Ref. Ares(2013)129715 - 01/02/2013

4CaaSt

Building the PaaS Cloud of the Future

Experimental Prototype of Software Components and Documentation:

D4.3.2

WP4 – Resource Deployment and Management

Dissemination Level: Public

Lead Editor: Sergio García Gómez, TID

31/01/2013

Status: Final

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 258862

Seventh Framework Programme

FP7-ICT-2009-5

Service and Software Architectures, Infrastructures and Engineering

This is a public deliverable that is provided to the community under a Creative Commons Attribution 3.0 Unported License: http://creativecommons.org/licenses/by/3.0/

You are free:

to Share — to copy, distribute and transmit the work

to Remix — to adapt the work

Under the following conditions:

Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

With the understanding that:

Waiver — Any of the above conditions can be waived if you get permission from the copyright holder.

Public Domain — Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.

Other Rights — In no way are any of the following rights affected by the license:

Your fair dealing or fair use rights, or other applicable copyright exceptions and limitations;

The author's moral rights;

Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.

Notice — For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

For a full description of the license legal terms, please refer to:

http://creativecommons.org/licenses/by/3.0/legalcode

Contributors:

Sergio García Gómez (TID)
Christof Momm (SAP)
Rouven Krebs (SAP)
Henar Muños Frutos (TID)
Manuel Escriche Vicente (TID)
Jozsef Biro (NSN)
Daniel Toth (NSN)
Pavlos Kranas (NTUA)
Jose Luis Vázquez-Poletti (UCM)

Internal Reviewer(s):

François Exertier (BULL)
Ricardo Jimenez (UPM)

Version History

Version	Date	Authors	Sections Affected
0.1	21.12.2011	Sergio García Gómez (TID)	Table of Contents
0.2	22.12.2011	Henar Muñoz (TID)	Section 3.2
0.3	05.01.2012	Christof Momm, Rouven Krebs (SAP)	Section 3.1. Section 2.3.1
0.3	13.01.2012	Jozsef Biro (NSN) and Daniel Toth (NSN)	Section 3.3
0.4	20.01.2012	Sergio García Gómez (TID)	Sections 1, 2, 4 and 5
0.5	20.01.2012	Henar Muñoz Frutos (TID)	Section 2.3.2
0.9	21.01.2012	Sergio García Gómez (TID)	Executive Summary and delivery of draft

Copyright © TID and other members of the 4CaaSt consortium 2013

1.0	02.02.2012	Sergio García Gómez (TID)	Final version after reviewers comments.
1.1	10.01.2012	Sergio Garcia Gomez (TID)	Update of table of contents
1.2	22.01.2012	Sergio Garcia Gomez (TID)	Update of sections: Introduction, Prototype Description, Conclusions, References.
1.3	23.01.2012	Christof Momm (SAP)	Update of section 2.1, 3.1 and 4.1
1.4	23.01.2012	Pavlos Kranas (NTUA)	New section 3.2.
1.5	23.01.2012	Henar Muñoz (TID)	New section 3.3 and update of 3.4.
1.6	28.01.2012	Jozsef Biro (NSN)	Update of section 3.5
1.7	29.01.2012	Jose Luis Vázquez-Poletti (TID)	Integration and draft edition.
1.8	29.01.2012	Sergio Garcia Gomez (TID)	Integration and draft edition.
1.9	31.01.2012	François Exertier (BULL), Ricardo Jimenez (UPM)	Document Review
2.0	31.01.2012	Sergio Garcia Gomez (TID)	Final version

Copyright © TID and other members of the 4CaaSt consortium 2013

Table of Contents

Executive Summary 9
1. Introduction 10
1.1. Purpose and scope 10
1.2. Document Overview 10
2. Prototype description 11
2.1. Overview 11
2.2. Infrastructure and Deployment 13
2.3. Relationship to other 4CaaSt Work Packages 15
2.3.1. Interfaces to WP3 15
2.3.2. Interfaces to WP5 15
3. Components management 16
3.1. Deployment Design Manager 16
3.1.1. Implementation Design 16
3.1.2. Compilation Instructions 16
3.1.3. Installation Instructions 16
3.1.4. Configuration 18
3.1.5. Execution Instructions 19
3.2. Resource Requirements Calculator (RRC) 19
3.2.1. Basic Concept 19
3.2.2. Compilation Instructions 20
3.2.3. Installation Instructions 21
3.2.4. Configuration 21
3.2.5. Execution Instructions 21
3.3. PaaS Manager 21
3.3.1. Compilation Instructions 22
3.3.2. Installation Instructions 22
3.3.3. Execution Instructions 24
3.4. Service Manager 24
3.4.1. Compilation Instructions 25
3.4.2. Installation Instructions 25
3.4.3. Execution Instructions 27
3.5. REC Manager 27
3.5.1. Basic REC Concepts 28
3.5.2. REC Manager Core 28
3.5.3. Chef Server 30

3.5.4. Chef-Agent 32
3.6. OpenNebula Clusters 33
4. User guide 36
4.1. Deployment Design Manager UI 36
4.1.1. Deployment Browser 36
4.1.2. Debug Interface 39
5. Conclusions 42
6. References 43

List of Figures

Figure 1. WP4 components deployment. 14
Figure 2. Tomcat Web Application Manager 18
Figure 3. Service Manager Architecture. 25
Figure 4. Managing OpenNebula Clusters with the Sunstone web UI. 35
Figure 5. Deployment Browser 36
Figure 6. Deployment Design Graph 37
Figure 7. Deployment Design XML 38
Figure 8. Original Blueprint 38
Figure 9. Generated OVF++ 39
Figure 10. Debug Interface 40
Figure 11. Generation of Sample Data 41

Copyright © TID and other members of the 4CaaSt consortium 2013

Abbreviations

4CaaSt	Building the PaaS Cloud of the future
AC	Application Component
APT	Advanced Packaging Tool
ARB	Abstract Resolved Blueprint
EMF	Eclipse Modelling Framework
FLEXISCALE	Public cloud platform provided by Flexiant Ltd.
IP	Internet Protocol
OS	Operating System
OVF	Open Virtualization Format
PaaS	Platform as a Service
PIC	Platform Instance Component
REC	Runtime Execution Environment
REST	Representational State Transfer
RESTful	Conforming to REST constraints
RRC	Resource Requirements Calculator
SDC	Software Deployment and Configuration
SMI	Service Manager Inteface (API Claudia)
SSH	Secure Shell
URL	Unified Resource Locator
VMI	Virtual Manager Interface (API Flexiscale)
WAR	Web application ARchive
WP	Work Package
YUM	Yellowdog Updater, Modified

Copyright © TID and other members of the 4CaaSt consortium 2013

Executive Summary

This deliverable describes the second release of the components delivered from WP4 to the 4CaaSt platform. These components are available now in the 4CaaSt Subversion[1] repository and the present deliverable explains how they are deployed, installed and configured as part of the 4CaaSt platform. The deployment of the components for demo purposes has been done on Flexiscale facilities.

Functionality-wise, the second release of WP4 components introduces a new component that is able to orchestrate the provision of cloud resources during an application deployment (the PaaS Manager) and their elasticity, together with a refinement of the deployment design process. This includes scalable and non-scalable designs, together with the generation of rules and hints for resources calculation and elasticity, and improved REC management capability that allows to define multiple recipes per product.

The next release of the components will be devoted to fine tuning of the elasticity processes and to the NaaS integration with the overall platform.

[1] https://svn.morfeo-project.org/4caast/

1. Introduction

1.1. Purpose and scope

The present deliverable aims at documenting the delivery of the second release of WP4 software components, corresponding to Month 32 in the project plan. This document reflects the technical and practical issues of those components described in the deliverable D4.2.2 [1], including how these components should be deployed, configured and used in the context of a 4CaaSt platform installation.

1.2. Document Overview

The following sections of the document are structured in this way:

- Chapter 1 (this chapter) gives an introduction to the document.
- Chapter 2 provides a high level description of the WP4 prototype, including relationships to other WPs.
- Chapter 3 describes the components and how they are deployed, installed and configured.
- Chapter 4 is devoted to the user interface of the components; in WP4 all the components belong to the backend layer and do not have a GUI.
- Chapter 5 gives some conclusions about the results obtained in the second iteration.

This document is an update of the deliverable D4.3.1. These are the main changes from previous version:

- Update of section 2, with the description of the current status of the components (update from D1.1.3b [6]). The information about the interfaces has been removed since it is described in other documents.
- Update of section 3.1, 3.4 and 3.5. Sections 3.2, 3.3 and 3.6 are new.
- New section 4.1.

Copyright © TID and other members of the 4CaaSt consortium 2013

2. Prototype description

2.1. Overview

Release 2.0 of 4CaaSt WP4 components covers several aspects of the features described in the deliverable D.1.1.3b [6], while others remain pending for future iterations. In the following, we provide a brief summary of the delivered functionality and remaining issues for the final release. The involved components are highlighted using *italic* font. For further details on the feature implementation status and future plans for release 3, please refer to the resubmission of D.1.1.3b [6].

- **F#WP4.01:** Deployment of applications over a platform
  - Supported functionality
    - Evaluation of resource requirements for given QoS constraints using *Resource Requirements Calculator*
    - Generation + evaluation of different (elastic) designs and provisioning of SaaS users using *Deployment Design Manager*
    - Generation of executable deployment designs based on OVF++ (*Deployment Design Manager*)
    - Automated deployment of generated executable designs (OVF++) using *PaaS Manager*, *REC Manager* and *Service Manager*
    - Chef enabler scripts for major 4CaaSt components available
  - Major open issues for release 3
    - Generation of elastic deployments
    - Approach for handling complex architectures
    - Support for mixed component/service deployments
- **F#WP4.02:** Automated Application Elasticity
  - Supported functionality
    - Definition of elasticity rules based on RIF format, reference by blueprints
    - Engine for executing rules in *PaaS Manager*, supporting usage of platform level KPIs
    - Implementation of elasticity actions at the *Service Manager* level, by cloning of virtual machines.
  - Major open issues for release 3
    - Rules and KPI pass-through in *Deployment Design Manager*
    - Deployment design switch as elasticity actions offered by *Deployment Design Manager*
    - Integration of elasticity support in *PaaS Manager* with monitoring analytics (predictions, ranges of normality)
- **F#WP4.03:** PaaS API
  - Supported functionality
    - API for management of the catalogue of products

        - API to deploy applications, also over different environments
        - Support for queries regarding the deployed resources
    - Major open issues for release 3
        - Configuration and application management APIs (start, stop, etc.)
- **F#WP4.04:** IaaS Aggregated API
    - Supported functionality
        - Full TCloud support (release 1)
        - OpenNebula driver added to *Service Manager* in the aggregated API implementation for the management of VMs and networks
    - Major open issues for release 3
        - Integration of NaaS if required at the IaaS Aggregated API level
- **F#WP4.05:** REC Creation & Virtual Machine Template (or base virtual appliance)
    - Supported functionality
        - *REC Manager* supports
            - End-to-end forwarding and of all artifacts, cookbooks and attributes
            - Fully modular installation/configuration of products
            - SaaS deployment (tenant provisioning)
            - Monitoring deployment
    - Major open issues for release 3
            - Support for template image (*REC Manager* + *Service Manager*)
            - *REC Manager* support for complex deployment cases, e.g. PIC-in-a-PIC and combined PICs
- **F#WP4.06:** OVF Extensions
    - Supported functionality
        - OVF++ defined supporting product specification, cookbooks, scalability information and balancer definition
        - OVF++ for services (SaaS offerings)
    - Major open issues for release 3
        - NaaS specification
- **F#WP4.07:** Improved network configuration for applications
    - Supported functionality (*NaaS Manager*)
        - Support of creation of virtual LANs on a point-to-point basis
        - Support of bandwidth in path computation
        - Automatic topology discovery
        - Support of configuring virtual and HW switches (OF1.0) end-to-end
        - GUI to display virtual and physical topology and allocations
        - Basic link failover handling
    - Major open issues for release 3
        - Integration with *PaaS Manager* based on OCCI-based “A” interface

        - End-to-End NaaS support (Deployment Design→OVF++→NaaS Manager)
        - QoS support
- **F#WP4.08:** Automated monitoring configuration
    - Supported functionality
        - KPI definition in Blueprints
        - Definition of chef recipes for probes management (deploy/start/stop/undeploy probes) on PICs and ACs, and how they are used by *REC Manager*.
    - Major open issues for release 3
        - End to end monitoring configuration (from KPI in blueprint to probe deployment from Chef Recipes)
        - Extended set of probes for major 4CaaSt products
- **F#WP4.09:** OpenNebula Cluster support
    - Supported functionality
        - Cluster management feature is fully operative and integrated with the latest OpenNebula version
    - Major open issues for release 3
        - Integration with *PaaS Manager* to support enforcement of availability SLAs
- **F#WP4.10:** Resource requirement calculation (SLA translation)
    - Implementation status
        - BP extension to include hints for resource requirements for ACs/PICs and definition of available IaaS offerings
        - Calculation of resource requirements and evaluation of feasibility using *Resource Requirements Calculator*
    - Major open issues for release 3
        - Entrain the concept of '4CaaSt Powerpoints' as hints at the PIC layer to decouple the AC from the REC layer
        - Benchmarking approach for 4CaaSt scenarios

2.2. Infrastructure and Deployment

The Figure 1 shows the deployment of the WP4 components corresponding to the second release of the platform. The different components have been deployed in separate resources from Flexiscale infrastructure [5].

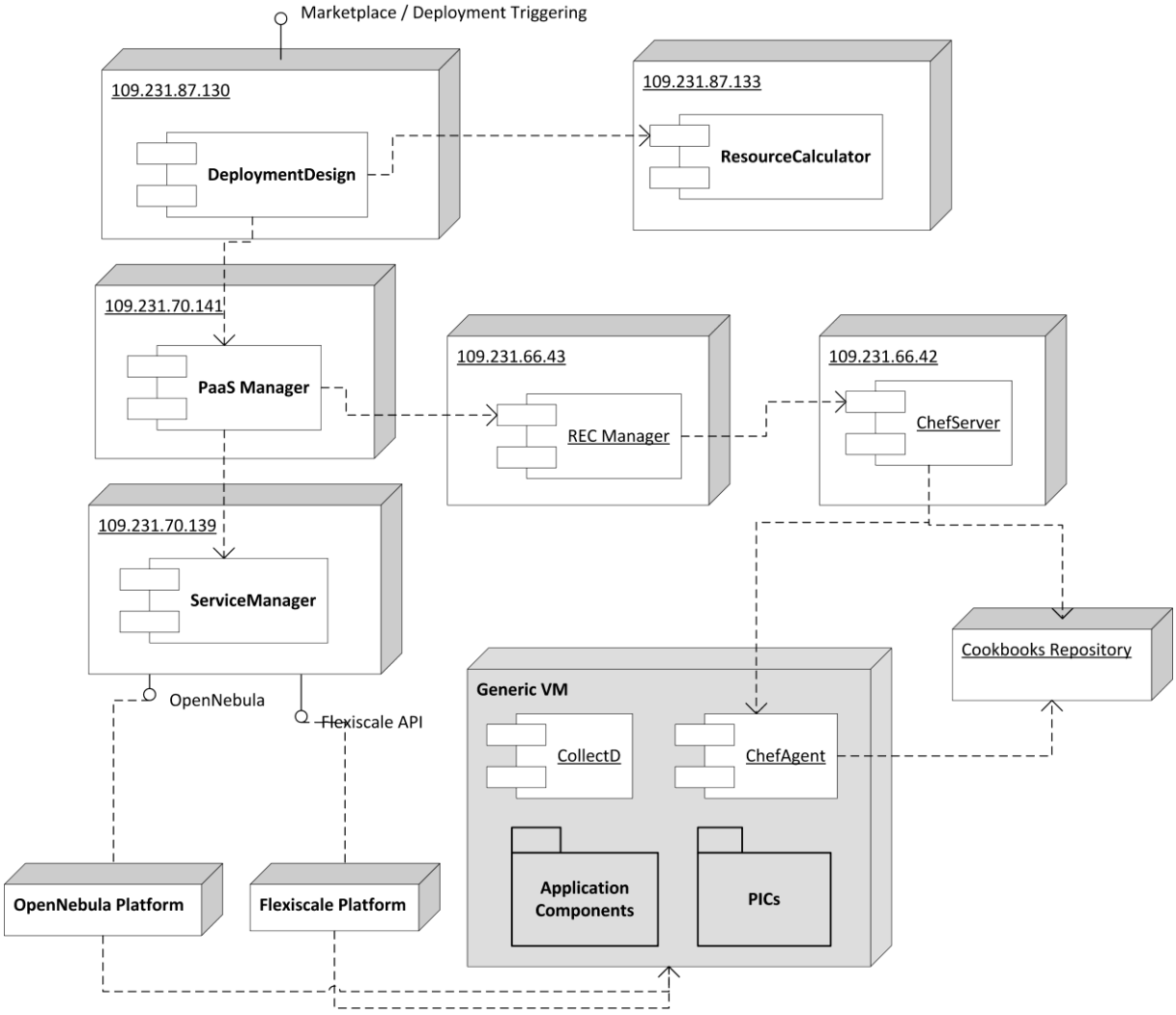

Figure 1. WP4 components deployment.

The links between the components described in previous figure represent the connection among them, meaning that they offer interfaces as described in D4.2.2 [1]:

- DeploymentDesign component offers an interface so that a deployment from WP3 can be triggered.
- The ResourcesCalculator offers an interface to DeploymentDesign to calculate the amount of resources and elasticity rules based on the customer's requirements and constraints.
- The PaaS Manager offers an extended OVF API to request an application deployment based on an architectural description.
- The Service Manager exports a TCloud based API to enable the deployment of resources described through OVF. It requests the provisioning of IaaS resources to the OpenNebula or Flexiscale platforms.
- The RECManager exposes an interface through which the software deployment and configuration process is started.
- A Chef Server provides Chef API in order to manage the deployment process through the REC manager. The cookbooks are stored separately in a repository.

- The figure also includes a typical VM for an application deployment, which includes the basic CollectD probe, the Chef agent to install software, and the products (PICs) and components the application consist of.

2.3. Relationship to other 4CaaSt Work Packages

2.3.1. Interfaces to WP3

The interface exposed to WP3 to trigger the deployment is described in D4.2.2 [1]. It receives an Abstract Resolved Blueprint together with the SLA constraints from the customer and a 4CaaSt identifier and it returns the completed 4CaaSt identifier that represents the deployment for further information polling.

2.3.2. Interfaces to WP5

The deliverable D5.2.2 [4] describes the interfaces offered to WP4: on the one hand there is an interface for configuring the monitoring probes, and for offering monitoring information through the TCloud Monitoring interface to the Service Manager, for elasticity purposes.

3. Components management

3.1. Deployment Design Manager

The Deployment Design Manager component receives the description of the application from the 4CaaSt marketplace in terms of an Abstract Resolved Blueprint (ARB) along with a definition of the QoS requirements and translates this into an optimized executable deployment based on OVF++. During optimization, the Resource Requirements Calculator component is used to calculate the resource requirements and general feasibility for different deployment designs. For details on the internal architecture and design of the Deployment Design Manager as well as the Resource Requirements Calculator please refer to D4.2.2 [1].

3.1.1. Implementation Design

This section briefly introduces the implementation design of the Deployment Design Component.

The general design is based on the RP1 version as documented in D4.3.1 [2]. Accordingly, there is still a REST Facade component hiding internal complexity and providing easy access for WP2 and WP3 and the actual Deployment Controller is based on the Command pattern. Major extensions to the last version are:

- Predefined command chains for different types of deployments (product and service deployment) as well as test and productive mode available
- New command DeploymentDesignBuilderCommand for service deployment generation
- Integration of Resource Requirements Calculator and extended optimization strategy in new DeploymentDesignBuilderCommand for products
- New OVF++ generation using OVF library provided by TID
- New internal deployment design model (EMF) supporting pass-through of cookbooks
- REST façade supports submission of QoS requirements in addition to ARB
- No standalone version anymore. Component only delivered as war-file, which has to be deployed on a Tomcat (tested with version 6.x.x)
- Added a graphical user interface to inspect the generated deployment designs and test the component (through an additional debug interface)

3.1.2. Compilation Instructions

Due to IPR constraints the sources are not published for the Deployment Design Component. We only deliver a pre-compiled binary, which is available via `https://svn.forge.morfeo-project.org/4caast/trunk/WP4/DeploymentManager`.

3.1.3. Installation Instructions

The software has been tested on Ubuntu 10.04 64-bit, but it should work on later versions and other Linux systems as well.

3.1.3.1. Preconditions

Ensure that the following conditions are fulfilled on your system:

- The OS is Ubuntu 10.04 64-bit.
- A java runtime environment 1.6 is installed.
- A Tomcat servlet container 6.x.x is installed (not tested with higher versions, but it should do)
- Port 8080 is not in use and not blocked by security settings.
- Tomcat Manager plugin installed, otherwise root access to the system is required for deployment
- Outgoing calls to the PaaSManager endpoint are possible and not blocked by security settings.
- Check that application can create new folders in usr directory (Should be possible per default).

3.1.3.2. Installation

1) Download war-file via `https://svn.forge.morfeo-project.org/4caast/trunk/WP4/DeploymentManager`
2) Deploy war-file in Tomcat, either by copying it to the webapps directory manually (using SSH) or by using the Tomcat Manager interface (see Figure 2) as follows:
    a. Access Tomcat Manager under `http://<server-ip>:8080/manager/html`
    b. Select war file to upload and press <deploy>

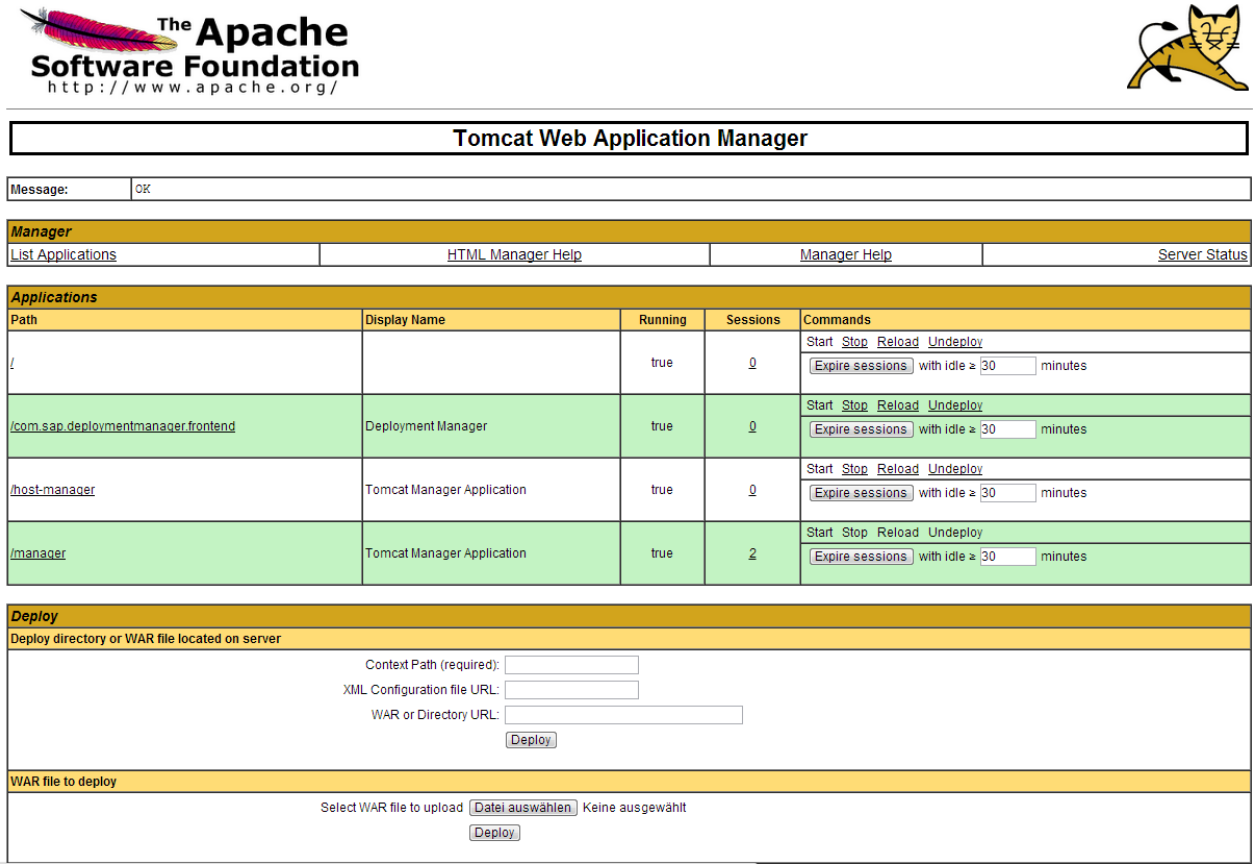

Figure 2. Tomcat Web Application Manager

3.1.4. Configuration

There are three items, which have to be configured:

a) IP + Port of the used PaaS Manager. If not specified, a default IP from the last review will be used (which is not likely to work!)

b) IP + Port of the used Resource Requirements Calculator. If not specified, an internal default library will be used.

c) Define location of repository file, which contains all past deployments (database). If not specified, a default is used which should work with standard Ubuntu installations.

All of these configurations are done within the `instantiationproperties.properties` file, which is located within the war file in the following directory:

```
com.sap.deploymentmanager.frontend.war\WEB-INF\classes\deploymentcontroller\process
```

The properties to set are:

```
#PaaSManager Endpoint. If empty, a default IP is used.
smi.host=
smi.port=
#Resource Requirements Calculator Endpoint. If empty, the internal version is used.
rrc.host=
rrc.endpoint=
#Repository location
```

```
repositoryLocation=/temp/ddm_repository/
```

To change the configuration, please extract the `instantiationproperties.properties` file from the war (e.g. using WinRAR), edit the properties and pack it back in the war.

3.1.5. Execution Instructions

The web app is started automatically by the Tomcat server after deployment and then uses the standard Tomcat logging mechanism (std.out and err.out in log directory). If available, you can use the Tomcat Manager (`http://<server-ip>:8080/manager/html`) to start, stop or undeploy the application.

If started successfully, the Deployment Design Manager should be available under the following URL:

`http://<server-ip>:8080/com.sap.deploymentmanager.frontend/`

Please refer to section 4.1 for a detailed guide on how to use the offered UI.

3.2. Resource Requirements Calculator (RRC)

The Resource Requirements Calculator receives from the Deployment Design component a list of proposed deployment options, and the Abstract Resolved Blueprint (ARB), enhanced with the user hints and the initialization rules, and the possible IaaS offerings of the selected infrastructure providers. The concepts of the hints and the initialization rules are already described at the relevant D4.2.2 document [1]. The Resource Requirements Calculator checks the feasibility of each proposed deployment, and prioritizes them according to its cost. It firstly associates each virtual node with the corresponding requirements for resources (in terms of CPU units and RAM size). Then, it calculates the overall requirements for every physical node, and checks if there is an available offering from the infrastructure provider that satisfies the needs for resources. It also takes into account scalability options for each application component. After calculating the resources needed for every proposed deployment option, the RRC defines the exact number of CPU units and RAM size to every node, and then runs an optimization algorithm to find the deployment design with the minimum cost. It then returns back the input received from the Deployment Design, fulfilled with the necessary values for proper initialization of the resources.

3.2.1. Basic Concept

The Resource Requirement Calculator consists of the requirements resolver subcomponent, which is the core of the RRC, and the façade. The requirements resolver uses various utilities that have been developed that associate the user selected hints with the proper initialization rules, and calculates the overall resources needed for every physical node. It also uses an internal cost resolver, which executes and optimization algorithm to prioritize the deployment design options according to their costs. The façade provides a REST Service as the interface with the Deployment Design component, thus encapsulating the internal complexity. Moreover, a standalone java client is also provided, which gives the ability to directly import the Resource Requirement Calculator as a jar file.

The overall functionality is implemented by executing the following method:

- **calculateAllDeploymentDesigns**: Receives a list of different Deployment Design options, and the ARB file, and returns back the valid deployment design options, their ranking in terms of minimum cost, and the exact specification of CPU units and RAM size.

3.2.2. Compilation Instructions

The Resource Requirement Calculator repository is

`https://svn.forge.morfeo-project.org/4caast/trunk/WP4/ResourceRequirementsCalculator/`

To compile it, download the content and use the NetBeans IDE 7.x to build, or any given standard java compiler. It requires Java 1.7.x version. The project also requires a bundle of libraries to build. These libraries must be imported to the project. They can be found at `https://svn.forge.morfeo-project.org/4caast/trunk/WP4/ResourceRequirementsCalculator/External_jars/`.

Make sure that you have previously completely removed any predefined libraries or external jars, and then re-import them. Also, make sure that the 'project.properties' file, under '/nbproject/' folder properly points to the valid folder where those jars are located.

After installing all external jars, try to clean and build the project, and deploy it to your favorite server container (this project has been tested with Tomcat7[2] server). You should access the http://<hostname>/ ResourceRequirementsCalculator/index.jsp page to verify that the project is built without errors.

The main service class of the façade is located here:

`gr.ntua.forcaast.rrcalculator.service. ResourceRequirementCalculatorService`

The REST service can be accessed via the following URL:

`http://<hostname>/ResourceRequirementsCalculator/resources/ res-req-calculator/`

You can test the service by sending a POST request with an xml input parameter, which can be found at the following URL:

`https://svn.forge.morfeo-project.org/4caast/trunk/WP4/ResourceRequirementsCalculator/files/`

Once you verify that everything works as expected, then right click the project folder under the NetBeans ID, and rebuild the project. A war file named "ResourceRequirementsCalculator.war" should be created under the 'dist' subfolder of the project.

[2] Note that different components are deployed on separate nodes and using different versions of Tomcat is not an issue.

3.2.3. Installation Instructions

This software was run on Debian 3.2.32-1 x86_64 GNU/Linux, but it should work on later version and other Linux systems as well.

3.2.3.1. Preconditions

Ensure that the following conditions are fulfilled on your system:

- The OS is Debian 3.2.32 or other compatible OS.
- You have root access to the system (needed for the default Installation steps described below).
- A java runtime environment 1.7 is installed
- A Tomcat7 application server is installed
- Port 8080 is not in use and not blocked by security settings. An alternative port number is possible at the relevant configuration file of the tomcat application server. This file is usually located at the `<tomcat>/conf/server.xml` file.

`Installation`

Start the tomcat server by executing "`./startup.sh`" (without the brackets), under the `<tomcat>/bin` directory.

Make sure you have proper administrative roles. This can be configured by editing the <tomcat>/conf/tomcat-users.xml file.

Copy the previously auto generated war file under the `<tomcat>/webapps` subfolder, or use the graphical interface by accessing the `<hostname>/manager/html` page.

3.2.4. Configuration

There are no specific actions needed for the configuration of the Resource Requirements Component at its current release.

3.2.5. Execution Instructions

3.2.5.1. Starting, Stopping and Logging

After ensuring that the Resource Requirements Calculator has been properly compiled and installed, you can simply start the application server by executing the <tomcat>/bin/startup.sh script. The server will be started with the REST service already deployed. In order to stop the service, you can simply execute the ./shutdown.sh script under the same folder. The logging files can be found under the `<tomcat>/logs` folder.

3.3. PaaS Manager

The PaaS Manager component is a new component included in the 4CaaSt architecture release 2. It is an orchestrator in charge of the management of the environments, that is, the whole infrastructure hardware and software required for the deployment of the application plus the management of the application itself. Details about the PaaS Manager can be found in D4.2.2 [1].

3.3.1. Compilation Instructions

The PaaS Manager source repository is `https://svn.forge.morfeo-project.org/4caast/trunk/WP4/paas-manager`. You can access to it with:

```
svn co https://svn.forge.morfeo-project.org/4caast/trunk/WP4/paas-manager
```

To compile and generate PaaS Manager binaries packages

```
mvn clean install
```

3.3.2. Installation Instructions

3.3.2.1. Requirements

In order to execute the PaaS Manager, it is needed to have previously installed the following software:

- Tomcat 7.X.X[3]
- PostgreSQL[4]

3.3.2.2. Database configuration

The PaaS Manager needs to have PostgreSQL installed in service mode and a database created called SDC (Software Deployment and Configuration). For CentOS, these are the instructions: First, it is required to install the PostgreSQL[5].

```
yum install postgreql postgresql-server posgresql-cotrib
```

3.3.2.2.1. Start Postgresql

Type the following commands to install the PostgreSQL as service and restarted

```
# chkconfig --add postgresql
# chkconfig postgresql on
# service postgresql start
```

3.3.2.2.2. Create the DB

Connect to PostgreSQL Server using

```
# su - postgres
```

[3] http://tomcat.apache.org/download-70.cgi

[4] http://www.postgresql.org/

[5] http://wiki.postgresql.org/wiki/YUM_Installation

Connect as postgres to the PostgreSQL database and set the password for user using `alter user` as below:

```
$ psql postgres postgres;
$ alter user postgres with password 'postgres';
```

Create the SDC DB

```
createdb paasmanager;
```

3.3.2.3. Apache Tomcat configuration

3.3.2.3.1. Install Tomcat 7

Install Tomcat 7 together with standard Tomcat samples, documentation, and management web apps:

```
yum install tomcat7-webapps tomcat7-docs-webapp tomcat7-admin-webapps
```

Start/Stop/Restart Tomcat 7 as a service.

start:

```
sudo service tomcat7 start
```

stop:

```
sudo service tomcat7 stop
```

restart:

```
sudo service tomcat7 restart
```

Add Tomcat 7 service to the autostart

```
sudo chkconfig tomcat7 on
```

3.3.2.4. Install PaaS Manager application

Once the prerequisites are satisfied, you shall create the context file. To do that, change sdc.xml found in distribution file and store it in folder `$CATALINA_HOME/conf/Catalina/localhost`.

See the snippet bellow to know how it works:

```
<Context path="/paasmanager" docBase="war path" reloadable="true"
debug="5">
  <Resource name="jdbc/sdc" auth="Container" type="javax.sql.DataSource"
driverClassName="org.postgresql.Driver"  <!-- select the driver-->
   url="jdbc:postgresql://localhost:5432/paasmanager" <!-- select the
connection url-->
   username="postgres" password="postgres" <!-- select the user/password-->
   maxActive="20" maxIdle="10" maxWait="-1"/>
</Context>
```

You also have to add the provided scripts found in the dist file (in folder `/opt/sdc/scripts/`) in the same folder (or everywhere you want if you prefer to change the default configuration).

Include the library postgresql-8.4-702.jdbc4.jar in `$CATALINA_HOME/lib`

Create directory `/opt/paasmanager/templates` and copy the following files. These files can be found at in the PaaS Manager distribution

- empty.ovf

- InstantiateVDCTemplate.xml
- EnvelopeTemplate.xml

3.3.3. Execution Instructions

In order to start the PaaS Manager application, it is only required to start the Tomcat where the Paas Manager is located. That is,

```
$CATALINA_HOME/bin/catalina.sh start
```

For stopping it:

```
$CATALINA_HOME/bin/catalina.sh stop
```

In order to check that the PaaS Manager has been deployed, it is possible to do a quick test to check that everything is up and running. It involves obtaining the product information stored in the catalogue. With it, we test that the service is running and the database correctly configured.

```
http://{IP VM SDC}:{port}/paasmanager
```

The request to test it in the testbed should be

```
curl -v -H "Access-Control-Request-Method: GET" -H "Content-Type:
application xml" -H "Accept: application/xml" -X GET "
http://130.206.80.112:8080/paasmanager
```

3.4. Service Manager

The Service Manager (Claudia) is an advanced service management toolkit that allows service providers to dynamically control the service provisioning and scalability in an IaaS Cloud. Claudia manages services as a whole, controlling the configuration of multiple VM components, virtual networks and storage support, also optimizing their utilization and dynamically scaling up/down services applying elasticity rules, SLAs and business rules. See D4.2.2 [1] for more details.

Claudia can deploy services in both OpenNebula and Flexiscale through the Aggregated API, which is implemented by TCloud. Thus, the main components in Claudia (already explained in D4.2.2 [1]) are:

- Clotho which is the service lifecycle manager and scalability and optimization
- TCloud server which implements the API to access Claudia and the aggregated API
- Claudia-client which is the client for accessing to Claudia

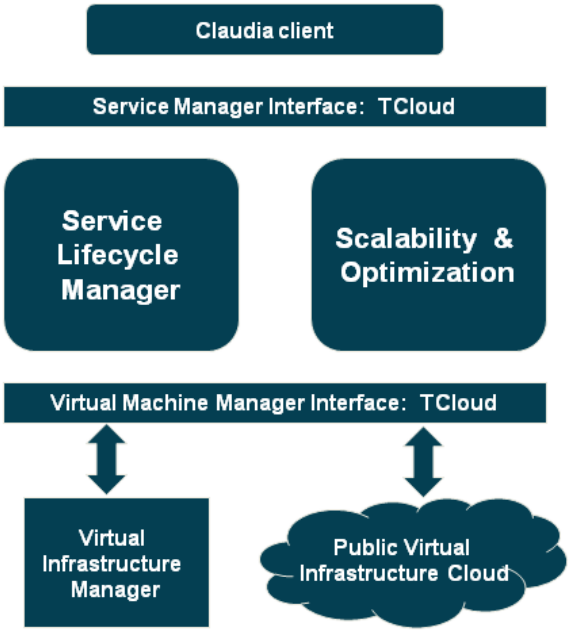

Figure 3. Service Manager Architecture.

3.4.1. Compilation Instructions

The Claudia repository is

`https://svn.forge.morfeo-project.org/4caast/trunk/WP4/claudia.`

You can access to it with:

```
svn co https://svn.forge.morfeo-project.org/4caast/trunk/WP4/claudia
```

To compile and generate rpm maven packages corresponding to the main Claudia components (clotho-rpm, tcloud-server-rpm and claudia-client-rpm)

```
mvn clean install
```

If we want to update the rpm packages into a RPM repository, we can modify the pom.xml and do

```
mvn deploy
```

Once we have the packages in a repository we can install the packages.

3.4.2. Installation Instructions

3.4.2.1. RPM packages

To install the Claudia, you must install the following rpm packages:

- clotho-rpm: the main component in Claudia.
- tcloud-server-.rpm: the APIs for Claudia (both SMI and VMI).
- claudia-client-rpm: the command line to access to Claudia.

Claudia runs its own YUM repository, so you must add it to your YUM configuration. Drop a file (named say claudia-releases.repo) in the `/etc/yum.repos.d/` with the following content:

```
[Claudia-Releases]
```

```
name=Claudia-Releases
baseurl= url
gpgcheck=0
enabled=1
```

With this in place, you can now install the packages:

```
$ yum install clotho-rpm
$ yum install tcloud-server-rpm
$ yum install claudia-client-rpm
$ yum install activemq-rpm
```

3.4.2.2. Configuration instructions

In order to configure Claudia, we need to configure different properties files for the different components.

3.4.2.3. sm.properties file (clotho configuration)

It is in /`opt/claudia/conf/sm.properties`

```
SiteRoot=grnet #Site root is the organization name for Fully Qualified
Names.

SMIHost=84.21.173.142 #IP where the Claudia server is deployed
SMIPort=8182 # The port where the Claudia server is deployed

VEEMHost=84.21.173.142  #IP where the aggregated API is deployed
VEEMPort=8182 #port where the aggregated API is deployed
VEEMPath=/ #port for the API

MonitoringAddress=84.21.173.142 #IP where Claudia is deployed (it is needed
to configure probes)

monitoring=true #Enabling monitoring
monitoringurl=http://84.21.173.132:4453/datasource-manager/datasource
#Monitoring URL
monitoringclass=com.telefonica.claudia.slm.paas.vmiHandler.NUBAMonitoringCl
ient #Monitoring class
```

3.4.2.4. tcloud.properties file (tcloud-server configuration)

It is in `/opt/claudia/conf/tcloud.properties`.

TCloud uses drivers to manage different services, more specifically for a) provisioning (IaaS provider); b) deployment (Claudia); and c) monitoring.

```
com.telefonica.claudia.smi.drivers.deployment=com.telefonica.claudia.smi.de
ployment.SMDeploymentDriver
com.telefonica.claudia.smi.drivers.provisioning=com.telefonica.claudia.smi.
provisioning.ONEProvisioningDriver
com.telefonica.claudia.smi.drivers.monitoring=com.telefonica.claudia.smi.mo
nitoring.OneMonitoringDriver
```

TCloud is in charge of the communication with the Cloud providers through the TClouddriver. We have different drivers for each IaaS provider:

For OpenNebula, we can set the ONE driver configuration: URL of the ONE driver frontend, user, encrypted password and network bridge.

```
# ONE Driver Configuration Section
oneUrl= http://localhost:2633/RPC2
oneUser=oneadmin
onePassword=7bc8559a8fe509e680562b85c337f170956fcb06
oneEnvironmentPath=/opt/claudia/repository/
oneNetworkBridge=br0
```

For Flexiant driver, it is required to set the user, password and address

```
flexiscaleUser=XXX
flexiscalePassword=XXX
flexiscaleAddresss=https://api2.flexiscale.com/index.php
```

3.4.2.5. Claudia-client properties file (Claudia-client configuration)

It is in `/opt/claudia/conf/claudia-client.properties`.

```
domain.root= grnet #Site root is the organization name for FQNs;
same as in sm.properties

#Information about the Claudia API
smi.host=http://84.21.173.142:
smi.port=8182
```

3.4.3.Execution Instructions

In order to start the clotho and tcloud services, the rpm installation installs two daemons in `/etc/init.d`, tcloudd, clothod. They can be started with

```
/etc/init.d/clothod start
/etc/init.d/tcloudd start
```

To stop them we can use

```
/etc/init.d/clothod stop
/etc/init.d/tcloudd stop
```

Besides in `/opt/Claudia/bin`, there is a script `claudiastartup.sh` which allows restarting the service.

In order to start Claudia, we have to execute the `ClaudiaC` command. This is a shell command which allows for deploying, undeploying and obtaining information of the service.

```
/opt/claudia/bin/ClaudiaC
deploy(customername,servicename,ovfpath)
info (customername,servicename)
undeploy(customername,servicename)
```

where `ovfpath` is the URL or the path where the OVF is located.

3.5. REC Manager

The REC Manager is responsible for the lifecycle management of the SW stack of the RECs, i.e. for PICs and ACs. Please note that the basic REC lifecycle (VM

deployment/start/stop/delete etc.) is managed by the Service Manager (see Section 3.4). The REC Manager receives the description of the required SW stack for each REC to deploy from the Service Manager and translates that into a concrete SW stack deployment.

The main components of the REC Manager are already explained in D4.2.2 [1] (see section 5.5.2). Thus, we provide only a short summary of the REC concepts used in this document so it will be easier to understand the configurations explained in this section.

3.5.1. Basic REC Concepts

The REC Manager is implemented by using a deployment agent installed in the REC. The key elements of the architecture are:

- in the REC:
  - Chef-Agent: responsible for executing PIC and AC related operations (install, etc.) locally.
- In the 4CaasT platform:
  - Chef Server: control all VMs (RECs) belonging to an application
  - REC Manager Core
    - Receives commands from the Resource Manager (Claudia)
    - Translates them to Chef Server commands
    - Maintains an internal model of relevant entities (e.g. PICs, ACs)

The order of the actual management steps for PIC and AC installation/configuration/etc will be determined by the Service Manager. The REC Manager will receive management commands from the Service Manager via its internal API and convert these commands to Chef Server actions via the Chef Server API.

3.5.2. REC Manager Core

3.5.2.1. Compilation Instructions

The REC Manager Core repository can be downloaded by svn with:

```
svn co https://svn.forge.morfeo-project.org/4caast/trunk/WP4/rec-manager-core/
```

Rec-manager-core project is a collection of sub-modules:

- *ovfmodel* – an EMF-based metamodel library for handling OVF documents from Java
- *rec-manager-model* – an EMF-based representation of *rec-manager*'s internal state
- *ovfserver* – the implementation of the rec-manager orchestration logic and rec-manager Rest-based API

The projects are Maven-based, to compile those you need to issue these commands:

```
mvn -DdownloadJavadocs=true -DdownloadSources=true -Dmaven.test.skip=true clean

mvn -DdownloadJavadocs=true -DdownloadSources=true -Dmaven.test.skip=true install
```

which will clean and compile all sub-modules of rec-manager-core. The resulting application is a WAR file found at:

```
rec-manager-core/ovfserver/target/ovfserver-<version>.war
```

There is an issue with the 3rd party libraries we depend on (*jclouds-chef* for accessing the Chef-server API from Java, and *gson* which is Google's JSoN Java library), which requires a specific order of class loading that conflicts with Tomcat's own mechanism for class loading (that uses alphabetical ordering) when loading the libraries packaged in a WAR archive. This issue is expected to be fixed in jclouds-chef, until then we need to ensure the correct load ordering by renaming gson.jar to z-gson.jar with the following commands:

```
      cd ovfserver/target
mv ovfserver-0.0.1-SNAPSHOT/WEB-INF/lib/gson-1.7.1.jar ovfserver-0.0.1-
SNAPSHOT/WEB-INF/lib/z-gson-1.7.1.jar
zip -d ovfserver-0.0.1-SNAPSHOT.war WEB-INF/lib/gson-1.7.1.jar
cd ovfserver-0.0.1-SNAPSHOT/
zip ../ovfserver-0.0.1-SNAPSHOT.war WEB-INF/lib/z-gson-1.7.1.jar
cd ..
```

3.5.2.2. Installation Instructions

The software was run on an Ubuntu 10.04 (specifically the Ubuntu 10.04 LTS 64bit base image found on Flexiscale infrastructure) system but it should work on later versions and other Linux systems as well.

3.5.2.2.1. Preconditions

Ensure that the following conditions are fulfilled on your system:

- OS is Ubuntu 10.04 (others may also be suitable, only 10.04 is tested)
- Tomcat6 and its prerequisites are installed

3.5.2.2.2. Installation

Copy the ovfserver WAR archive to Tomcat's webapps directory (removing the version from the file name, for correct URL mapping), and start Tomcat. Tomcat will auto-deploy the WAR application at boot time.

3.5.2.3. Configuration

The following configuration steps should take place.

Firstly an API client key has to be created in Chef-Server (see also 3.5.3.2) for rec-manager, for example by using knife. Make sure the client has admin rights (needed as rec-manager will be registering new nodes).

```
knife client create -a  -f rec-manager-client.pem rec-manager-client
```

Two key files will need to be copied from the Chef-Server to the rec-manager:

- `validation.pem` – a key used by new client nodes when they want to request registration into a Chef-Server themselves. Rec-manager will deploy this key onto the client VMs.
- `rec-manager-client.pem` – the admin key used by rec-manager

There is a central configuration file for rec-manager, found at: webapps/ovfserver/WEB-INF/ovfserver.properties, having the following options:

- `com.nsn.fourcaast.ovfserver.persistmodel` – local directory for storing the EMF state models of REC Manager (for future use, this feature is not tested yet)
- `com.nsn.fourcaast.ovfserver.chef.serveraddress` – IP address or hostname of Chef-Server

- `com.nsn.fourcaast.ovfserver.chef.validatorname` – client name for Chef-Server validator key, usually `chef-validator`
- `com.nsn.fourcaast.ovfserver.chef.validatorkey` – path of validator pem key file in the local filesystem.
- `com.nsn.fourcaast.ovfserver.chef.clientname` – client name of the admin API client created in the above step
- `com.nsn.fourcaast.ovfserver.chef.clientkey` – path of admin API client key in the local filesystem, created in the above step
- `com.nsn.fourcaast.ovfserver.url` – the URL where the rec-manager will be accessible from the VMs. This will be passed to the VMs, for directly reporting back the status to the rec-manager.

3.5.2.4. Execution Instructions

3.5.2.4.1. Starting, Stopping and Logging

The REC Manager runs as a regular web application in Tomcat, so its lifecycle is handled by the tomcat system service:

```
sudo invoke-rc.d tomcat6 stop
```

```
sudo invoke-rc.d tomcat6 start
```

The log output from the rec-manager can be found in:

```
/var/lib/tomcat6/logs/catalina.out
```

3.5.2.4.2. Using the API from command line

Note that normally the REC manager receives all commands from the Service Manager via its REST-based API. For testing purposes, though, `curl` can be used to send commands to the REC Manager and retrieve results.

Example for listing the current jobs:

```
curl http://localhost:8080/ovfserver/jobs -H "Content-Type:
application/xml" -X GET -w '\n HTTP code: %{http_code} \n ContentType:
%{content_type}\n'
```

Example for posting a new application:

```
curl http://localhost:8080/ovfserver/applications -d "$(cat
test.ovfenvelope)" -H "Content-Type: application/ovf+xml" -X POST -w '
CODE: %{http_code} \n'
```

3.5.3. Chef Server

A virtual machine image with a pre-installed Chef Server instance can be found on the testbed: *4CaaSt-Chef-Server*[6].

[6] Due to the storage requirements, the image is only available in the testbed.

The following instructions are listed as a reference for re-building an equivalent installation.

3.5.3.1. Installation Instructions

Preparing a Chef-Server component to use with the REC Manager is basically done by following the Chef's official installation instructions, which use Debian (deb) packages:

http://wiki.opscode.com/display/chef/Package+Installation+on+Debian+and+Ubuntu

Rather than installing from Ruby GEM repository, we preferred installing deb packages from an APT repository. Care must be taken not to use the outdated Chef-Server package found in the official Ubuntu 10.04 repository.

To add Opscode's Chef repository the following shell commands must be issued:

```
echo "deb http://apt.opscode.com/ lucid-0.10 main" | sudo tee /etc/apt/sources.list.d/opscode.list
```

```
wget -qO - http://apt.opscode.com/packages@opscode.com.gpg.key | sudo apt-key add –
```

```
sudo apt-get update
```

Install chef common package:

```
sudo apt-get install chef
```

When interactively asked by defconf for chef server URL, leave the default: `http://chef.example.com:4000`, this can be changed later.

Install chef-server package:

```
sudo apt-get install chef-server
```

Specify and take note of the RabbitMQ and chef-webui password when interactively asked by defconf. Later they can be changed at the following places:

RabbitMQ password is set in `/etc/chef/{server.rb,solr.rb}` files, `amqp_password` entry. After changing also execute: `sudo rabbitmqctl change_password chef` <new password> for the new configuration to take effect.

WebUI password is set in `/etc/chef/webui.rb` under the `web_ui_admin_default_password` entry.

3.5.3.2. Configuration

The *knife* command line tool needs to be set up, to administrate the Chef-Server. Knife is necessary for creating keys for API clients, such as the REC Manager.

```
mkdir -p /home/ubuntu/.chef
```

```
sudo cp /etc/chef/validation.pem /etc/chef/webui.pem /home/ubuntu/.chef/
```

```
sudo chown -R ubuntu ~/.chef
```

```
knife configure -i
```

Follow the instructions and answer the question prompts, simply press return if the default (shown in [ ] brackets) is acceptable. Here you have an example of the questions issued by the knife command and some possible answers:

```
Where should I put the config file? [~/.chef/knife.rb]
```

```
Please enter the chef server URL:[http://ubuntu.amb.flexiant.com:4000] http://localhost:4000
```

```
Please enter a clientname for the new client:[ubuntu]
```

```
Please enter the existing admin clientname:[chef-webui]
```

```
Please enter the location of the existing admin client's private key:
[/etc/chef/webui.pem] .chef/webui.pem

Please enter the validation clientname:[chef-validator]

Please enter the location of the validation key: [/etc/chef/validation.pem]
/home/ubuntu/.chef/validator.pem

Please enter the path to a chef repository (or leave blank):

Creating initial API user...

Created client[ubuntu]

Configuration file written to /home/ubuntu/.chef/knife.rb
```

After the configuration is complete knife will be able to query and modify resources on the Chef-Server via its REST-based API. Verify this by listing the clients:

```
knife client list
```

There is also a Web user interface available, which can be accessed via the following URL, by using the password specified at installation:

```
http://localhost:4040
```

3.5.3.3. Execution Instructions

The following services are needed for the correct operation of the Chef Server:

- rabbitmq-server
- couchdb
- chef-solr
- chef-expander
- chef-server
- optionally: chef-server-webui

These services are all installed automatically as dependencies to the Chef-Server package. In case a service needs to be started or stopped, the standard commands apply, for example:

```
sudo invoke-rc.d chef-server start

sudo invoke-rc.d chef-server stop
```

3.5.4. Chef-Agent

The Chef-Agent is preinstalled and preconfigured in the following VM image: *4CaaSt-Chef-Client-v02.* The Service Manager will create all RECs using this image and the REC Manager will automatically execute any necessary configuration steps remotely before actually starting the agent. Thus, normally there is no need to manually install and configure the Chef-Client.

The following instructions are listed as a reference for re-building an equivalent installation.

3.5.4.1. Installation and configuration instructions

Preparing a Chef-Client component for use with REC Manager basically follows the official installation instructions:

```
http://wiki.opscode.com/display/chef/Package+Installation+on+Debian+and+Ubu
ntu
```

Rather than installing from Ruby GEM repository, we preferred installing deb packages from an APT repository. Care must be taken not to use the outdated Chef-Client package found in the official Ubuntu 10.04 repository.

Add Opscode's Chef repository:

```
echo "deb http://apt.opscode.com/ lucid-0.10 main" | sudo tee /etc/apt/sources.list.d/opscode.list
```

```
wget -qO - http://apt.opscode.com/packages@opscode.com.gpg.key | sudo apt-key add –
```

```
sudo apt-get update
```

Install Chef-Client package:

```
sudo apt-get install chef
```

When interactively asked by defconf for the chef server URL, leave the default: `http://chef.example.com:4000` (this can be changed later in `/etc/chef/client.rb`, under the `chef_server_url` entry). It is not necessary to do this manually, REC Manager automatically changes the Chef Server URL to the correct URL whenever a REC is deployed.

The operating system needs two minor configuration changes to allow the REC Manager to configure the VM via SSH.

In `/etc/sudoers`, make sure *requiretty* option is disabled by default, which is needed to execute commands via SSH in programmatic, non-interactive fashion:

```
Defaults   !lecture,tty_tickets,!fqdn,!requiretty
```

Add the *NOPASSWD* option to the admin group permission, in order to allow programmatic execution of command as a root user, without interactive password prompts:

```
%admin ALL=(ALL) NOPASSWD: ALL
```

3.5.4.2. Execution Instructions

Updating the configuration by forcefully running Chef-Client (e.g. for debugging chef recipes) can be manually triggered by running:

```
sudo chef-client
```

The Chef-Client application normally runs as a periodic job, which is controlled by a standard OS level service using the standard commands:

```
sudo invoke-rc.d chef-client start
```

```
sudo invoke-rc.d chef-client stop
```

Running the Chef-Client services is unnecessary in case the VM is being deployed via the REC Manager, as Chef-Client is forcefully invoked (via SSH) whenever a configuration change needs to be applied. It's recommended to disable the Chef-Client service via:

```
sudo update-rc.d chef-client disable
```

3.6. OpenNebula Clusters

Clusters are pools of hosts that share datastores and virtual networks. Clusters are used for load balancing, high availability, and high performance computing.

A Host is a server that has the ability to run Virtual Machines and that is connected to OpenNebula's Frontend server. OpenNebula can work with Hosts with a heterogeneous

configuration, i.e. you can connect Hosts to the same OpenNebula with different hypervisors or Linux distributions as long as these requirements are fulfilled:

- Every Host need to have an oneadmin account.
- OpenNebula's Frontend and all the Hosts need to be able to resolve, either by DNS or by /etc/hosts the names of all the other Hosts and Frontend.
- The oneadmin account in any Host or the Frontend should be able to ssh passwordlessly to any other Host or Frontend. This is achieved either by sharing the $HOME of oneadmin across all the servers with NFS or by manually copying the ~/.ssh directory.
- It needs to have a hypervisor supported by OpenNebula installed and properly configured. The correct way to achieve this is to follow the specific guide for each hypervisor.
- ruby >= 1.8.5

Here is an example of adding a host to a specific cluster:

```
$ onehost list
ID NAME         CLUSTER     RVM   TCPU   FCPU   ACPU   TMEM   FMEM   AMEM STAT
0 host01       -             7    400    290    400   3.7G   2.2G   3.7G   on

$ onecluster addhost production host01


$ onehost list
ID NAME         CLUSTER     RVM   TCPU   FCPU   ACPU   TMEM   FMEM   AMEM STAT
0 host01       producti      7    400    290    400   3.7G   2.2G   3.7G   on

$ onecluster show production
CLUSTER 100 INFORMATION
ID             : 100
NAME           : production
HOSTS
0
VNETS
DATASTORES
```

For more information on managing OpenNebula Clusters, see [7]. Besides the command line interface, clusters can be also managed using the Sunstone web interface [8].

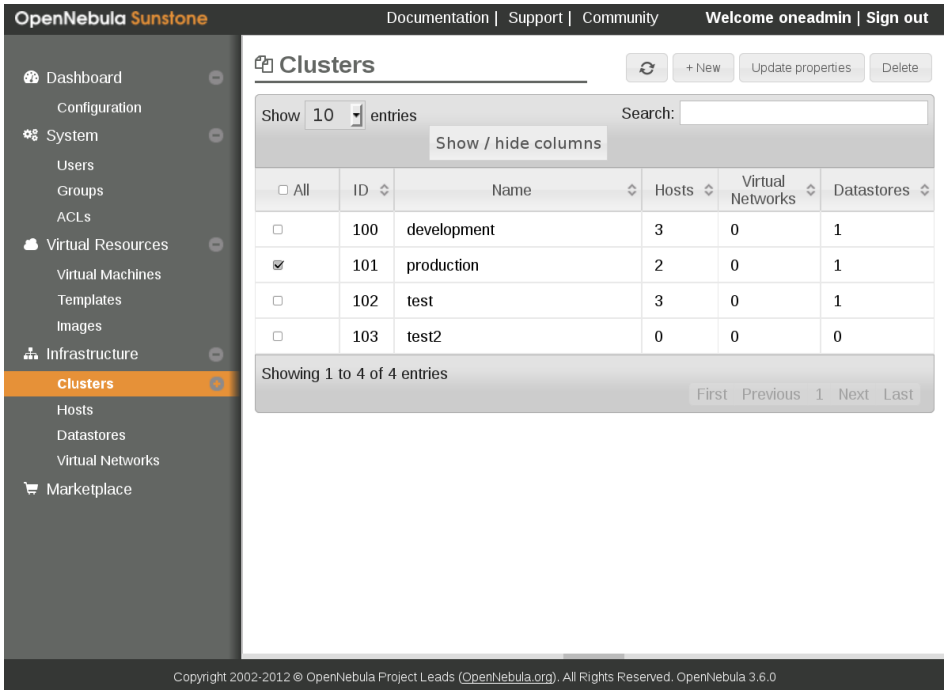

Figure 4. Managing OpenNebula Clusters with the Sunstone web UI.

4. User guide

Most 4CaaSt WP4 components belong to the backend layers of the 4CaaSt architecture and therefore do not expose specific Graphical User Interfaces (UI) to be described in this section. Only the Deployment Design Manager offers a specific UI, which is explained in the following.

4.1. Deployment Design Manager UI

The Deployment Design Manager offers a web-based UI using SAP's UI5 technology and supports the following two use cases:

a) Inspect generated deployment designs based on the data available in the repository (Deployment Browser)

b) Test the component (Debug Interface).

Please navigate to the following URL to start the UI:

`http://<server-ip>:8080/com.sap.deploymentmanager.frontend/`

4.1.1. Deployment Browser

The start screen of the web app represents the Deployment Browser (Figure 5). It contains a list of all past deployments. Using the drop-down box, the selection can be narrowed down to certain periods of time, e.g. past day, week, month.

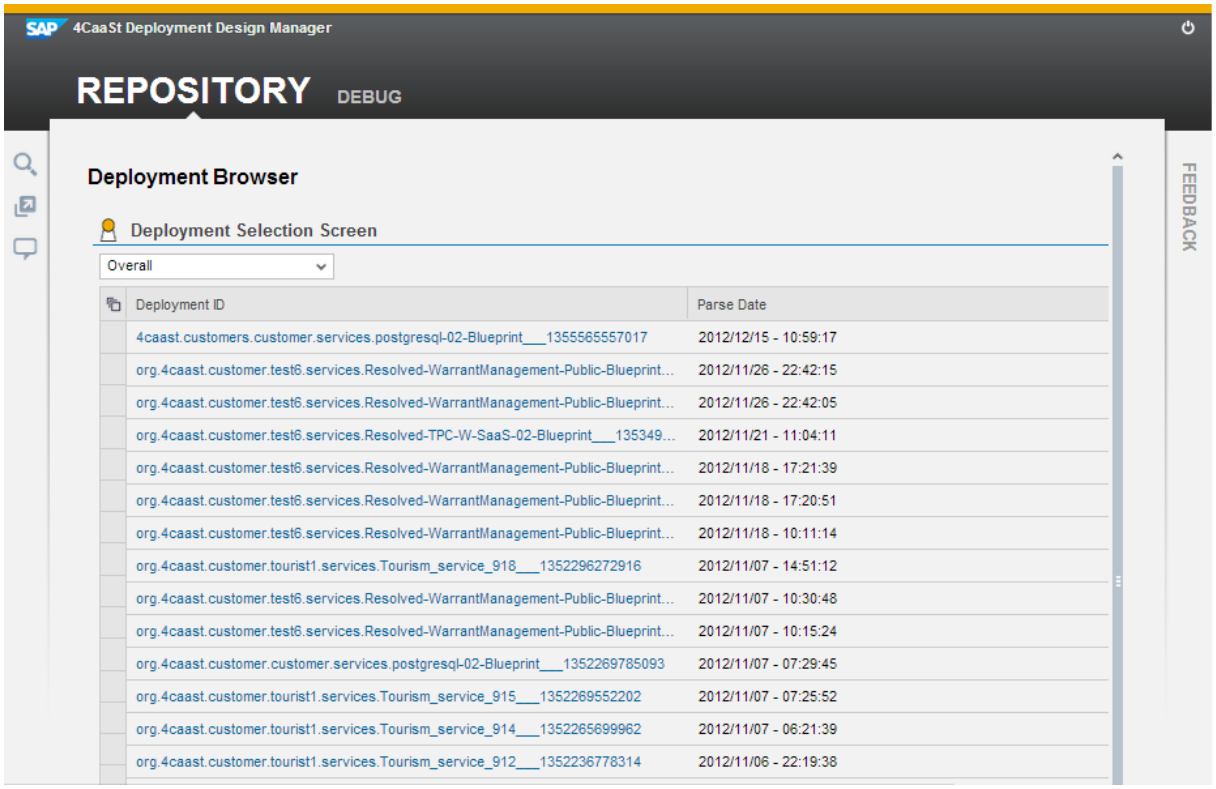

Figure 5. Deployment Browser

Clicking on a deployment will open a pop-up window showing the details of this particular deployment:

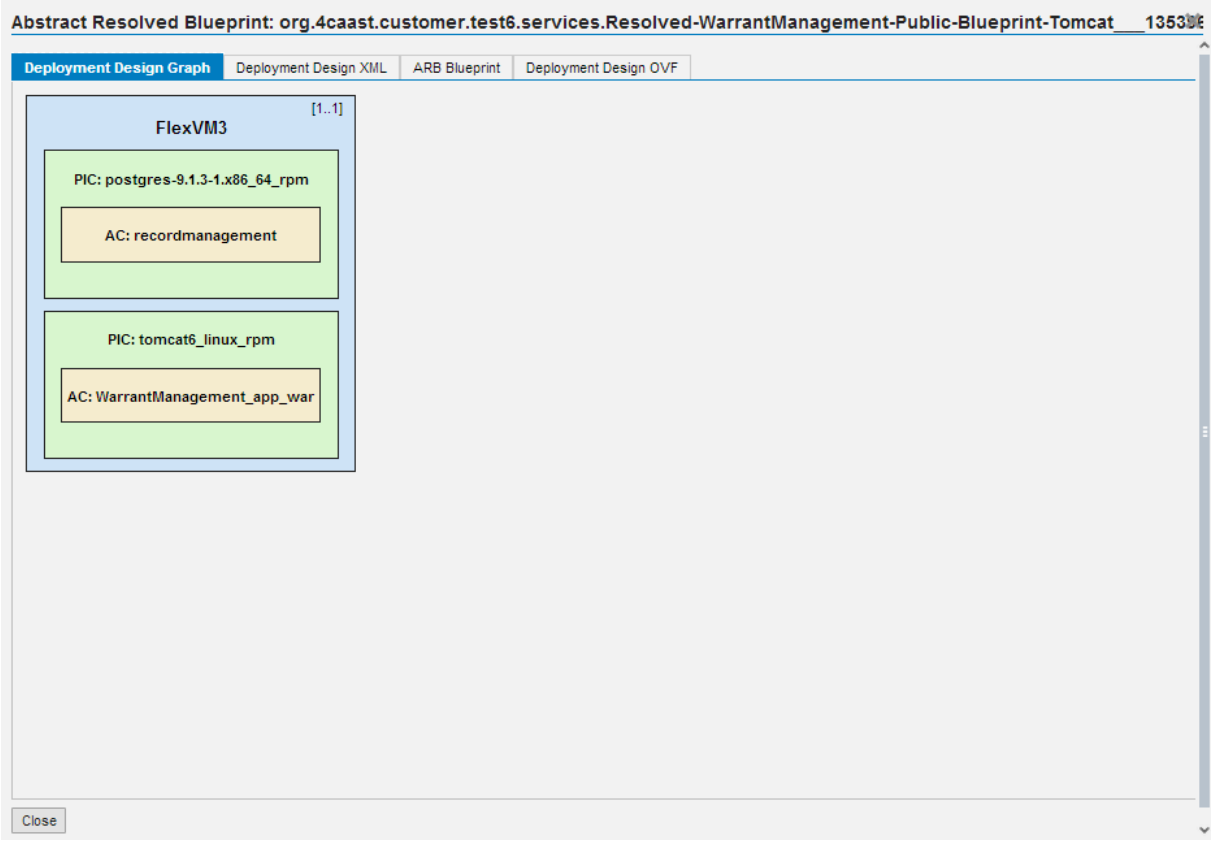

Figure 6. Deployment Design Graph

The details window includes four tabs. The deployment design graph tab you see above (Figure 6) provides a graphical representation of the generated deployment design, whereas the other tabs show the corresponding xml files:

- Deployment Design XML (Figure 7): Internal specification of the deployment design (EMF-based) as also used by the Resource Requirements Calculator.
- ARB Blueprint (Figure 8): The original blueprint from which the design was generated.
- Deployment Design OVF (Figure 9): The generated executable design based on OVF++, which is sent to the PaaS Manager.

The following screenshots show examples for the three tabs:

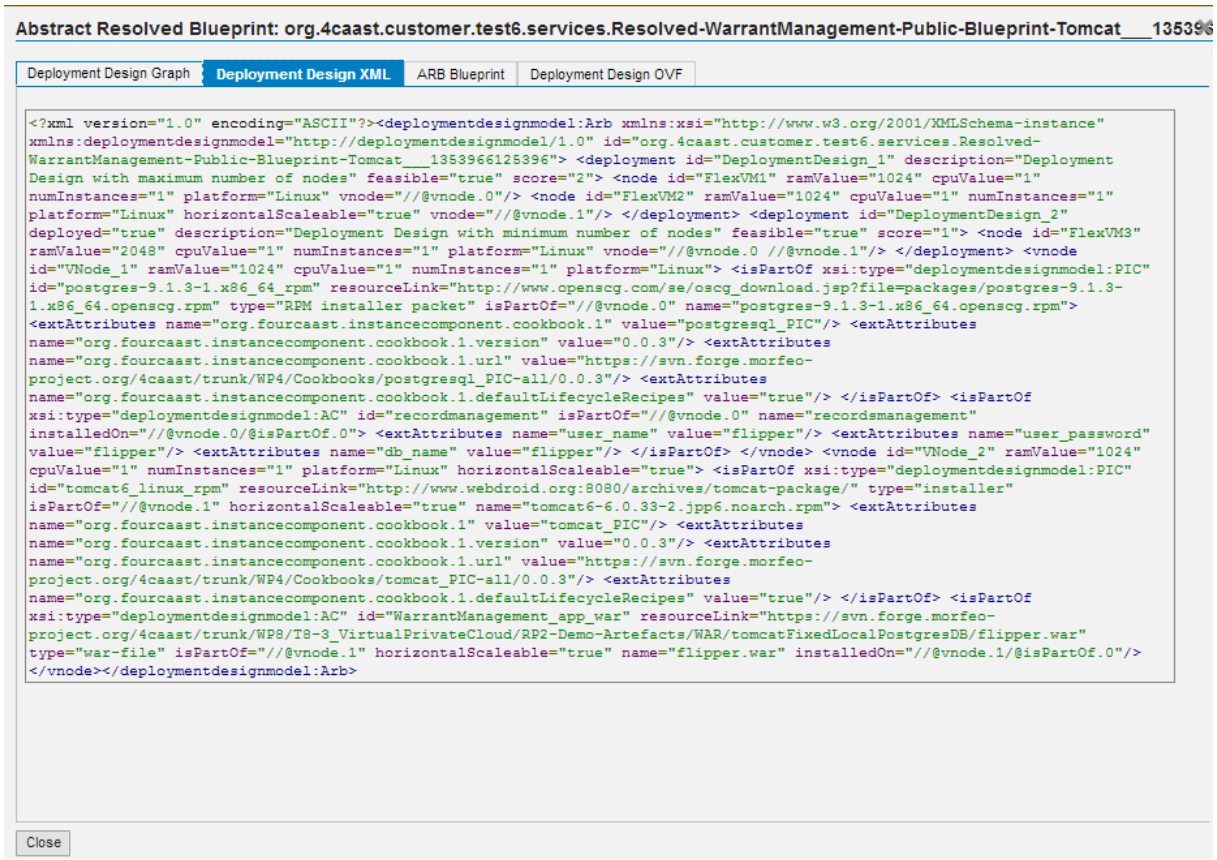

Figure 7. Deployment Design XML

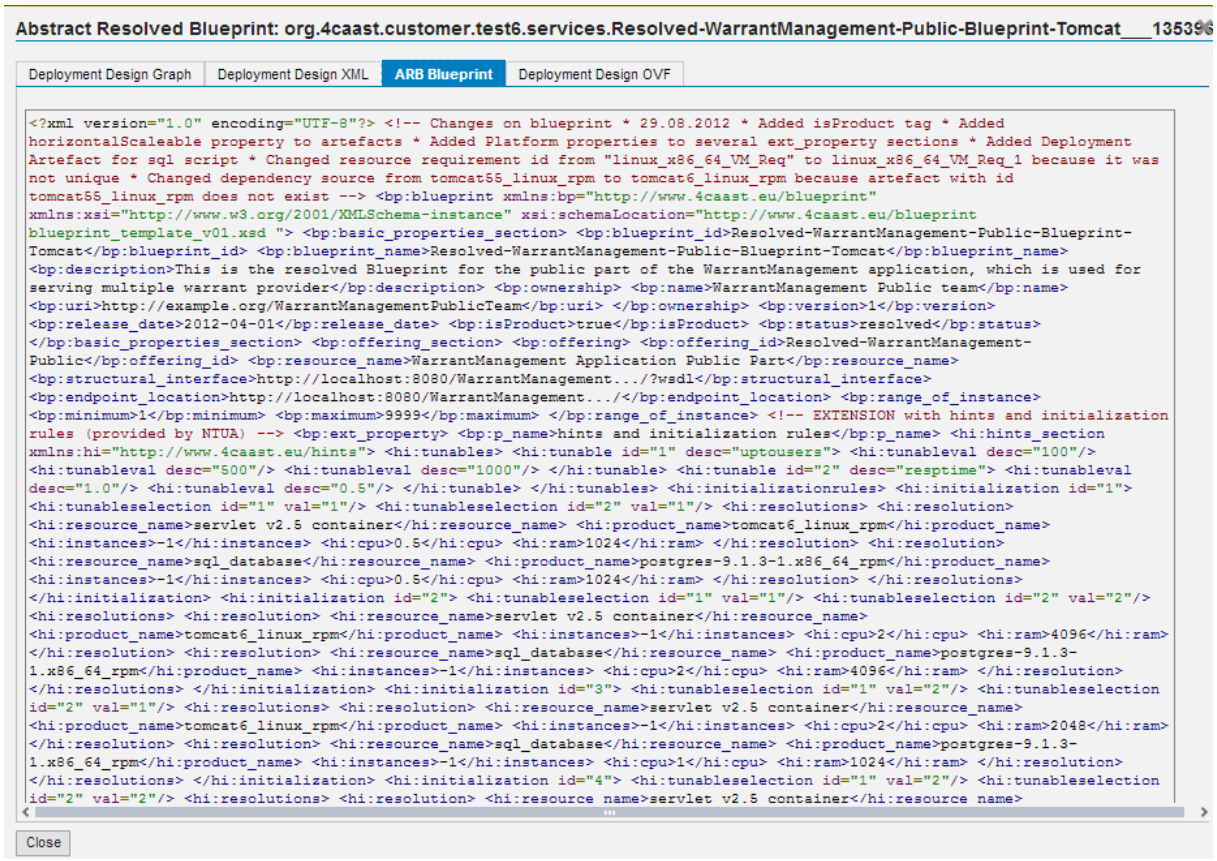

Figure 8. Original Blueprint

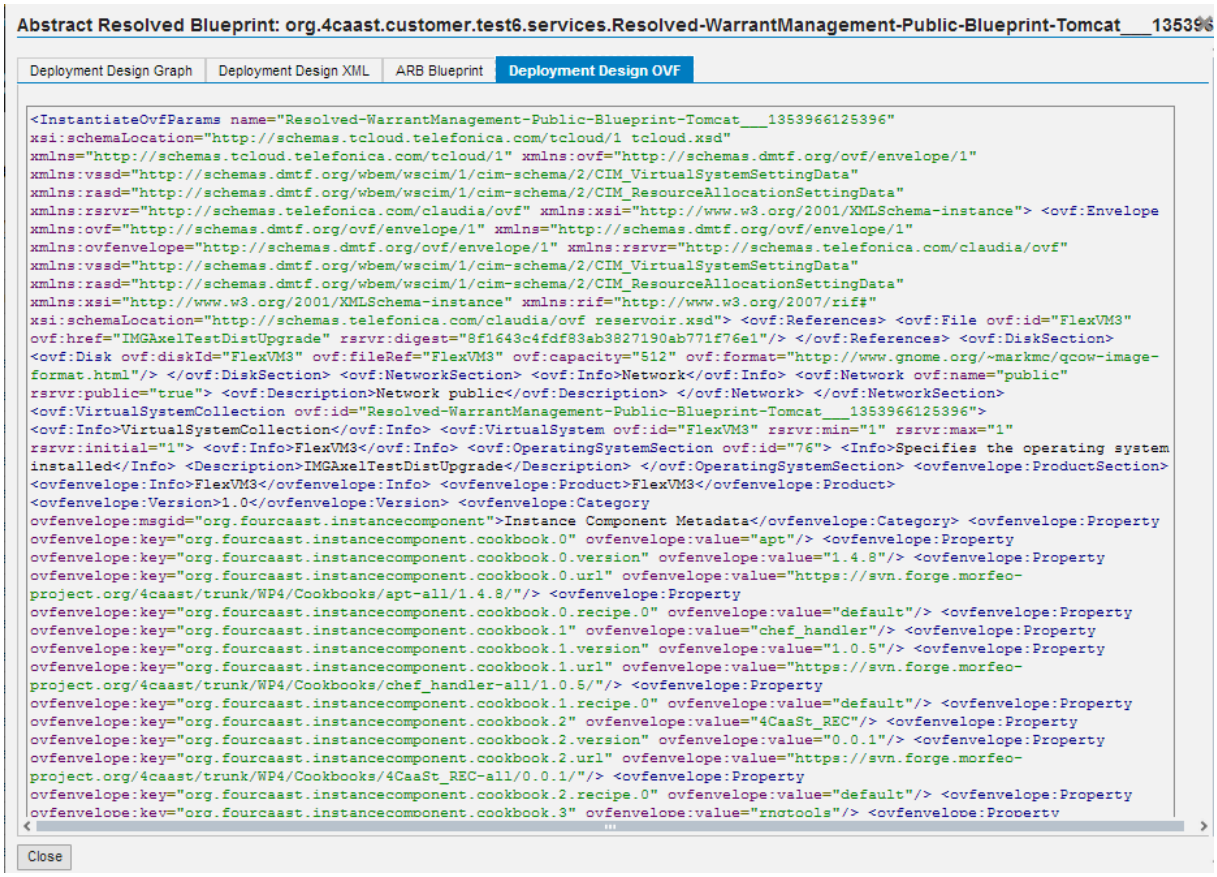

Figure 9. Generated OVF++

4.1.2. Debug Interface

In addition to the deployment browser the UI offers a simple debug interface (Figure 10).

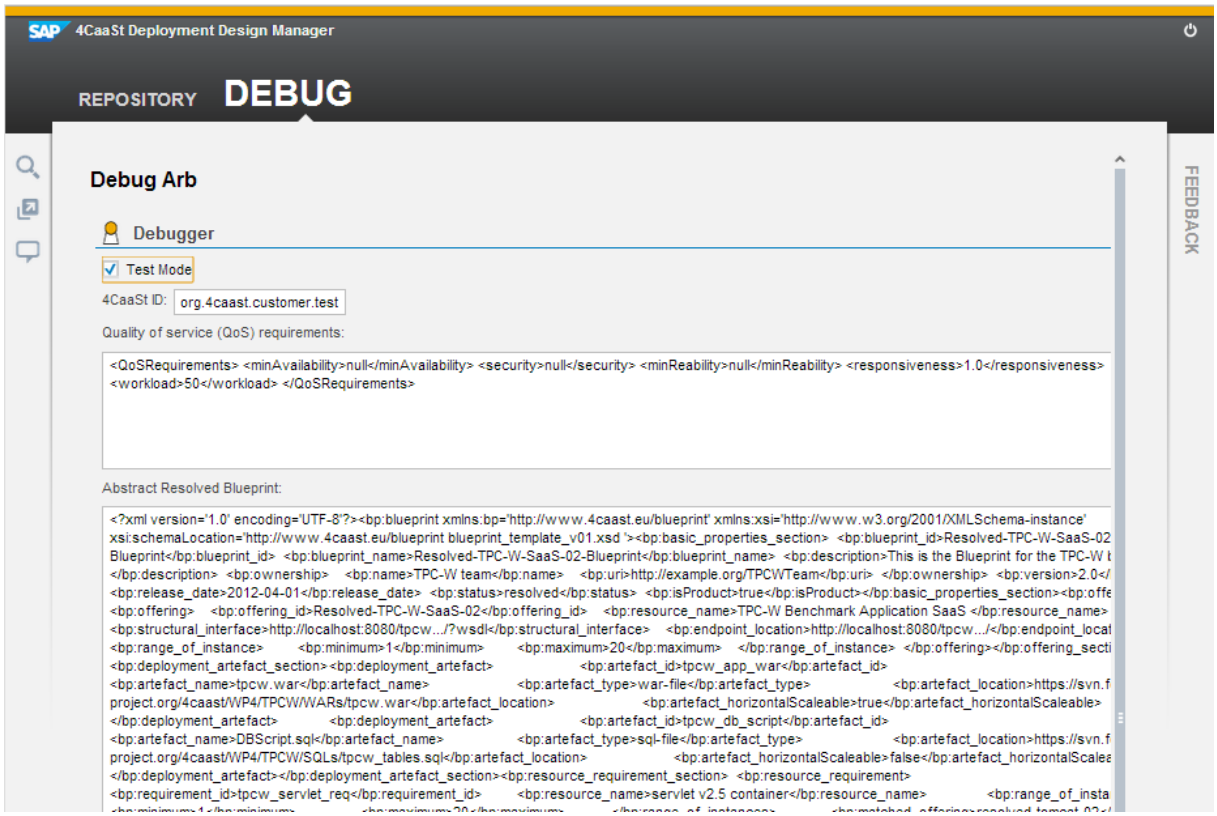

Figure 10. Debug Interface

To test the component you need to define the 4CaaSt ID, QoS requirements and the corresponding ARB. Sample data can be generated by pressing the “Load Sample Data” button at the lower end (see Figure 11). Pressing the “Send ARB to Deployment Manager” button then triggers the deploy method of the Deployment Design Manager with the specified data. If the checkbox “TestMode” is checked (Figure 10), the PaaS Manager invocation will be skipped. Otherwise, the generated design is sent to the PaaS Manager, which results in the provisioning of new resources.

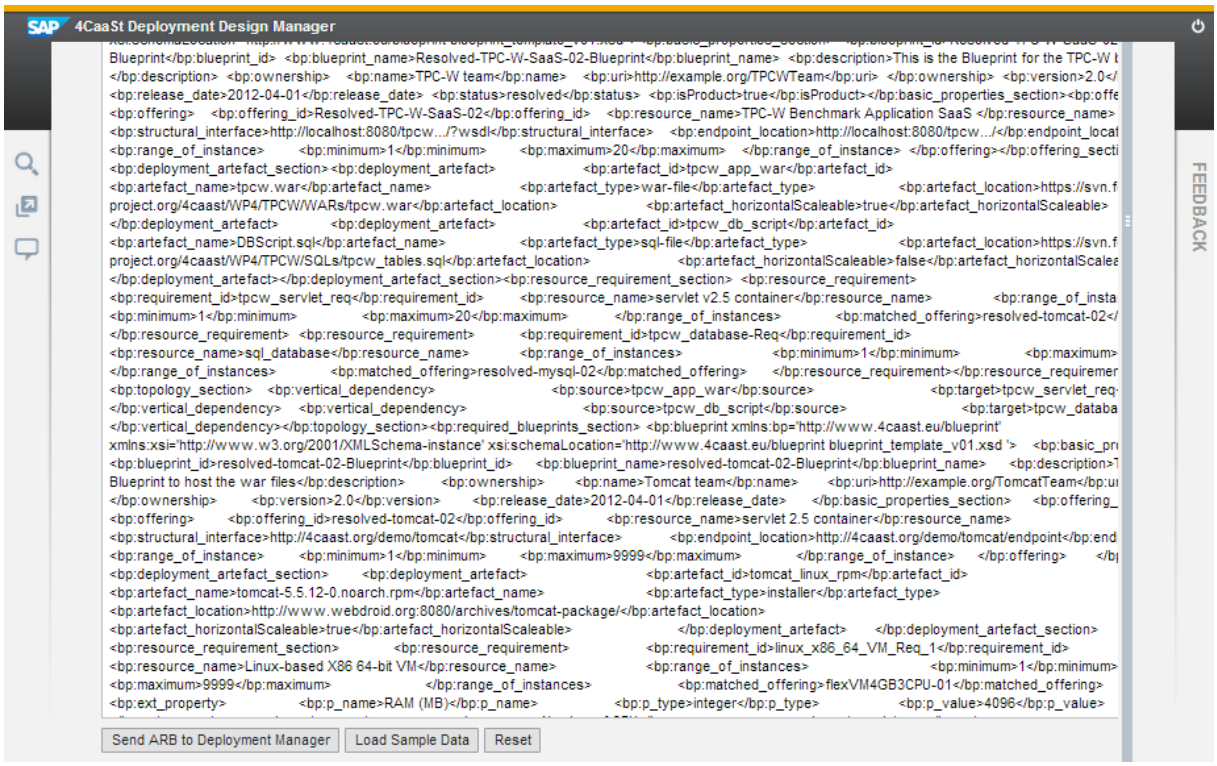

Figure 11. Generation of Sample Data

5. Conclusions

This deliverable describes the second release of the components devoted to implement the resources management layer of the 4CaaSt platform. In a nutshell, the second release implements the full deployment of an application, from the blueprint-based specification of its logical architecture and SLA constraints, to the instantiation of virtual machines, deployment and configuration of software products and components over them in order to run the application, and the monitoring setup and elasticity rules required to scale the application. Releases 3 will incorporate further refinements about applications configuration and NaaS integration with the platform and with OpenNebula.

All the components described in this document, which are part of the architecture depicted in D4.2.2 [1], are deployed over the infrastructure provided by the Flexiscale facilities and interact with that platform in order to manage the virtual machines used to deploy applications.

The instructions for the installation and configuration of the described components are explained in this document. In contrast their operation by final users has not been described since they are backend components handled by other components in higher layers of the 4CaaSt architecture.

6. References

[1] 4CaaSt Consortium. D4.2.2 Resource Deployment and Management. Components Design and Open Specification. August 2012.

[2] 4CaaSt Consortium. D4.3.1 Resource Deployment and Management. Experimental Prototype of Software Components and Documentation. August 2012.

[3] 4CaaSt Consortium. D3.2.2 Marketplace Functions. Components Design and Open Specification. August 2012.

[4] 4CaaSt Consortium. D5.2.2 Administration, Accounting, Monitoring and Analytics. Components Design and Open Specification. August 2012.

[5] 4CaaSt Consortium. D.1.3.2. Integration of results. To be published (February 2013).

[6] 4CaaSt Consortium. D.1.1.3b. Integration of results. Resubmission December 2012.

[7] OpenNebula Project. Managing Clusters 3.8. Available online `http://opennebula.org/documentation:rel3.8:cluster_guide`

[8] OpenNebula Sunstone: The Cloud Operations Center 3.8. Available online `http://opennebula.org/documentation:rel3.8:sunstone`