (step 3). In step 4, the CSB requests the PaaS Manager to deploy a new version of the application, sending in the HTTP request body the ZIP file.

PRIVATE PLATFORM AS A SERVICE

The Private PaaS is a unique service of which the CSB provisions itself a PaaS cloud using a PaaS software platform on top of most known and used IaaS clouds such as Amazon AWS,

Rackspace, GoGrid, OpenStack and many others. Therefore the CSB can be seen as a PaaS CSP provider, competing with remaining PaaS CSPs. The major depending mechanisms employed by the CSB to setup and offer a PaaS service of its own were previously described. In Section 5.1.1, an IaaS Manager was implemented for provisioning virtual machines on multiple IaaS CSPs, and in Section 5.1.2 a system where a Cloud Foundry software instance is setup atop of a virtual machine. So far these two components do not minister the necessary means to give CSB users the faculty of choosing these private PaaSs as they are not exposed in the PaaS offerings list available through the CSB API. Consequently it's up to the CSB to glue all the pieces together to close the PaaS provisioning cycle. It is noteworthy that private PaaS instances are only provisioned when users request them, this is, on a lazy initialization design pattern. Following a lazy initialization, the CSB delays the creation of the PaaS cloud until at least one user requires his application to be deployed on the requested private PaaS. This strategy introduces an adverse point when contrasting with an eager initialization; users requesting application deployments on yet unprovisioned private PaaSs will have to be put on hold until the cloud environment is assembled. Although, the *delay until needed* approach enables the operator of the CSB to save money as it will only started being charged for the virtual machine on a IaaS CSP once it's up and running. Next it will be detailed how the CSB takes advantage of the services provided by the IaaS Manager, the Private PaaS Manager and the PaaS Manager to offer additional PaaS clouds to their users. The explanation can be accompanied with Figure 5.5.

Based on the principle that the CSB knows the features provided by Cloud Foundry (e.g., runtimes, frameworks, services and metrics), it can append to the PaaS offerings list this offering times the number of supported IaaS clouds. Users can opt to choose one of these private PaaS offerings rather than public ones. The Manifest should reflect such preference when pushing the application source code to the SCM repository. The CSB application deployment service is triggered, as early discussed in Section 5.1.3. This is now the part of which the CSB introduces supplementary steps towards provisioning the application on a private platform. The CSB checks the PaaS and notices it is targeted for a private cloud that has not yet been instantiated. The CSB sets the deploying state of the user application to `PENDING` in the database table `apps` and schedules two cascaded service jobs in the Quartz job scheduler. The jobs will run in the background while the CSB returns an `HTTP 202 Accepted`. The first one job, *MachineJob*, is triggered off right away and will create a virtual machine on the IaaS provider correspondent to the private PaaS selected backing on the IaaS Manager (refer to Section 5.1.1) via the developed CIMI client library. Once the virtual machine is created and running the SSH server, the *MachineJob* triggers off the second job, *PaasBootstrapJob*. This job will request the Private PaaS Manager to setup Cloud Foundry on the just created machine. For more information on the setup process, consult Section 5.1.2. Once the setup is done the Private PaaS Manager will callback the CSB `POST /ppm/register` service informing the broker that the setup process is completed. The CSB registers the new PaaS on the PaaS Manager and deploys all pending applications.

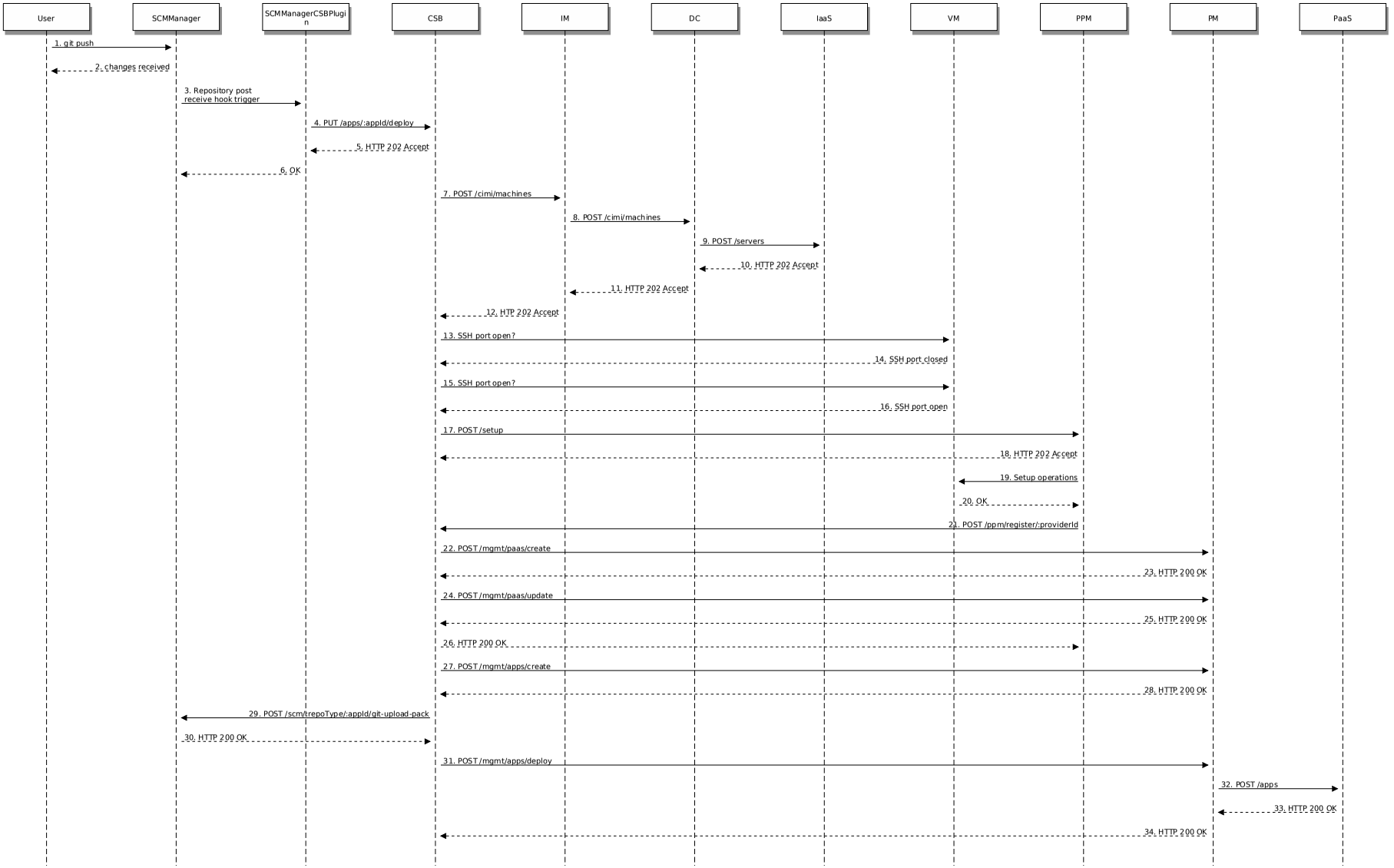

Figure 5.5: Sequence diagram of an application deployment on a provisioned private PaaS.

5.1.4 USER INTERFACES

An user interface is a program that sits above an information system with which a human being interacts with. The goal is to ease the communication between users and a computer program, otherwise a powerful program will have little value to users if the user interface is non-existent or poorly designed. A type of user interface are client libraries that consist on a collection of code that eases the job of programmers with an abstraction layer to a system. It takes many of the concepts of a system and makes them accessible via code. Another type of interface can be a set of commands where users issue commands to the program in the form of text lines, commonly known as Command Line Interface (CLI). Yet another interface is the graphical interface. In a graphical interface users are allowed to interact with a program through manipulation of graphical elements, selecting commands from various menus and icons displayed on the screen. CSB features three user interfaces and of different types developed and available to users.

CSB GEM

CSB gem[20] is a Ruby client library to interact with the CSB's REST API from a Ruby application. CSB gem wraps the CSB API in an API client that follows Ruby conventions and requires almost no knowledge of REST. Methods have positional arguments for required input and an options hash for optional parameters, headers or other options.

The gem depends on Faraday[21] as the HTTP client library and Rack[22] middleware for processing requests to and responses from the CSB API, and provides full support for HTTP over SSL connections to the CSB API. HTTP errors returned from the API are mapped to custom Ruby error classes for easy rescuing from CSB errors. Response bodies received in JSON or XML data format types are parsed to Hash or Array data structure objects.

CSB API requires third-party applications to authenticate via OAuth to consume resources on behalf of users. Applications should register with the CSB in order to get assigned with a consumer key and consumer secret. For authentication, another gem called *omniauth-csb* was created as a *strategy* for *OmniAuth*[23]. *OmniAuth* is a Ruby authentication framework aimed to abstract various types of authentication providers and is widely used by the Ruby community.

WEB PORTAL

The Web Portal is a graphical interface that serves as an entry point on the Internet for users to create and manage cloud computing services from the CSB. A Web graphical

[20]Gems are Ruby programs and libraries distributed on a self-contained format. The RubyGems is a package manager software that allows gems to be downloaded, installed and used on a system.

[21]Faraday: `https://github.com/lostisland/faraday`

[22]Rack: `http://rack.github.io`

[23]OmniAuth: `http://intridea.github.io/omniauth`

interface leverages users with the aptness to access the portal anytime and anywhere from a device connected to the Internet. As with any SaaS cloud application, the portal can be deployed also anywhere and have multiple instances running concurrently. Delivery of updates is accelerated and executed by the operator of the interface as the software is hosted centrally on the cloud, contrary to software installed on user's computer.

The Web interface was developed in Ruby using the Web application framework RoR[24] and Bootstrap[25] for designing the front-end. The portal proxy calls to the CSB platform between users and the CSB REST services acting as a third-party party application consuming resources from CSB services and producing new value-added services to its users. Because of that, the portal doesn't require storing any persistent data. All information related to cloud services available by the CSB are retrieved purely from the CSB API using exclusively the `csb-ruby` Ruby library to communicate with the core of the CSB platform through its API.

COMMAND-LINE INTERFACE

The CSB command-line interface is a unified tool to manage CSB services. The CLI provides a Java language binding for the CSB REST API models. Listing 1 shows the command line parameters available on the CSB Java CLI.

```
$ java -jar csb-client-0.0.1-SNAPSHOT-jar-with-dependencies.jar
[2013-10-21 23:53:56,560] Target server is set to URI
    http://10.115.1.19:8080/csb/rest
---
usage: java -jar csb_client-0.0.1-SNAPSHOT-jar-with-dependencies.jar
---
Options:
 -ca,--create-app <arg>     create an app
 -da,--deploy-app <arg>     deploy an app
 -dla,--delete-app <arg>    delete an app
 -h,--help                  print this message
 -lp,--list-paas            list all available PaaS providers
 -m,--manifest <arg>        best matching cloud providers given a manifest
 -ra,--restart-app <arg>    restart an app
 -sa,--start-app <arg>      start an app
 -sca,--scale-app <arg>     scale app instances
 -spa,--stop-app <arg>      stop an app
 -sta,--status-app <arg>    get an app status
```

Listing 1: CSB Java CLI parameters

As the client application is in preliminary stage, it uses the HTTP Basic Authentication [57] mechanism to authenticate users on the CSB API, although it can be changed to support OAuth. With that purpose in mind, *Scribe-Java*[26] OAuth library for Java was extended to support CSB API OAuth 1.0a, and an example of a generic CSB client is available.

[24]Ruby on Rails: `http://rubyonrails.org`
[25]Bootstrap: `http://getbootstrap.com`
[26]Scribe-Java: `https://github.com/fernandezpablo85/scribe-java`

5.2 RESULTS

5.2.1 TEST BED DESCRIPTION

To validate the herein presented work, an IaaS cloud was setup using the OpenStack platform. The deployed architecture is based on multiple nodes. Nodes run the operating system Linux Ubuntu 12.04 LTS x84_64 and the OpenStack *Folsom* release. Systems were updated to the latest security and bug fixes available at the time of setup. Servers specifications were:

Control node: Asus RS120-E5/PA4 rack server (Intel Xeon 2.66GHz dual-core CPU, 4GB Random Access Memory (RAM) ECC, RAID-5 storage)

Compute node: HP ProLiant BL460c G7 blade server (24 X5675 3.07GHz CPU cores, 192GB RAM ECC, RAID-5 storage)

Network node: Generic PC Desktop (AMD Athlon 64bit 3000+, 2GB RAM ECC)

Storage node: NetApp storage system

The private cloud hosted a virtual instance on the compute node where the CSB was run at. It ran Ubuntu 12.04 LTS x84_64 on an Intel Core2Duo T7700 2.40GHz CPU, processor of which selected by the KVM hypervisor, and 2GB of RAM. Software dependencies were also deployed on the same virtual instance. Those dependencies include the PaaS Manager and IaaS Manager Web application ARchive (WAR) files deployed to a JBoss Application Server version 7.1.1 on an OpenJDK 6 environment, the Private PaaS Manager on Ruby version 1.9.3, MySQL version 5.1.67, SCM-Manager version 1.28 and DeltaCloud version 1.1.3.

5.2.2 RECOMMENDATION BASED ON APPLICATION TECHNOLOGY

In the scope of the CSB, all applications must be accompanied with a manifest describing minimum requirements for recommendation of and execution on PaaS clouds. The CSB offers users four public PaaS cloud providers (Cloudbees, Heroku, Cloud Foundry and Iron Foundry) and three private PaaS solutions built on top of IaaS cloud platforms (OpenStack, Rackspace and Amazon AWS). In this test the Rule Engine is analyzed as well as the processing time from the different components involved.

The submitted Manifest for assessment contains six rules (see Listing 10) and can be summarized as an Ruby application depending on a version of Ruby equal or prior to version 1.9.3 and depending on the Ruby on Rails framework of a version equal or later to version 3.0. The application requires a PostgreSQL version 9.1 database. The developer wants to be able to monitor, at least, the responsiveness time and usage of the CPU on the PaaS platform. The CSB receives the Manifest document and runs it through the Rule Engine

(see Section 5.1.3). The result of the decision-making assessment results in a list of PaaS CSPs matching the user's requirements. In this case scenario, Heroku is the only CSP fully matching all requirements (refer to Listing 11).

Figure 5.6 illustrates which entities are involved and how they operate with one another arranged in time sequence. The sequence starts when a user invokes the REST API Manifest service and submits a Manifest document, either in XML or JSON data format, to the CSB. After the CSB receives the request, it interacts with the PaaS Manager to get a list of PaaS CSPs and their offerings. The CSB invokes the Rule Engine to assess possible matches between the Manifest elements and the list of PaaS offerings. The result is a list of PaaS providers containing no elements, a single or multiple ones.

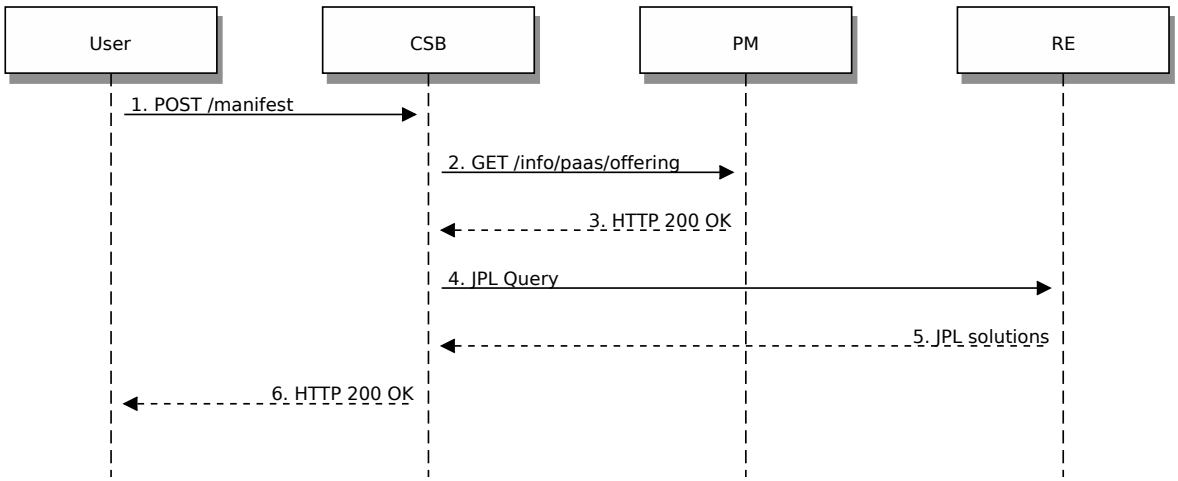

Figure 5.6: Recommended cloud providers

To evaluate the responsiveness of the implemented system sequential HTTP requests were sent from the user to the CSB API. Metric data were recorded in three locations:

- **User:** sent from a PC connected one hop away from the destination network. Apache JMeter[27] 2.9 was used to load test functional behavior and measure performance
- **CSB:** metric data recorded by the time requests arrive to the requested API after authentication and authorization
- **PaaS Manager:** time before network packets are created and sent from the CSB

A JMeter test plan was created to send 1000 HTTP requests towards the Manifest RESTful API service. The HTTP headers of a HTTP request and response are shown in Listing 2 and Listing 3, respectively. The HTTP request body can be found in Listing 10 and the HTTP response body in Listing 11.

[27]http://jmeter.apache.org

```
POST /csb/rest/manifest HTTP/1.1
Connection: keep-alive
Authorization: Basic Y2dvbmNhbHZlczpxd2VydHkxMjM=
Content-Type: application/xml
Content-Length: 717
Host: 10.115.1.19:8080
User-Agent: Apache-HttpClient/4.2.3 (java 1.5)
```

Listing 2: HTTP headers of a request to the Manifest API service

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/xml
Content-Length: 2117
Date: Mon, 21 Oct 2013 13:23:53 GMT
```

Listing 3: HTTP headers of a response from a call made to the Manifest API service

Figure 5.7 plots the response times of the sequential requests. The CSB time is the time elapsed in the CSB and includes the time also elapsed in the Rule Engine. The total time is the sum of the elapsed time in the CSB and in the PaaS Manager. An early analysis reveals that most of the time processing the user request is spent by the PaaS Manager as the difference between the total time and the CSB time represents the time elapsed by such component. In Section 5.2.2 the average total time is the sum of the PaaS Manager time, CSB time and network latency (average of 6.205 milliseconds).

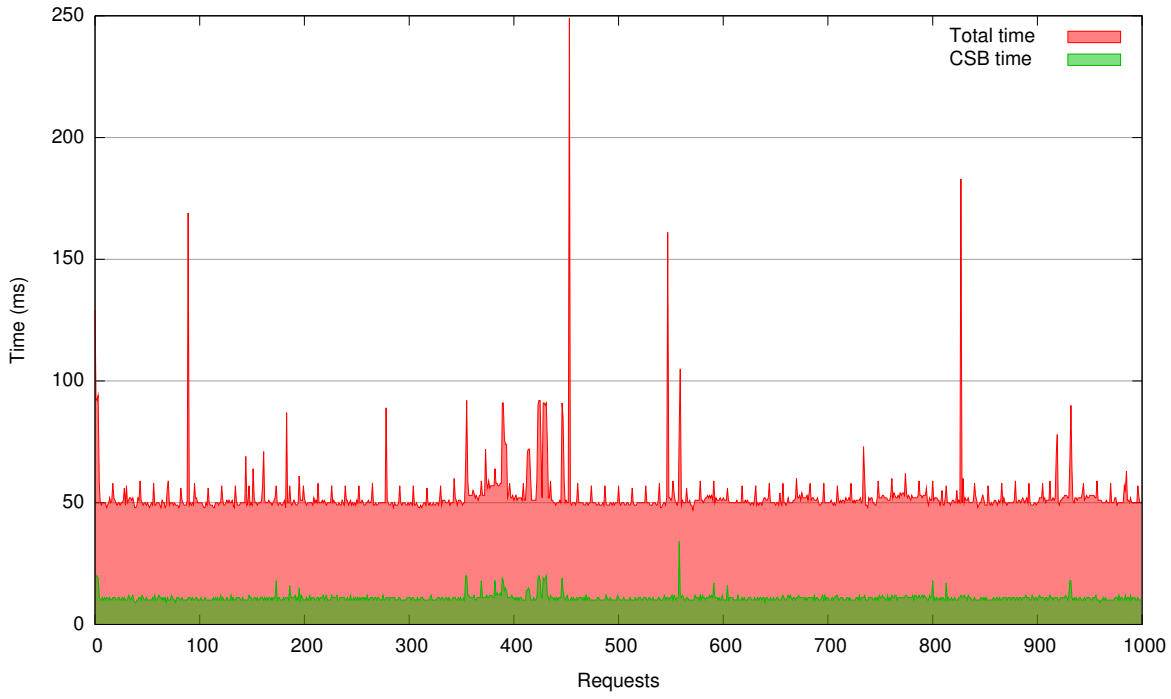

Figure 5.7: Manifest assessment responsiveness

	PaaS Manager	CSB	Total
Average (ms)	35.878	10.893	52.976
Confidence Interval (95%)	0.645	0.819	0.710
Standard Deviation (ms)	10.3867	1.641	11.449
Error (%)	1.797	0.935	1.341

Table 5.1: Statistics of one thousand invocations towards the Manifest API service

5.2.3 SCALING AND MIGRATION OF APPLICATIONS

This section demonstrates how elasticity of Web applications is obtained in a PaaS cloud using the CSB platform. In addition, results of migrating an application from one PaaS provider to another are validated. It is assumed that the application was previously deployed to a PaaS cloud, Cloud Foundry, and it is a Java Web application developed to output the network port instances are running at. Also for demonstrating another result of the herein presented work, the CSB Java CLI is used to request the scaling of the application `myJavaWebApp` from one instance up to two instances and migration process. The Java CLI uses HTTP Basic Authentication [57] to authenticate on the CSB and invokes HTTP requests to the REST API services.

```
$ java -jar csb-client-0.0.1-SNAPSHOT-jar-with-dependencies.jar --scale-app
    myJavaWebApp 2
[2013-10-22 10:02:02,842] Target server is set to URI
    http://10.115.1.19:8080/csb/rest
[2013-10-22 10:02:03,230] Get connection for route
    HttpRoute[{}->http://10.115.1.19:8080]
[2013-10-22 10:02:03,243] Connecting to 10.115.1.19:8080
[2013-10-22 10:02:03,291] CookieSpec selected: best-match
[2013-10-22 10:02:03,302] Re-using cached 'basic' auth scheme for
    http://10.115.1.19:8080
[2013-10-22 10:02:03,309] Attempt 1 to execute request
[2013-10-22 10:02:03,309] Sending request: PUT
    /csb/rest/apps/myJavaWebApp/scale/2 HTTP/1.1
[2013-10-22 10:02:03,311] >> PUT /csb/rest/apps/myJavaWebApp/scale/2 HTTP/1.1
[2013-10-22 10:02:20,004] >> [... HTTP headers ...]
[2013-10-22 10:02:20,004] Receiving response: HTTP/1.1 200 OK
[2013-10-22 10:02:20,004] << HTTP/1.1 200 OK
[2013-10-22 10:02:20,004] << [... HTTP headers ...]
[2013-10-22 10:02:20,009] Connection can be kept alive indefinitely
[2013-10-22 10:02:20,011] Application myJavaWebApp scaled to 2 instances
    successfully
```

Listing 4: Scaling an application through the CSB Java CLI

Listing 4 shows the CLI invoking a HTTP PUT request to the `/apps/:appId/scale/:numInstances` REST API service of which response returned successfully (`HTTP 200 OK`). HTTP headers were omitted for simplicity. Listing 5 shows two HTTP GET requests to the Web application index page. As expected, there are two application instances running and serving users' HTTP requests. The PaaS cloud set up a

load balancer, of which selection of the load balancing algorithm (e.g., Round Robin, Dynamic Round Robin, least connections or random) depends upon on each PaaS platform setup. One instance is running internally on port 63126 while the other instance is running internally on port 62491, but both externally accessible on port 80.

```
$ curl http://myJavaWebApp.cfapps.io
Application instance running on port: 63126

$ curl http://myJavaWebApp.cfapps.io
Application instance running on port: 62491
```

Listing 5: Load balancing of HTTP requests between multiple instances

Redeploying an application from a cloud provider to another is possible. The user can at anytime request the CSB to migrate his application to a specific cloud and it will be done for him automatically without users needing to create and redeploy by themselves. The CSB handles all the necessary steps internally. Application gets a new public URL.

Application configurations may need to be updated in order to reflect configuration data on the new provider. A common example is database credentials that can be reconfigured with no human interaction whatsoever. Some PaaS platforms (e.g., Cloud Foundry platform) features auto configuration tools that detect the type of the Web application and reconfigure accordingly, without even needing to apply environment flags or other possible helpers.

5.2.4 WEB PORTAL AND MONITORING APPLICATION RESOURCES USAGE

Figure 5.8, Figure 5.9 and Figure 5.10 are examples of Web pages of the developed Web Portal interface. Figure 5.8 lists all applications created in the CSB by the authenticated user. In that page, users can check the current status of all applications, the PaaS provider to where each application was deployed, open the Web site and delete applications. Clicking on the "New App" button, users are redirected to a new Web page (Figure 5.9).

In this new Web page users can comfortably describe their application requirements to which a list of PaaS CSPs matching the preconditions is disclosed. This process of which the users provides the application requirements is mapped to the Manifest data model in a JSON data format. Changing form values results in such values to be dynamically sent to the CSB for re-assessment with the list of PaaSs being live updated. Upon form submission, a new application is registered. The CSB creates a SCM repository in accordance to the SCM repository type previously selected. Pushing code changes to the SCM repository will trigger a new application deployment with the latest code available in the repository.

Once the application is created and deployed, users can perform a diversity of actions. Examples of actions include: create, bind and delete services, scale up/down instances,

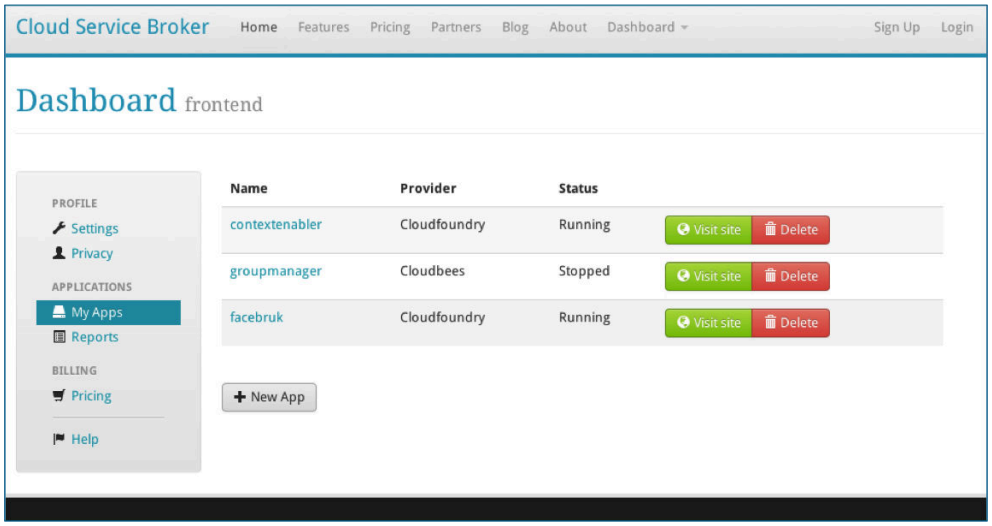

Figure 5.8: Web page listing all applications created.

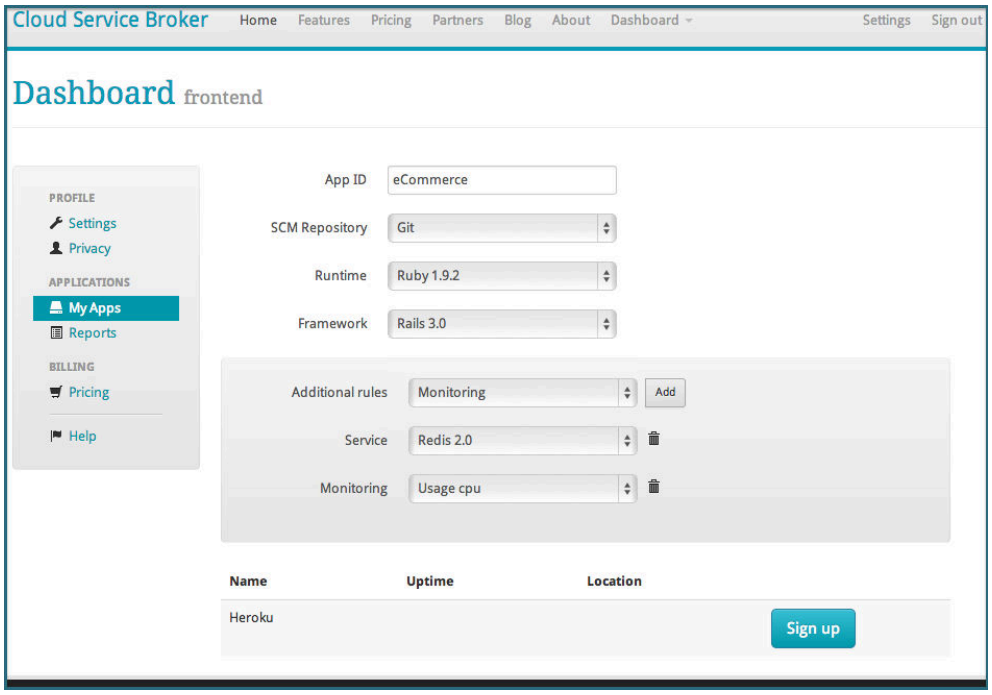

Figure 5.9: Web page where recommendation is given based on requirements.

migrate the application from the current PaaS to another, and check the application logs (see Figure 5.10).

Yet another feature of the CSB platform offered to users is the capability to collect and report monitoring application resources usage. Available metrics can vary depending on the PaaS applications are running on. The CSB harmonizes different metric names across PaaSs in a common metric name. An example of this feature is illustrated in Figure 5.11. A table of metrics data is shown and a chart of metrics is plotted. Users can zoom in or out in the chart to a given temporal sampling or click on the predefined buttons on the top left corner

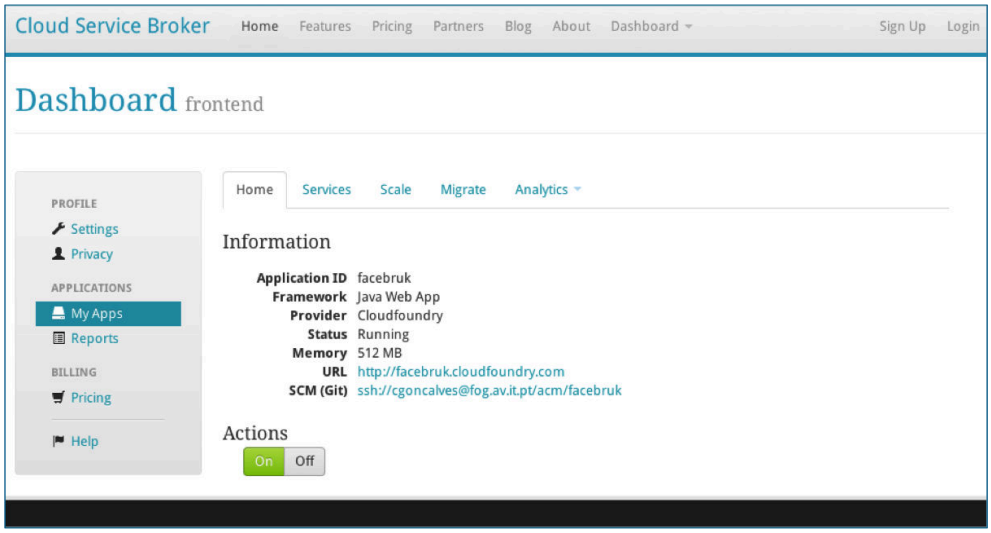

Figure 5.10: Home of an application where operations can be performed

to display data from the last minute, five minutes or all. The chart can as well be printed and downloaded as an image. New metric data is fetched and periodically live updated.

Collecting these information and reporting to users is of the most relevance. Based on the resource usage data, users can decide of whether the running instances suit the demand needs. Upon such analysis, new instances can be provisioned, this is scale up, or the number of running instances can be scaled down.

Users are authenticated on the Web Portal via OAuth mechanism to the CSB. The interface can be seen as a third-party entity that consumes data from the CSB and on top of that adds value-added to it with an easy and graphical interactive representation of the all RESTful API resources and services to the end user.

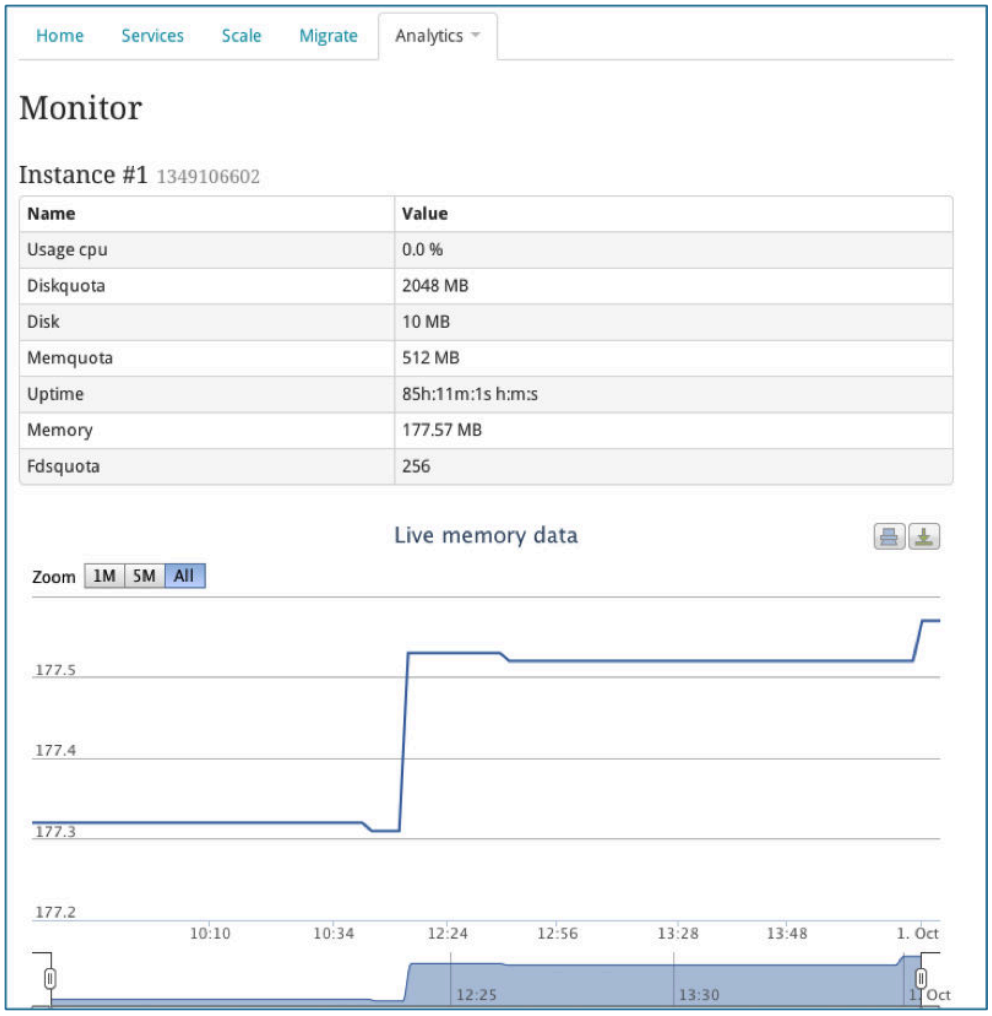

Figure 5.11: Live application monitoring on the Web Portal

5.2.5 APPLICATION PROVISIONING ON A PRIVATE PLATFORM AS A SERVICE

A full cycle for deploying an application on a private PaaS comprehends the five stages enumerated in Section 4.2.3 and described in more detail in Section 5.1.3. To corroborate the effectiveness of the implemented solution on provisioning a private PaaS to the CSB on top of an IaaS cloud, a Sugar[28] instance will be deployed on that same private PaaS. Sugar is a flexible customer relationship management web-based software system developed by SugarCRM. The system includes sales force automation, marketing campaigns, support cases, project management and calendaring. Sugar is available in both open source and commercial editions; the one used for testing was the open source edition, SugarCE (Community Edition) version 6.5.16. Sugar proves to be a good candidate for testing because it's an open-platform trusted and used by hundreds of companies known worldwide[29]. Sugar framework is written

[28]Sugar: `http://www.sugarcrm.com`

[29]SugarCRM customers include Coca-Cola Enterprise, FujiFilm, GALP Energia SA and Zurich Insurance Group Ltd: `http://www.sugarcrm.com/customers`

in PHP programming language and supports multiple databases (e.g., MySQL, Microsoft SQL Server, IBM DB2 and Oracle Database), web servers (Apache and Microsoft IIS) and operating systems (Windows, Unix, Linux, Mac OS X and IBM i).

Following a close look on how the process flows is presented and inspected some of the actions involved between the components of the platform during the various stages. HTTP headers were omitted for simplicity. Moreover, validating the CSB gem client library a short Ruby client in Listing 6 was written for creating the application and a database. For deploying a Sugar application, a Manifest document file was created specifying PHP as both runtime and framework, a MySQL database as a service, the *PT_OPENSTACK* PaaS provider for hosting the application, and added to the root directory along with the Sugar source code.

```
require 'csb'

options = {}
options[:consumer_key] = CSB_CONSUMER_KEY
options[:consumer_secret] = CSB_CONSUMER_SECRET
options[:oauth_token] = OAUTH_TOKEN
options[:oauth_token_secret] = OAUTH_SECRET

client = Csb::Client.new(options)
client.create_app 'sugar', 'git'
client.create_service 'sugar_db', 'sugar', 'MYSQL_5_1'
```

Listing 6: Creating an application via the CSB Ruby gem

In stage 1, the application source code is pushed to the remote Git repository. Associated with this stage are sequences 1-6 outlined in Figure 5.5. The CSB detects that an uninstantiated private PaaS was requested and quickly calls the IaaS Manager to create a virtual machine via the CIMI API. The virtual machine will be instantiated on the OpenStack cloud setup and mentioned in Section 5.2.1. The IaaS Manager CIMI response is listed in Listing 7.

```
> POST http://10.115.1.19:8080/iaasmanager/cimi/machines
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MachineCreate xmlns="http://schemas.dmtf.org/cimi/1">
  <machineTemplate>
    <machineConfig
        href="http://10.115.1.19:8080/iaasmanager/cimi/machine_configurations/
␣␣␣␣␣␣␣␣3200263630478682511364080097905965549 23"/>
    <machineImage
        href="http://10.115.1.19:8080/iaasmanager/cimi/machine_images/
␣␣␣␣␣␣␣␣cb92c1cf-2863-45d5-8dd7-5274669158ea"/>
  </machineTemplate>
</MachineCreate>

<
<Machine xmlns="http://schemas.dmtf.org/cimi/1"
    resourceURI="http://schemas.dmtf.org/cimi/1/Machine">
  <id>http://10.115.1.19:8080/iaasmanager/cimi/machines/
      96028645-b95b-4c4d-9e86-894554cf1523</id>
  <name>server2013-11-07 14:09:27 +0000</name>
```

```
  <description>No description set for Machine server2013-11-07 14:09:27
      +0000</description>
  <created>2013-11-07T14:09:25Z</created>
  <realm>default</realm>
  <machineImage href="http://10.115.1.19:8080/iaasmanager/cimi/machine_images/
␣␣␣␣␣␣cb92c1cf-2863-45d5-8dd7-5274669158ea" />
  <state>CREATING</state>
  <cpu>1</cpu>
  <memory>4194304</memory>
  <disks href="http://10.115.1.19:8080/iaasmanager/cimi/machines/
␣␣␣␣␣␣96028645-b95b-4c4d-9e86-894554cf1523/disks" />
  <volumes href="http://10.115.1.19:8080/iaasmanager/cimi/machines/
␣␣␣␣␣␣␣␣96028645-b95b-4c4d-9e86-894554cf1523/volumes" />
</Machine>
```

Listing 7: Request to and response from the IaaS Manager on creating a machine

Stage 2 lasts from sequence flow 7 to 16 when the virtual machine starts the SSH server. This constitutes the end of the scheduled *MachineJob* job. Following next is stage 3 with *PaasBootstrapJob* job generating flows 17-21. In the course of this stage, the Private PaaS Manager is invoked to bootstrap a PaaS instance on top of the virtual machine. It promptly responds with a `HTTP 202 Accepted` and schedules and runs a background job to bootstrap the cloud (flows 19-20). The output of these steps is in Listing 8 and the Cloud Foundry PaaS started. Last line outputted fires off stage 4.

```
14:13:37 web.1     | 10.115.1.19 - - [07/Nov/2013 14:13:37] "POST /setup
    HTTP/1.1" 202
14:13:41 resque.1 | Running setup...
14:14:38 resque.1 | Replacing vcap.me with cf.cgoncalves.pt in files listed in
    cf.txt...
14:16:49 resque.1 | Starting PaaS instance...
14:17:52 resque.1 | Targeting deployment "devbox" with cloudfoundry home
    "/home/ubuntu/cloudfoundry"
14:17:52 resque.1 | Setting up cloud controller environment
14:17:52 resque.1 | Setting up the uaa environment
14:17:52 resque.1 | Using cloudfoundry config from
    /home/ubuntu/cloudfoundry/.deployments/devbox/config
14:18:20 resque.1 | cloud_controller              :      RUNNING
14:18:40 resque.1 | stager                        :      RUNNING
14:19:00 resque.1 | rabbitmq_node                 :      RUNNING
14:19:20 resque.1 | postgresql_node               :      RUNNING
14:19:34 resque.1 | dea                           :      RUNNING
14:19:54 resque.1 | mysql_node                    :      RUNNING
14:20:14 resque.1 | rabbitmq_gateway              :      RUNNING
14:20:28 resque.1 | uaa                           :      RUNNING
14:20:49 resque.1 | vblob_node                    :      RUNNING
14:21:09 resque.1 | router                        :      RUNNING
14:21:25 resque.1 | mongodb_node                  :      RUNNING
14:21:35 resque.1 | redis_gateway                 :      RUNNING
14:21:42 resque.1 | postgresql_gateway            :      RUNNING
14:21:47 resque.1 | mysql_gateway                 :      RUNNING
14:21:51 resque.1 | vblob_gateway                 :      RUNNING
14:21:55 resque.1 | mongodb_gateway               :      RUNNING
```

```
14:22:01 resque.1 | health_manager                    :       RUNNING
14:22:09 resque.1 | redis_node                        :       RUNNING
14:22:11 resque.1 | Registering user on http://api.cf.cgoncalves.pt/users
14:22:13 resque.1 | User registered!
14:22:13 resque.1 | Setup about to end. Calling POST callback
    http://10.115.1.19:8080/csb/rest/ppm/register?id=PT_OPENSTACK
```

Listing 8: Bootstrapping a PaaS cloud

On stage 4, once the CSB accepts the callback invocation it registers the new PaaS on the PaaS Manager and updates its features on flows 22-28. Last stage, stage 5, the CSB selects all pending applications to the until now uninitialized PaaS for deployment, and proceeds accordingly. The *sugar* application is now deployed and accessible from `http://sugar.cf.cgoncalves.pt` (see Figure 5.12)

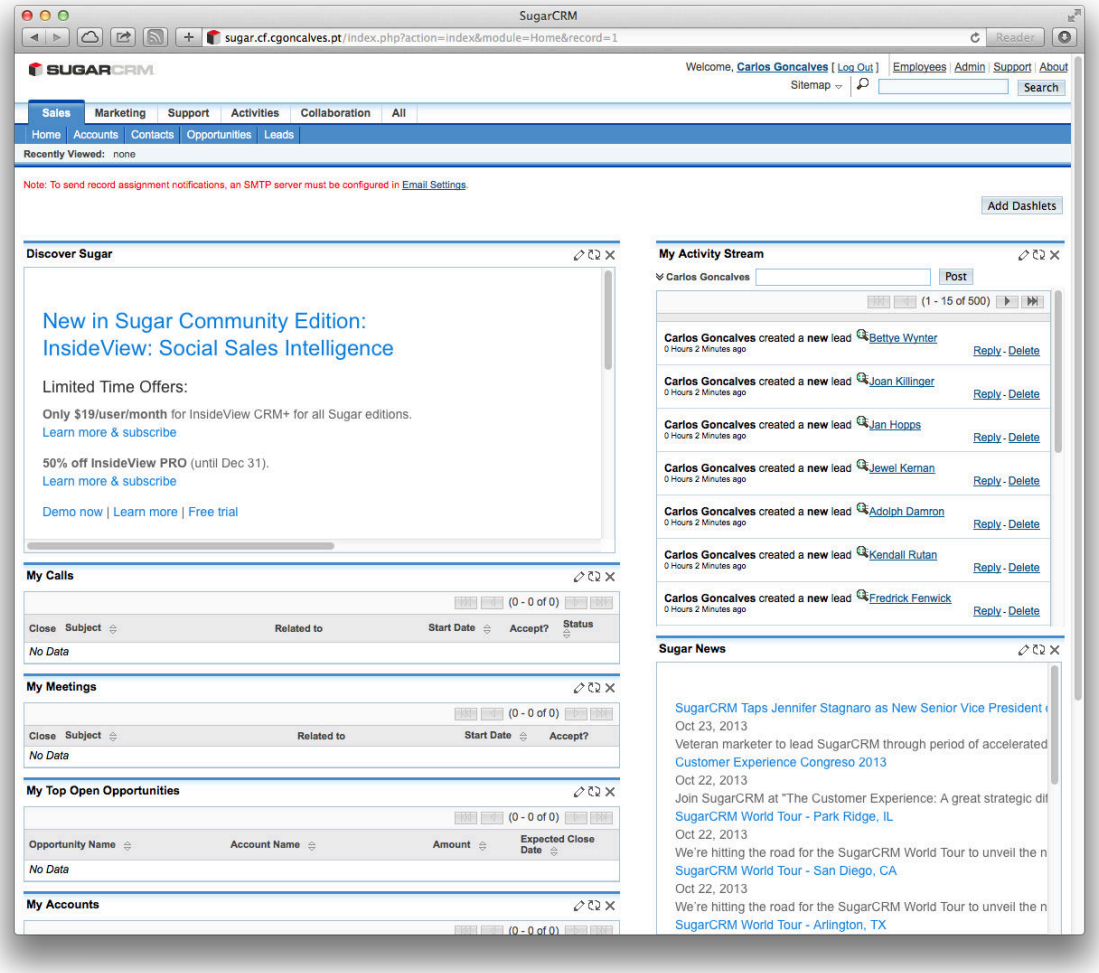

Figure 5.12: Sugar application deployed on a private PaaS

CHAPTER 6

Conclusions

Cloud computing presents a new paradigm for business companies to explore new ways of offering better solutions to customers. IT infrastructure resources are optimized to accommodate and distribute loads equally across the available computational resource pools.

Many organizations are researching, developing and venturing on cloud services solutions and services on the cloud. CSPs are emerging and with time revealing to compete among themselves on distinct technical offers (e.g., runtimes, frameworks and add-ons), monitoring and accountability, pricing and SLA. Nevertheless, vendor lock-in is in many cases imposed. Migrating from one CSP to another can be tricky, time-consuming and expensive. Cloud interoperability should be of paramount importance.

The proposed CSB is a key part of the framework by delivering and evaluating, the most appropriate platform from a catalog of miscellaneous PaaS offerings, based on the application profile. However, if the supported platforms don't cover all the users' needs, the developer shall be forwarded and assisted in preparing and setting up a PaaS on demand, settled on lowest cloud layer solutions. This approach create a bond between IaaS and PaaS, which is orchestrated via the CSB and implemented through the outlined cloud managers, PaaS Manager, IaaS Manager and Private PaaS Manager.

The CSB was developed taking into consideration many aspects of nowadays barriers on aggregating on a single platform a variety of PaaS clouds. CSB operates as a gateway to a range of services through a public API and other user interfaces: Web portal (Section 5.1.4), Ruby client library (Section 5.1.4) and a command-line interface. The process of recommending users clouds that better fits their needs was attained. A rule engine (Section 5.1.3) was developed and formalized a XML/JSON document called Manifest. Third-party entities can be given permission to access end user's information on the platform thanks to OAuth [59] and hence creating new services on top of the CSB.

Four PaaS cloud services were supported and extensively tested successfully. IaaS clouds were also implemented, being the OpenStack platform the one studied. Extending IaaS

offerings to new clouds is uncomplicated since the IaaS Manager (Section 4.2.2) was designed and implemented from the very beginning to support multi-cloud capabilities.

An example of market players who can take advantage of such multipurpose cloud framework are the communications service providers. With the new added-value service providers, operators are becoming just a data-pipe guarantying connectivity between both ends. Communication service providers are undoubtedly interested in taking a share of the growing cloud market, setting a major position. The exploitation of two-sided business revenues with third-parties developers toward the management and migration of applications to private or hybrid cloud products is a likely model.

6.1 FUTURE WORK

The work hereby presented was supported by PT Inovação S.A. It is currently being integrated in a larger project where the CSB plays an important role on recommending CSPs and executing applications on the cloud. Some important ideas for the further improvement of the system are summarized as follows.

Predicting costs for clients running applications through the CSB is troublesome as PaaS CSPs bill differently according to their business models (e.g., quantity of allocated RAM, number of running instances, bandwidth). Studies on a general business model are necessary. It would then be needed to add new rules to the rule engine, to extend the Manifest API service for returning the quota for each CSP, and to present the information on the various available user interfaces.

At the time the CAMP API was announced, the CSB had already a defined public API. Reevaluating further developments on the CAMP API, and implementing it on the public CSB API, is of major importance for greater interoperability with third-party entities.

The CSB can request the Private PaaS Manager for provisioning a new Cloud Foundry PaaS instance, and deploy applications into there. Although currently it does not have the intelligence for requesting the unprovisioning of a PaaS instance. This is of great interest for the operator of the CSB platform for cost-savings. In case a PaaS instance turned not to be running applications any more (e.g., applications hosted there were deleted or migrated to another CSP), the instance can be unprovisioned.

References

[1] C. Gonçalves, D. Cunha, P. Neves, P. Sousa, J. P. Barraca, and D. Gomes, "Towards a Cloud Service Broker for the Meta-Cloud", in *CRC 2012 12ª Conferência sobre Redes de Computadores*, D. Gomes and P. Salvador, Eds., Universidade de Aveiro, 2012, pp. 7–13. [Online]. Available: http://revistas.ua.pt/index.php/crc/article/view/2035.

[2] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges", *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010, ISSN: 1867-4828. [Online]. Available: http://dx.doi.org/10.1007/s13174-010-0007-6.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing", University of California, Berkeley, Tech. Rep., 2009.

[4] P. Mell and T. Grance, "The NIST Definition of Cloud Computing", National Institute of Standards and Technology (NIST), Gaithersburg, MD, Tech. Rep. 800-145, Sep. 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[5] S. K. Nair, S. Porwal, T. Dimitrakos, A. J. Ferrer, J. Tordsson, T. Sharif, C. Sheridan, M. Rajarajan, and A. U. Khan, "Towards Secure Cloud Bursting, Brokerage and Aggregation", in *2010 Eighth IEEE European Conference on Web Services*, IEEE, Dec. 2010, pp. 189–196, ISBN: 978-1-4244-9397-5. DOI: 10.1109/ECOWS.2010.33. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5693261.

[6] D. Petcu, "Portability and interoperability between clouds: challenges and case study", in *Proceedings of the 4th European conference on Towards a servicebased internet*, vol. 6994 LNCS, Springer-Verlag Berlin, 2011, pp. 62–74, ISBN: 9783642247545. DOI: 10.1007/978-3-642-24755-2_6. [Online]. Available: http://dl.acm.org/citation.cfm?id=2050869.2050876.

[7] M. Belcourt, "Outsourcing — The benefits and the risks", *Human Resource Management Review*, vol. 16, no. 2, pp. 269–279, Jun. 2006, ISSN: 1053-4822. DOI: http://dx.doi.org/10.1016/j.hrmr.2006.03.011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1053482206000234.

[8] S. Slaughter and S. Ang, "Employment outsourcing in information systems", *Communications of the ACM*, vol. 39, no. 7, pp. 47–54, Jul. 1996, ISSN: 00010782. DOI: 10.1145/233977.233994. [Online]. Available: http://portal.acm.org/citation.cfm?doid=233977.233994.

[9] C. Harding, *Cloud Computing for Business - The Open Group Guide*, First edit, M. S. Pamela K. Isom and C. Harding, Eds. Van Haren Publishing, Zaltbommel, 2011, ISBN: 978 90 8753 657 2.

[10] N. Bridge and G. Research, *2013 Cloud Computing Survey*, 2013. [Online]. Available: http://www.northbridge.com/2013-cloud-computing-survey.

[11] C. S. D. Agostino, M. Ahronovitz, J. Armstrong, N. Davalbhakta, R. Gogulapati, E. Lau, E. Luster, A. A. M. Matsui, A. Mohammed, D. Moskowitz, M. Nolan, S. Porwal, A. R. Radhakrishnan, J.-l. Richet, P. Rimal, D. Russell, M. B. Sigler, K. Sreenivasan, R. Syputa, D. Tidwell, and K. Venkatraman, "Moving to the Cloud: A white paper produced by the Cloud Computing Use Cases Discussion Group", Tech. Rep. February, 2011, pp. 1–11.

[12] C. Apte, "Data mining: an industrial research perspective", *IEEE Computational Science and Engineering*, vol. 4, no. 2, pp. 6–9, 1997, ISSN: 10709924. DOI: 10.1109/99.609825. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=609825.

[13] B. Kepes, "Moving your Infrastructure to the Cloud: How to Maximize Benefits and Avoid Pitfalls", Diversity Limited, Tech. Rep., 2010, p. 20.

[14] R. Abrams, *Bringing the Cloud Down to Earth: How to choose, launch, and get the most from cloud solutions for your business*, R. Gaspar, Ed. PlanningShop, 2012, p. 168, ISBN: 9781933895291. [Online]. Available: http://dl.acm.org/citation.cfm?id=2361573.

[15] KPMG Advisory N.V, "From Hype to Future: KPMG's 2010 Cloud Computing Survey", p. 44, 2010. [Online]. Available: http://www.kpmg.com/ES/es/ActualidadyNovedades/ArticulosyPublicaciones/Documents/2010-Cloud-Computing-Survey.pdf.

[16] Intel IT Center, "What's Holding Back the Cloud?", Intel, Tech. Rep. May, 2012, p. 31. [Online]. Available: http://www.intel.com/content/dam/www/public/us/en/documents/reports/whats-holding-back-the-cloud-peer-research-report2.pdf.

[17] International Organization For Standardization, *ISO/IEC 7498-1:1994 Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model*, 1996. [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/s025022%5C_ISO%5C_IEC%5C_7498-3%5C_1997(E).zip.

[18] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing", *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1–11, Jan. 2011, ISSN: 10848045. DOI: 10.1016/j.jnca.2010.07.006. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1084804510001281.

[19] D. Zissis and D. Lekkas, "Addressing cloud computing security issues", *Future Generation Computer Systems*, vol. 28, no. 3, pp. 583–592, 2012, ISSN: 0167-739X. DOI: http://dx.doi.org/10.1016/j.future.2010.12.006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X10002554.

[20] L. M. Kaufman, "Data Security in the World of Cloud Computing", *IEEE Security & Privacy Magazine*, vol. 7, no. 4, pp. 61–64, Jul. 2009, ISSN: 1540-7993. DOI: 10.1109/MSP.2009.87. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5189563.

[21] G. a. Lewis, "Role of Standards in Cloud-Computing Interoperability", in *2013 46th Hawaii International Conference on System Sciences*, IEEE, Jan. 2013, pp. 1652–1661, ISBN: 978-1-4673-5933-7. DOI: 10.1109/HICSS.2013.470. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6480040.

[22] DMTF, "Open Virtualization Format Specification", DMTF Virtualization Management (VMAN) Initiative, Tech. Rep., 2010, pp. 1–42. [Online]. Available: http://www.dmtf.org/standards/ovf.

[23] ——, *DMTF's Open Virtualization Format Achieves ANSI Adoption*, 2010. [Online]. Available: http://dmtf.org/news/pr/2010/8/dmtf?s-open-virtualization-format-achieves-ansi-adoption (visited on 09/11/2013).

[24] CDMI, "ISO/IEC 17826:2012 - Information technology – Cloud Data Management Interface (CDMI)", Tech. Rep., 2012, p. 224. [Online]. Available: http://www.iso.org/iso/catalogue%5C_detail.htm?csnumber=60617.

[25] Storage Networking Industry Association, "Cloud Data Management Interface", pp. 1–224, 2012. [Online]. Available: http://snia.org/sites/default/files/CDMI%20v1.0.2.pdf.

[26] A. Edmonds, T. Metsch, A. Papaspyrou, and A. Richardson, "Open Cloud Computing Interface: Open Community Leading Cloud Standards.", *ERCIM News*, no. 83, pp. 23–24, 2010. [Online]. Available: http://ercim-news.ercim.eu/en83/special/open-cloud-computing-interface-open-community-leading-cloud-standards.

[27] T. Metsch, A. Edmonds, and R. Nyrén, "Open Cloud Computing Interface - Core", *Open Grid Forum*, p. 17, 2011. [Online]. Available: http://www.gridforum.org/Public%5C_Comment%5C_Docs/Documents/2010-12/ogf%5C_draft%5C_occi%5C_core.pdf.

[28] A. Edmonds, T. Metsch, and A. Papaspyrou, *Grid and Cloud Database Management*, S. Fiore and G. Aloisio, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 23–49, ISBN: 978-3-642-20044-1. DOI: 10.1007/978-3-642-20045-8. [Online]. Available: http://www.springerlink.com/index/10.1007/978-3-642-20045-8.

[29] A. Interface and M. C. Infrastructure, "Cloud Infrastructure Management Interface (CIMI) Model and RESTful HTTP-based Protocol An Interface for Managing Cloud Infrastructure", pp. 1–178, 2012.

[30] D. Davis and G. Pilz, "Cloud Infrastructure Management Interface (CIMI) Model and REST Interface over HTTP", *vol. DSP-0263, May*, 2012.

[31] T. Binz, G. Breiter, F. Leyman, T. Spatzier, and W.-s. Workflow, "Portable Cloud Services Using TOSCA", *IEEE Internet Computing*, vol. 16, no. 3, pp. 80–85, 2012, ISSN: 10897801. DOI: 10.1109/MIC.2012.43. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=%5C&arnumber=6188582%5C&contentType=Journals+%5C&+Magazines%5C&queryText=Portable+Cloud+Services+Using+TOSCA.

[32] OASIS Topology and Orchestration Specification for Cloud Applications Technical Committee, "Topology and Orchestration Specification for Cloud Applications", Tech. Rep. March, 2013, pp. 1–114.

[33] M. Carlson, M. Chapman, A. Heneveld, S. Hinkelman, D. Johnston-Watt, A. Karmarkar, T. Kunze, A. Malhotra, J. Mischkinsky, A. Otto, V. Pandey, G. Pilz, Z. Song, and P. Yendluri, "Cloud Application Management for Platforms Version 1.0", Tech. Rep., 2012.

[34] M. Ahrens, P. Barot, F. Behr, I. Brandic, M. Brunner, J. Clarke, S. Clayman, T. Coupaye, P. Dickman, M. Dor, A. Edlund, E. Elmroth, A. Ebert, J. Falkner, A. M. J. Ferrer, M. Fisher, A. Geiger, A. G. Tato, Y.-K. Guo, G. Hogben, K. Jeffery, R. Jimenez-Peris, F. Leymann, I. M. Llorente, B. di Martino, B. Neidecker-Lutz, K. Oberle, D. Petcu, T. Piwek, H. Schoening, T. Varvarigou, and Y. Wolfsthal, "Advances in Clouds", European Union, Tech. Rep., 2012, p. 84. [Online]. Available: http://cordis.europa.eu/fp7/ict/ssai/docs/future-cc-2may-finalreport-experts.pdf.

[35] D. Huang, "Mobile Cloud Computing", *IEEE COMSOC Multimedia Communications Technical Committee (MMTC) E-Letter*, vol. 6, no. 10, pp. 27–30, 2011. [Online]. Available: http://committees.comsoc.org/mmc/e-news/E-Letter-October11.pdf.

[36] A. Jamakovic, T. M. Bohnert, and G. Karagiannis, "Mobile Cloud Networking: Mobile Network, Compute, and Storage as One Service On-Demand", in *The Future Internet*, A. Galis and A. Gavras, Eds., Springer, 2013, pp. 356–358, ISBN: 978-3-642-38081-5. DOI: 10.1007/978-3-642-38082-2_33. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-38082-2%5C_33.

[37] R. Bryant, R. H. Katz, and E. D. Lazowska, *Big-Data Computing: Creating Revolutionary Breakthroughs in Commerce, Science and Society*, 2008.

[38] A. Voulodimos, S. V. Gogouvitis, N. Mavrogeorgi, R. Talyansky, D. Kyriazis, S. Koutsoutos, V. Alexandrou, E. Kolodner, P. Brand, and T. Varvarigou, "A Unified Management Model for Data Intensive Storage Clouds", in *Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on*, IEEE Computer Society, 2011, pp. 69–72. DOI: 10.1109/NCCA.2011.18. [Online]. Available: http://dx.doi.org/10.1109/NCCA.2011.18.

[39] S. V. Gogouvitis, G. Kousiouris, G. Vafiadis, E. K. Kolodner, and D. Kyriazis, "OPTIMIS and VISION cloud: how to manage data in clouds", in *Proceedings of the 2011 international conference on Parallel Processing*, ser. Euro-Par'11, Berlin, Heidelberg: Springer-Verlag, 2012, pp. 35–44, ISBN: 978-3-642-29736-6. DOI: 10.1007/978-3-642-29737-3_5. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29737-3%5C_5.

[40] M. Allalouf, A. Averbuch, L. Bonelli, P. Brand, G. Chevalier, M. Dao, A. Eckert, E. Elmroth, and S. Gogouvitis, "VISION Cloud D10.2: High Level Architectural Specification, Release 1.0", Tech. Rep., 2011. [Online]. Available: http://visioncloud.eu/resource.php?resourceID=193.

[41] M. Stonebraker, D. Abadi, D. J. DeWitt, S. Madden, E. Paulson, A. Pavlo, and A. Rasin, "MapReduce and parallel DBMSs: friends or foes?", *Commun. ACM*, vol. 53, no. 1, pp. 64–71, Jan. 2010, ISSN: 0001-0782. DOI: 10.1145/1629175.1629197. [Online]. Available: http://doi.acm.org/10.1145/1629175.1629197.

[42] C. Kotselidis, M. Ansari, K. Jarvis, M. Luján, C. Kirkham, and I. Watson, "DiSTM: A Software Transactional Memory Framework for Clusters", in *2008 37th International Conference on Parallel Processing*, IEEE, Sep. 2008, pp. 51–58. DOI: 10.1109/ICPP.2008.59. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4625832.

[43] M. M. Saad and B. Ravindran, "HyFlow: a high performance distributed software transactional memory framework", in *Proceedings of the 20th international symposium on High performance distributed computing*, ser. HPDC '11, New York, NY, USA: ACM, 2011, pp. 265–266, ISBN: 978-1-4503-0552-5. DOI: 10.1145/1996130.1996167. [Online]. Available: http://doi.acm.org/10.1145/1996130.1996167.

[44] P. Romano, L. Rodrigues, N. Carvalho, and J. Cachopo, "Cloud-TM: harnessing the cloud with distributed transactional memories", *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 1–6, Apr. 2010, ISSN: 0163-5980. DOI: 10.1145/1773912.1773914. [Online]. Available: http://doi.acm.org/10.1145/1773912.1773914.

[45] J. Martins, J. Pereira, S. M. Fernandes, and J. Cachopo, "Towards a Simple Programming Model in Cloud Computing Platforms", *2011 First International Symposium on Network Cloud Computing and Applications*, pp. 83–90, Nov. 2011. DOI: 10.1109/NCCA.2011.21. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6123443.

[46] M. Raskino and J. Lopez, "CEO Survey 2012: The Year of Living Hesitantly", Gartner, Tech. Rep., 2012. [Online]. Available: http://www.gartner.com/id=1957515.

[47] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, "NIST Cloud Computing Reference Architecture Recommendations of the National Institute of Standards and", *Nist Special Publication*, http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/ReferenceArchitectureTaxonomy/NIST_SP_500-292_-_090611.pdf, vol. 292, no. 9, p. 35, 2011. [Online]. Available: http://www.nist.gov/customcf/get%5C_pdf.cfm?pub%5C_id=909505.

[48] D. C. Plummer, B. J. Lheureux, and F. Karamouzis, "Defining Cloud Services Brokerage : Taking Intermediation to the Next Level", Gartner, Tech. Rep. October, 2010, p. 13. [Online]. Available: http://www.gartner.com/id=1448121.

[49] Gartner, *Gartner Says Cloud Consumers Need Brokerages to Unlock the Potential of Cloud Services*, 2009. [Online]. Available: http://www.gartner.com/newsroom/id/1064712.

[50] ActiveState Software Inc, *ActiveState Acquires Appsecute: Private-PaaS Leader Purchases Social DevOps Solutions Provider*, 2013. [Online]. Available: http://www.activestate.com/press-releases/activestate-acquires-appsecute.

[51] D. Petcu, B. Martino, S. Venticinque, M. Rak, T. Máhr, G. Lopez, F. Brito, R. Cossu, M. Stopar, S. Šperka, and V. Stankovski, "Experiences in building a mOSAIC of clouds", *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 2, no. 1, p. 12, 2013, ISSN: 2192-113X. DOI: 10.1186/2192-113X-2-12. [Online]. Available: http://www.journalofcloudcomputing.com/content/2/1/12.

[52] D. Petcu, C. Crăciun, M. Neagul, S. Panica, B. Di Martino, S. Venticinque, M. Rak, and R. Aversa, "Architecturing a sky computing platform", in *Proceedings of the 2010 international conference on Towards a service-based internet*, ser. ServiceWave'10, Berlin, Heidelberg: Springer-Verlag, 2011, pp. 1–13, ISBN: 978-3-642-22759-2. [Online]. Available: http://dl.acm.org/citation.cfm?id=2050107.2050109.

[53] N. Grozev and R. Buyya, "Inter-Cloud architectures and application brokering: taxonomy and survey", *Wiley InterScience Software: Practice and Experience*, pp. 1–22, Dec. 2012, ISSN: 00380644. DOI: 10.1002/spe.2168. [Online]. Available: http://dx.doi.org/10.1002/spe.2168%20http://doi.wiley.com/10.1002/spe.2168.

[54] P. Pawluk, B. Simmons, M. Smit, M. Litoiu, and S. Mankovski, "Introducing STRATOS: A Cloud Broker Service", in *2012 IEEE Fifth International Conference on Cloud Computing*, IEEE, Jun. 2012, pp. 891–898, ISBN: 978-1-4673-2892-0. DOI: 10.1109/CLOUD.2012.24. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6253593.

[55] A. J. Ferrer, F. Hernández, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame, W. Ziegler, T. Dimitrakos, S. K. Nair, G. Kousiouris, K. Konstanteli, T. Varvarigou, B. Hudzia, A. Kipp, S. Wesner, M. Corrales, N. Forgó, T. Sharif, and C. Sheridan, "OPTIMIS: A holistic approach to cloud service provisioning", *Future Generation Computer Systems*, vol. 28, no. 1, pp. 66–77, Jan. 2012, ISSN: 0167-739X. DOI: http://dx.doi.org/10.1016/j.future.2011.05.022. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X1100104X.

[56] D. Cunha, P. Neves, and P. Sousa, "A Platform-as-a-Service API Aggregator", in *Advances in Information Systems and Technologies SE - 75*, ser. Advances in Intelligent Systems and Computing, Á. Rocha, A. M. Correia, T. Wilson, and K. A. Stroetmann, Eds., vol. 206, Springer Berlin Heidelberg, 2013, pp. 807–818, ISBN: 978-3-642-36980-3. DOI: 10.1007/978-3-642-36981-0_75. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-36981-0%5C_75.

[57] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart, *HTTP Authentication: Basic and Digest Access Authentication*, RFC 2617 (Draft Standard), Internet Engineering Task Force, 1999.

[58] S. Turner and T. Polk, *Prohibiting Secure Sockets Layer (SSL) Version 2.0*, RFC 6176 (Proposed Standard), Internet Engineering Task Force, 2011.

[59] E. Hammer-Lahav, *The OAuth 1.0 Protocol*, RFC 5849 (Informational), Internet Engineering Task Force, 2010.

[60] D. Eastlake 3rd and T. Hansen, *US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)*, RFC 6234 (Informational), Internet Engineering Task Force, 2011.

REST API

The CSB API supports a Representational State Transfer (REST) model for accessing a set of resources through a fixed set of operations. The following resources are accessible through the RESTful model.

CSB REST API

APPLICATION RESOURCES

Resource	**Description**
`GET /apps`	Returns all applications of the authenticated user.
`GET /apps/:appId`	Returns the application specified by the `:appId` parameter.
`POST /apps/:appId?repository_type=:type`	Creates a new application identified by the `:appId` parameter with a SCM repository type specified by the `:type` parameter.
`DELETE /apps/:appId`	Deletes the application specified by the `:appId` parameter.
`PUT /apps/:appId/deploy`	Trigger a new deployment process of the application identified by the `:appId` parameter.
`GET /apps/:appId/info`	Retrieves a more detailed information of the application identified by the `:appId` parameter.
`GET /apps/:appId/log`	Gets the log of the application specified by the :appId from the PaaS platform it is running on
`POST /apps/:appId/manifest`	Adds the Manifest document file included in the HTTP body into the repository of the application specified by the `:appId` parameter.
`GET /apps/:appId/monitor`	Retrieves metric data from the application specified by the `:appId` parameter.
`GET /apps/:appId/status`	Gets the status of the application specified by the `:appId` parameter.

`PUT /apps/:appId/start`	Starts the application identified by the `:appId` parameter.
`PUT /apps/:appId/stop`	Stops the application identified by the `:appId` parameter.
`PUT /apps/:appId/restart`	Restarts the application identified by the `:appId` parameter.
`PUT /apps/:appId/migrate/:providerId`	Migrates the application specified by the `:appId` parameter to the PaaS provider identified in the `:providerId` parameter.
`PUT /apps/:appId/scale/:numInstances`	Scales up or down a number of instances in the `:numInstances` parameter of the application specified by the `:appId` parameter.
`GET /apps/:appId/services`	Gets all services associated to the application specified by the `:appId` parameter.
`GET /apps/:appId/services/:serviceId`	Gets the service identified by the `:serviceId` associated to the application specified in the `:appId` parameter.
`POST /apps/:appId/services/:serviceId`	Creates a new service identified by the `:serviceId` parameter and associates it application defined in the `:appId` parameter.
`DELETE /apps/:appId/services/:serviceId`	Deletes the service identified by the `:serviceId` associated to the application specified in the `:appId` parameter.

Table 1: CSB Application resources

MANIFEST RESOURCES

Resource	**Description**
`POST /manifest`	Makes a decision-assessment and returns a list of PaaS providers matching the SLA elements in the Manifest file.

Table 2: CSB Manifest resources

USER RESOURCES

Resource	**Description**
`GET /users`	Get all registered users.
`POST /users`	Creates a new user.
`DELETE /users/:userId`	Deletes a registered user specified by the :userId parameter.

Table 3: CSB User resources

PRIVATE PAAS MANAGER RESOURCES

Resource	Description
`POST /ppm/register`	Registers a new PaaS platform in the CSB.

Table 4: CSB Private PaaS Manager resources

IAAS MANAGER REST API

MACHINE RESOURCES

Resource	Description
`GET /users`	Get all registered users.
`POST /users`	Creates a new user.
`DELETE /users/:userId`	Deletes a registered user specified by the :userId parameter.

Table 5: IaaS Manager Machine resources

IMAGE RESOURCES

Resource	Description
`GET /machines`	Returns all machines in a CimiMachineCollection.
`GET /machines/:id`	Returns a machine in a CimiMachine.
`POST /machines`	Creates a new machine and returns a CimiMachine containing information regarding just created machine.

Table 6: IaaS Manager Image resources

PRIVATE PAAS MANAGER REST API

PAAS RESOURCES

Resource	Description
`GET /providers`	Returns a list of all PaaS providers.
`POST /setup`	Setups a new PaaS in a machine.
`POST /start`	Starts a PaaS.
`POST /stop`	Stops a PaaS.
`POST /restart`	Restarts a PaaS.
`GET /components`	Retrieves a collection of all PaaS components.
`GET /components/:id`	Retrieves information of a PaaS component specified by the `:id` parameter.

Table 7: Private PaaS Manager resources

CSB Data Models

MANIFEST

A Manifest is a well defined and structured document. Listing 9 formalizes the abstract representation in a XML schema. Listing 10 is an example of a Manifest submitted to the CSB for assessment and Listing 11 the response returned to the user containing a list of PaaS providers.

```
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="manifest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="rules" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="rule" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="provider" type="xs:string" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="rule">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string" minOccurs="0"/>
        <xs:element name="params" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="param" type="xs:string" minOccurs="0"
                 maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="provider">
    <xs:sequence>
      <xs:element name="id" type="xs:string" minOccurs="0"/>
    </xs:sequence>
```

```
  </xs:complexType>
</xs:schema>
```

Listing 9: Manifest XML schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<manifest>
  <rules>
    <rule>
      <name>runtime</name>
      <params>
          <param>Ruby</param>
          <param>less-equal</param>
          <param>1.9.3</param>
      </params>
    </rule>
    <rule>
      <name>framework</name>
      <params>
        <param>Rails</param>
        <param>greater-equal</param>
        <param>3.0</param>
      </params>
    </rule>
    <rule>
      <name>service</name>
      <params>
        <param>POSTGRESQL_9_1</param>
      </params>
    </rule>
    <rule>
      <name>metric</name>
      <params>
        <param>response_time</param>
      </params>
    </rule>
    <rule>
      <name>metric</name>
      <params>
        <param>usage_cpu</param>
      </params>
    </rule>
  </rules>
</manifest>
```

Listing 10: Manifest data model example in XML format

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<paas_providers>
  <paas_provider>
    <id>HEROKU</id>
    <runtimes>
      <runtime>
        <id>JAVA_1_6</id>
```

```
      <name>Java</name>
      <version>1.6</version>
    </runtime>
    <runtime>
      <id>SCALA</id>
      <name>Scala</name>
      <version>-</version>
    </runtime>
    <runtime>
      <id>NODE_0_6_8</id>
      <name>Node</name>
      <version>0.6.8</version>
    </runtime>
    <runtime>
      <id>RUBY_1_9_2</id>
      <name>Ruby</name>
      <version>1.9.2</version>
    </runtime>
    <runtime>
      <id>PYTHON_2_7_2</id>
      <name>Python</name>
      <version>2.7.2</version>
    </runtime>
    <runtime>
      <id>LEININGEN_1_7_1</id>
      <name>Leiningen</name>
      <version>1.7.1</version>
    </runtime>
    <runtime>
      <id>RUBY_1_9_3</id>
      <name>Ruby</name>
      <version>1.9.3</version>
    </runtime>
  </runtimes>
  <frameworks>
    <framework>
      <id>GRAILS</id>
      <name>Grails</name>
      <version>-</version>
    </framework>
    <framework>
      <id>JAVA_WEB_APP</id>
      <name>Java Web App</name>
      <version>-</version>
    </framework>
    <framework>
      <id>PLAY</id>
      <name>Play!</name>
      <version>-</version>
    </framework>
    <framework>
      <id>SPRING</id>
      <name>Spring</name>
      <version>-</version>
```

```
    </framework>
    <framework>
      <id>NODE</id>
      <name>Node</name>
      <version>-</version>
    </framework>
    <framework>
      <id>SINATRA</id>
      <name>Sinatra</name>
      <version>-</version>
    </framework>
    <framework>
      <id>RACK</id>
      <name>Rack</name>
      <version>-</version>
    </framework>
    <framework>
      <id>RAILS_3_0</id>
      <name>Rails</name>
      <version>3.0</version>
    </framework>
    <framework>
      <id>DJANGO</id>
      <name>Django</name>
      <version>-</version>
    </framework>
    <framework>
      <id>EXPRESS</id>
      <name>Express</name>
      <version>-</version>
    </framework>
  </frameworks>
  <service_vendors>
    <service_vendor>
      <id>POSTGRESQL_8_3</id>
      <name>PostgreSQL</name>
      <version>8.3</version>
    </service_vendor>
    <service_vendor>
      <id>POSTGRESQL_9_1</id>
      <name>PostgreSQL</name>
      <version>9.1</version>
    </service_vendor>
    <service_vendor>
      <id>MYSQL_5_5_25</id>
      <name>MySQL</name>
      <version>5.5.25</version>
    </service_vendor>
    <service_vendor>
      <id>REDIS_2_0</id>
      <name>Redis</name>
      <version>2.0</version>
    </service_vendor>
  </service_vendors>
```

```
    <metrics>
      <metric name="apdex"/>
      <metric name="usage_cpu"/>
      <metric name="memory"/>
      <metric name="response_time"/>
      <metric name="throughput"/>
      <metric name="usage_database"/>
    </metrics>
  </paas_provider>
</paas_providers>
```

Listing 11: Decision-assessment result in XML