Grant Agreement No.: 604590
Instrument: Large scale integrating project (IP)
Call Identifier: FP7-2012-ICT-FI

eXperimental Infrastructures for the Future Internet

D3.3: Infrastructures Management Toolkit API

Revision: v.2.0

Work package	WP 3
Task	Task 3.3
Due date	31/12/2013
Submission date	6/2/2014
Deliverable lead	SYNELIXIS
Authors	Th. Zahariadis (SYN), P. Trakadas (SYN), A. Papadakis (SYN), P. Karkazis (SYN), Th. Orfanoudakis (SYN), A. Martellone (CREATE-NET)
Reviewers	A. Willner (TUB), E. Power (WIT)

Abstract	Deliverable D3.3 describes the first version of the infrastructure management tools to be utilized within the XIFI federated infrastructure. These components are the Infrastructure ToolBox (ITBox) and the Deployment and Configuration Adapter (DCA). The deliverable provides the necessary documentation of each of the components, the architectural design, the installation guide, the test cases and the user and developer guides. It also links to the source code.
Keywords	Deployment, Configuration, bare metal, Generic Enabler, FI-WARE

Document Revision History

Version	Date	Description of change	List of contributor(s)
V1.0	30.01. 2014	First complete version of the deliverable.	Th. Zahariadis (SYN), P. Trakadas (SYN), A. Papadakis (SYN), A. Voulkidis (SYN), P. Karkazis (SYN), F. Orfanoudakis (SYN), F. Facca (CREATE-NET), S. Cretti (CREATE-NET), A. Martellone (CREATE-NET)
V2.0	05.02.2014	Comments by the reviewers have been incorporated	E. Power (TSSG), D. Nehls (TUB), A. Willner (TUB)

Project co-funded by the European Commission in the 7th Framework Programme (2007-2013)		
Nature of the Deliverable:	P (Prototype)	
Dissemination Level		
PU	Public	✓
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to bodies determined by the XIFI project	
CO	Confidential to XIFI project and Commission Services	

[1] http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en_US

EXECUTIVE SUMMARY

The main objective of XIFI is the design and development of the appropriate software, tools and procedures that will address the capacity building requirements of the pan-European federated cloud infrastructure, (following and being compatible with the architecture and development activities of the FI-PPP programme, as a whole, but mostly with FI-WARE project), and thus being able to accommodate Use Case projects' requirements. In this perspective, as also stated in the DoW, Task 3.3 provides the means to "*design and develop tools to facilitate the management of the XIFI federation and the FI enablers on top of the infrastructures available at the different nodes in the federation. In order to support deployment on top of infrastructures, we will develop a toolbox that will install a distribution with all the XIFI tools needed for the single node to manage its connection to XIFI federation. Moreover, data on Generic Enablers installed on a single node will be published to the XIFI federation, through APIs, to support add-on federation services developed in WP4*".

In other words, from a technical perspective, Task 3.3 deals with the design and development of the tools and adapters that, on one hand will take care of the processes required for a new infrastructure owner (from bare-metal) to become IaaS/PaaS compatible in XIFI federation level and, on the other hand, to provide/develop tools that will provide information on federation level regarding the deployment and configuration of the Generic Enablers (GEs). The aforementioned objectives of Task 3.3 (and its accompanying deliverable) have been addressed by the introduction of two components in the XIFI architecture, namely ITBox and DCA. In brief, ITBox is an IaaS deployment tool supporting the automated installation of the main components of a node (including host operating system, hypervisor, DCRM, monitoring and networking adapters) to become compatible with the XIFI federated infrastructure. In parallel to ITBox, Task 3.3 offers DCA, that caters for the management issues related to the deployment and configuration of multiple GEs in the federated XIFI infrastructure, such as the availability of resources prior to the deployment and configuration of GEs, the design and development of a component that will store all data related to GEs deployment and configuration and provide data to respective entities/components developed within XIFI, such as the Federation Manager, Resource Catalogue and Recommendation Tool.

From a business perspective, this task (and the accompanying deliverable) provides a first version of the tools that are necessary for an infrastructure owner, in order to join the XIFI federation (from the IaaS/PaaS point of view), but also tools that will help application developers to review, compare and select GEs to deploy innovative services. The developments described herein, focused on the requirement that such processes and tools have to be as automated as possible in order to minimizing time-to-market and assist XIFI sustainability.

Being a "Prototype" in nature, this deliverable provides the source code and all the necessary accompanying documents (installation manual, user manual, developer guide, test cases) that a XIFI user (either an infrastructure owner or an application developer, as mentioned above) has to follow in order to deploy and configure the appropriate tools and GEs in the XIFI federated infrastructure. Following the common branding strategy between FI-WARE and XIFI, source code and respective documents for ITBox and DCA will become available through FI-Ops, in order to better position the results of XIFI and contribute to the overall success of FI-PPP and its recognition outside the FI-PPP.

Finally, it is important to highlight that this document structure, the text and the developments, as presented in this deliverable, have taken into account and addressed the comments received by the reviewers during the first XIFI project review.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

API: Application Programming Interface

DC: Datacentre

DCA: Deployment and Configuration Adapter

DCRM: Data Centre Resource Manager

DEM: Datacentre and Enablers Monitoring

FI-PPP: Future Internet - Private Public Partnership

GE: Generic Enabler

IaaS: Infrastructure as a Service

ITBox: Infrastructure Toolbox

NAM: Network Active Monitoring

NPM: Network Passive Monitoring

PaaS: Platform as a Service

SaaS: Software as a Service

UC: Use Case

URL: Uniform Resource Locator

1 INTRODUCTION

This deliverable provides an in-depth technical analysis and description of the first version of the ITBox and DCA components' functionality, links to their available source code as well as the necessary documentation, including the installation guide, the test cases, the user guide and developer guide.

The motivation for the development of these components stems from the following two reasons:

1. There is a lack of a modular tool that will allow for a highly automated installation and configuration of several bare-metal servers belonging to an infrastructure node willing to become part of the XIFI federation. Addressing this challenge, we are developing the Infrastructure Toolbox (ITBox) that offers IaaS FI-WARE-compatible, ready-to-use servers (including the installation of the operating system, the hypervisor, the cloud management software and XIFI monitoring and network adaptation tools) through an appropriate API.
2. Management (including installation and monitoring) of several GEs, especially in the federated XIFI landscape, must be simplified as much as possible to assist XIFI (and FI-PPP) sustainability. To this end, we are developing the Deployment and Configuration Adapter (DCA) whose main objective is to provide an adaptation layer between the XIFI portal and other components (such as PaaS, SDC, Scalability Manager) that are responsible for the deployment and configuration of the Generic Enablers (GEs).

1.1 Purpose

The purpose of this deliverable is to describe the development (a) of the ITBox that aims at simplifying the installation and configuration of appropriate tools for an infrastructure owner to become compliant with XIFI IaaS/PaaS requirements and (b) of DCA that caters for the automated deployment, configuration and management of GEs to the XIFI federated nodes that will help application providers to easily review, compare and select the proper GEs to be deployed and configured in order to support the development of innovative services.

1.2 Overview

The structure of this deliverable is as follows: the *Summary* section provides a brief description of the ITBox and DCA components. The *Motivation* section deals with the reasoning for the design and development of these components, while *State of the Art* section provides the rationale for the selection of components to be used, extended and adapted. The next section, *Architecture Design*, presents the functionality that each component provides, along with the dependencies and inter-connections to other components (either FI-WARE or XIFI). Finally, the last sections include detailed description of the *Installation Manual*, *Test Cases*, *User Manual* and *Developer Guide* that will become publicly available through FI-Ops and will help infrastructure owners and application developers to deploy and test the functionality of the respective components.

2 COMPONENTS

2.1 Infrastructure ToolBox (ITBox)

2.1.1 Summary

The ITBox supports the automated installation of the main components of a XIFI node. The download version and some configuration parameters will be provided by the portal, following the registration of the new node. ITBox distribution will include the deployment of the necessary XIFI components, tools and GEs to complete the XIFI node installation. Figure 1 shows the detailed scheme of the XIFI Reference Architecture and the location of the Infrastructure Toolbox (see yellow box).

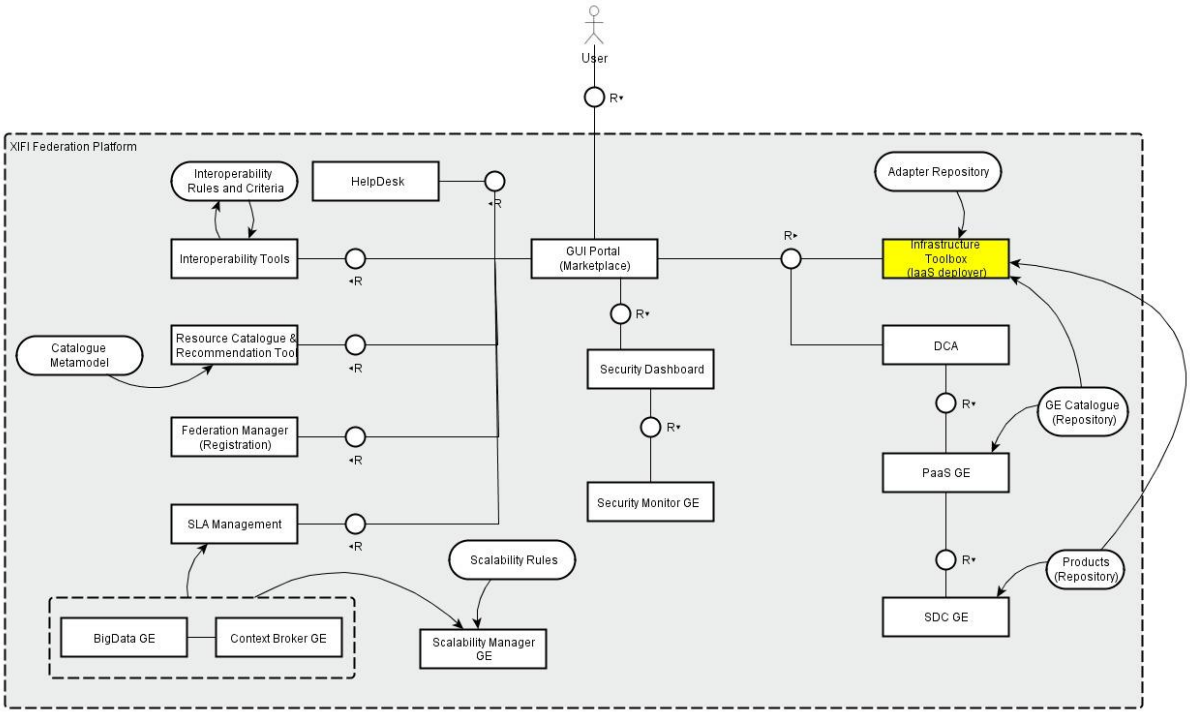

Figure 1: Location of the ITBox in the XIFI Reference Architecture

The ITBox consists of

- ITBox Interface;
- ITBox Middleware.

Depends on

- Network Active Monitoring (NAM)
- Datacentre and Enablers Monitoring (DEM)
- Network Passive Monitoring (NPM)
- OpenStack Data Collector
- Data Centre Resource Management (DCRM)

Further dependencies may be more evident with future evolutions.

Reference Scenarios	UC 1 - Joining the Federation
Reference Stakeholders	Developers: those who want to install a private XIFI node. Infrastructure owners: those who want to join XIFI federation or build a private XIFI node.
Type of ownership	Extension
Original tool	Mirantis Fuel 3.2.1
Planned OS license	Apache License Version 2.0. As the original tool.
Reference OS community	OpenStack, Fuel will be incubated there.

2.1.2 Component responsible

Developer	Email	Company
Alessandro Martellone	amartellone@create-net.org	CREATE-NET

2.1.3 Motivation

The deployment of a large distributed infrastructure is a complex task that requires automation to scale. The Infrastructure Toolbox aims at providing such automation in the context of XIFI cloud services. From the overall scenario, we derived a number of cases that represent requirements to be supported by the tool. Developments will occur according to elicited priorities (by infrastructure owners or external adopters) and available resources in the project.

Section *State of the Art* provides an overview of available open source tools, which we analysed prior to selecting Fuel as the starting point. Fuel was selected mainly for the following reasons [2]:

- It natively supports OpenStack (required to create a new FI-WARE Cloud instance)
- The graphical interface is intuitive
- It is going to be incubated in OpenStack
- It is a mature an stable solution
- It natively supports several of the configuration requirements to install DCRM, which are appropriate for the XIFI case

2.1.4 User Stories backlog

User story id	User story name	Actors	Description
1	Support Ubuntu LTS 12.04	Infrastructure Administrator (IA), Infrastructure Toolbox (ITBox)	The IA installs Ubuntu LTS 12.04 on the node through the Preboot eXecution Enviroment (PXE).
2	Include support for configuration without separate cinder node	Infrastructure Administrator (IA), Infrastructure	The IA can choose, using user interface (UI), if to install cinder on compute node or in a separate node.

		Toolbox (ITBox)	If a cinder node is not specified then the compute node acts automatically also as a cinder node. Moreover, the Infrastructure Toolbox verifies, in compliance with DCRM set-up, if the Network File System (NFS) is the current solution for cinder deployment.
3	Deploy DCRM using CLI	Infrastructure Administrator (IA), Infrastructure Toolbox (ITBox)	The IA deploy DCRM using command line interface (CLI), setting node id and the others requirement parameters. Once the installation of DCRM is successfully completed, the IA can select the OpenStack scheduler among default, Pivot, Pulsar.
4	Deploy DCRM using UI	Infrastructure Administrator (IA), Infrastructure Toolbox (ITBox)	The IA can deploy DCRM using UI. Once the installation of DCRM is successfully completed, the IA can select the OpenStack scheduler among default, Pivot, Pulsar.
5	Deploy the Identity Manager (IdM) GE and the Access Control (AC) GE using CLI	Infrastructure Administrator (IA), Infrastructure Toolbox (ITBox)	The IA decides to install, on an already provisioned node, the Identity Manager (IdM) GE and the Access Control (AC) GE using CLI. He launches deploy command, setting node id and the others requirement parameters.
6	Deploy the Identity Manager (IdM) GE and the Access Control (AC) GE using GUI	Infrastructure Administrator (IA), Infrastructure Toolbox (ITBox)	The IA selects a deployed node. The ITBox displays details about it and a list of installable GEs. The IA selects the Identity Manager (IdM) GE and the Access Control (AC) GE and next he starts the installation.
7	Deploy the Monitoring GE and adapters using CLI	Infrastructure Administrator (IA), Infrastructure Toolbox (ITBox)	The IA decides to install, on an already provisioned node, the Monitoring GE and its related adapters using CLI. He launches update command, setting node id and the others requirement parameters.
8	Deploy the Monitoring GE and adapters using GUI	Infrastructure Administrator (IA), Infrastructure Toolbox (ITBox)	The IA selects a deployed node. The ITBox displays details about it and a list of installable GEs. The IA selects the Monitoring GE and its related adapters and finally he starts the installation.
9	Send deployment test report to Federation Manager	Infrastructure Administrator (IA), Infrastructure Toolbox (ITBox)	The IA selects the current environment and runs tests on the deployed nodes. When the tests are finished, then ITBox will send a report to Federation Manager.
10	Enable Quantum using GUI	Infrastructure Administrator (IA), Infrastructure Toolbox (ITBox)	The IA selects a provisioned node. The ITBox displays details about it and a list of available options. The IA selects Quantum and he fills its parameter fields (e.g. tenant network type {gre|vlan}, segment range and so on).

11	Include deployment of DOVE as plugin for Quantum using CLI	Infrastructure Administrator (IA), Infrastructure Toolbox (ITBox)	The IA decides to install, on an already provisioned node, the DOVE plugin using CLI. He launches deploy command, setting node id and the others requirement parameters.
12	Include deployment of DOVE as plugin for Quantum using GUI	Infrastructure Administrator (IA), Infrastructure Toolbox (ITBox)	The IA selects a deployed node. The system displays details about it and the list of installable plugins. The IA selects DOVE and finally he starts the installation.
13	Include deployment of Object Storage using CLI	Infrastructure Administrator (IA), Infrastructure Toolbox (ITBox)	The IA decides to deploy a new Object Storage node into current infrastructure. He launches deploy command, setting node id and the others requirement parameters.
14	Include (when available) test scenarios from WP2 continuous integration	Infrastructure Administrator (IA), Infrastructure Toolbox (ITBox)	The IA selects the current environment and runs WP2 continuous integration tests on the deployed infrastructure. When the tests are finished, then ITBox will send a report to Federation Manager.
15	Recommender for matching servers with roles according to their capacity	Infrastructure Administrator (IA), Infrastructure Toolbox (ITBox)	The IA selects the current infrastructure. The ITBox shows the list of discovered nodes and in according to their computational capacity it suggests the best role for each node.

2.1.5 State of the Art

FI-WARE developed the SDC tool [1] to support GE installation. So far, however, there is no plan to support the deployment of IaaS DCRM and related tools in a similar manner. This is of primary importance to build the so-called Infrastructure Toolbox. A number of tools exist, to support deployment of IaaS solutions, some are OpenStack focused and other are IaaS solution independent. In order to provide a manageable and flexible toolbox, which gives the ability to manage and control the deployment of services across the datacenter, we have analysed the following products:

- Fuel: a tool focused on OpenStack, developed by Mirantis [2];
- Tuskar: an open source project tailored for OpenStack deployments [3];
- Foreman: an independent solution that covers the complete lifecycle management tool for physical and virtual servers [4];
- Rackspace Private Cloud: a tool that enables users to deploy a private cloud OpenStack cluster configured according to the recommendations of Rackspace OpenStack specialists [5].

An OpenStack cloud installation consists of many packages from different projects, each with its own requirements, installation procedures and configuration management.

The scope of these tools is to help the cloud environments administrators to organize and manage a large variety of infrastructures and services, thus to have a systematic deployment and maintenance process.

Fuel for OpenStack

Fuel uses Puppet [6], an IT automation software (it provides, through appropriate scripts, an easy way to automate repetitive tasks). It automates provisioning and deployment of all core OpenStack

components including Nova, Glance, Horizon, Swift, Keystone, Neutron/Quantum and Cinder.

In the last release 3.2.1, Mirantis provides an OpenStack distribution onto Red Hat Enterprise Linux, CentOS and Ubuntu 12.04 LTS version. Fuel is composed of Fuel Master node (where all the necessary modules are installed, such as the provisioning agent, the web user interface, the nodes discovery agent), the Puppet scripts and the OpenStack packages.

In practice, after the setup of the Fuel Master Node, the cloud infrastructure administrator can discover his virtual or physical nodes and configure his OpenStack cluster using the Fuel UI. Finally, he deploys his OpenStack cluster on discovered nodes. Fuel will perform the entire deployment process by applying pre-configured and pre-integrated Puppet manifests.

One of the advantages of Fuel is that it comes with a number of pre-built deployment configurations that can be used by users to build their OpenStack cloud infrastructure. These are well-specified configurations of OpenStack, according to the best practices recommended by OpenStack specialists [2].

Tuskar

Tuskar, similar to Fuel, provides cloud environment administrators with the ability to control OpenStack deployment across their datacentres.

With Tuskar, the administrators model their cluster of nodes into "resource classes". Every resource class is composed of a service type, more rack elements (where a rack models is a set of nodes, capacities and network properties) and one or more flavours (where a flavour is a virtual machine template).

Tuskar services are available via a RESTful API or via management console, through which administrators are able to classify their hardware and define their data centers. In addition, Tuskar provides a set of functionalities for performance monitoring, health statistics, and usage metrics. It aims to support cloud environment administrators in making the right decisions. The only limitation of Tuskar is that it is in an early development phase and it does not have a stable version. However, at the moment, the OpenStack community discusses the future integration between Fuel and Tuskar.

Foreman

Foreman is a tool, independent from OpenStack installation, which manages every lifecycle management phase of physical and virtual servers. It contains a collection of Puppet modules and it uses native open source packaging (e.g. RPM and .deb packages). A Foreman installation, similarly to Fuel, contains a central Foreman instance which is responsible for providing the web based GUI node configuration, and initial host configuration files.

Rackspace Private Cloud

Rackspace offers a toolset that enables users to quickly deploy a private cloud OpenStack cluster configured according to the recommendations of Rackspace OpenStack specialists [5]. Rackspace Private Cloud enables users to create an OpenStack cluster on Ubuntu, CentOS, or RHEL, using a set of installation scripts. This tool supports many OpenStack components, such as floating IP address management, security groups, availability zones, and the python-novaclient command line client. However, it does not support such functionalities as Nova Object Storage, Nova Volumes and centralized metadata servers.

2.1.6 Architecture Design

In order to support the installation, updating and managing of XIFI nodes, we have designed a modular tool, named Infrastructure Toolbox (ITBox). Its architecture is shown in Figure 2.
The architecture design has been driven by the analysis of existing tools that allow bare-metal deployment of IaaS platforms.

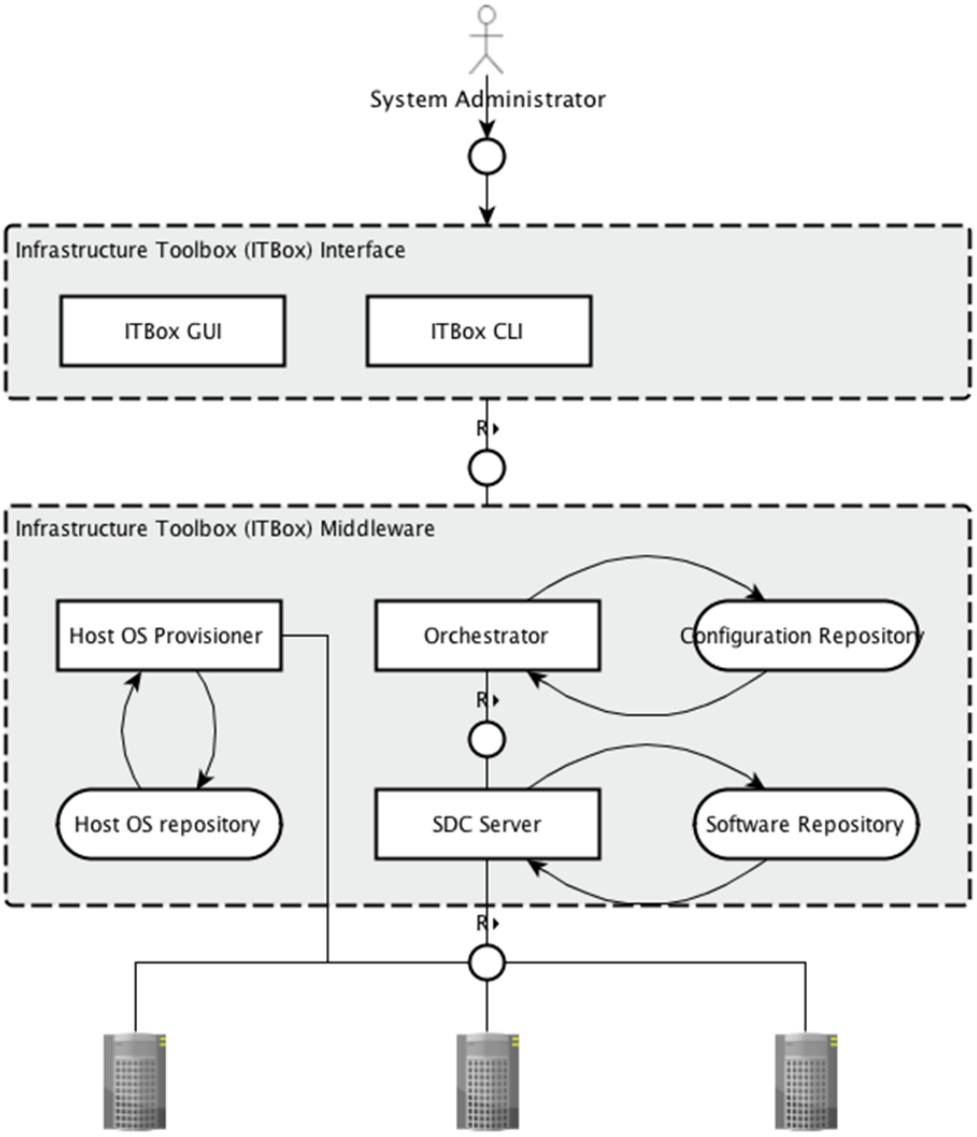

Figure 2: ITBox Reference Architecture

The Infrastructure Toolbox architecture is composed of an interface layer named the Infrastructure Toolbox Interface (shown in the upper box) and of the Infrastructure Toolbox Middleware (lower box).
We now see in detail how these two layers are composed.

- ITBox Interface that allows the System Administrator interfaces to run the set-up and installation of a new XIFI node. In particular the Interface provides:
  - ITBox Graphical User Interface: a web based interface with simplified functionalities for set-up and installation.
  - ITBox Command Line Interface: a shell based interface with advanced functionalities for set-up and installation.
- ITBox Middleware that provides the services that run the actual provision and deployment of OS and services on top of bare-metal infrastructure.
  - The Host Operating System Provisioner: a server that installs via network operating system on node discovered via PXE protocol or similar. It has a repository of host operating systems that can be provisioned on the bare-metal servers.

- The Orchestrator: a server that coordinates the deployment of services on the different servers according to the configuration passed by the ITBox Interface. The orchestration among different servers is needed to ensure proper set-up and configuration of the XIFI node (e.g. knowing when a cloud controller server is ready to register compute servers). The orchestrator relies on a repository of scripts.
- The Software Deployment and Configuration (SDC):a server that provides access to packages and scripts for the installation of services on a single node. It relies on a Software and Scripts repository. (Note: this service has the same role as the FI-WARE SDC GE, but a different scope, i.e. it is not meant for the installation of GEs on top of VMs provisioned by the DCRM, but for the installation of DCRM itself and additional services).

2.1.7 Release Plan

Version Id	Milestone	User Stories
1.0	30.11.2013	1,2
1.1	30.12.2013	3,4,7,8,10
1.2	30.01.2014	5,6,9
2.0	28.02.2014	11,12,13,14
3.0	30.06.2014	15

2.1.8 Test Cases

In order to verify the correct installation of the ITBox, the user can use the following command:

```
''curl http://10.20.0.2:8000/api/releases/''
```

The answer of the ITBox Master server (v1.1 version) should be as follows:

```
[
  {
    "operating_system": "CentOS",
    "description": "This option will install the OpenStack Grizzly packages using CentOS as a base operating system. With high availability features built in, you are getting a robust, enterprise-grade OpenStack deployment.",
    "roles": [
      "cinder",
      "controller",
      "compute",
      "ceph-osd"
    ],
    "state": "available",
    "version": "2013.1.4",
    "roles_metadata": {
      "cinder": {
        "name": "Cinder",
        "description": "Cinder provides an infrastructure for managing block storage volumes in OpenStack.  Block storage can be used for database storage, expandable file systems, or providing a server with access to raw block level storage."
```

```
      },
      "controller": {
        "conflicts": [
          "compute"
        ],
        "name": "Controller",
        "description": "The controller initiates orchestration activities and provides an external API.  Other components like Glance (image storage), Keystone (identity management), Horizon (OpenStack dashboard) and Nova-Scheduler are installed on the controller as well."
      },
      "compute": {
        "conflicts": [
          "controller"
        ],
        "name": "Compute",
        "description": "A compute node creates, manages and terminates virtual machine instances."
      },
      "ceph-osd": {
        "name": "Ceph OSD",
        "description": "Ceph OSD is the object storage daemon for the Ceph distributed file system.  It is responsible for storing objects on a local file system and providing access to them over the network, for example to Glance."
      }
    },
    "modes_metadata": {
      "ha_compact": {
        "description": "This configuration requires multiple OpenStack controllers (3+) and provides high availability for all OpenStack components, including MySQL/Galera, RabbitMQ, and Cinder, as well as OpenStack API services. Select this configuration if you want to build a production-grade OpenStack cloud with 6 nodes or more."
      },
      "multinode": {
        "description": "In this configuration the OpenStack controller is deployed separately from the compute and cinder nodes. This mode assumes the presence of 1 controller node and 1 or more compute/cinder nodes. You can add more nodes to scale your cloud later."
      }
    },
    "id": 1,
    "name": "Grizzly on CentOS 6.4"
  },
  {
    "operating_system": "RHEL",
    "description": "This option will install the Red Hat Enterprise Linux OpenStack Platform using RHEL as a base operating system.  With high availability features built in, you are getting a robust enterprise-grade OpenStack deployment.",
    "roles": [
      "cinder",
      "controller",
      "compute"
```



```
    ],
    "state": "not_available",
    "version": "2013.1.4",
    "roles_metadata": {
      "cinder": {
        "name": "Cinder",
        "description": "Cinder provides an infrastructure for managing block storage volumes
in OpenStack.  Block storage can be used for database storage, expandable file systems, or
providing a server with access to raw block level storage."
      },
      "controller": {
        "conflicts": [
          "compute"
        ],
        "name": "Controller",
        "description": "The controller initiates orchestration activities and provides an
external API.  Other components like Glance (image storage), Keystone (identity management),
Horizon (OpenStack dashboard) and Nova-Scheduler are installed on the controller as well."
      },
      "compute": {
        "conflicts": [
          "controller"
        ],
        "name": "Compute",
        "description": "A compute node creates, manages and terminates virtual machine
instances."
      }
    },
    "modes_metadata": {
      "ha_compact": {
        "description": "This configuration requires multiple OpenStack controllers (3+) and
provides high availability for all OpenStack components, including MySQL, QPID, and Cinder, as
well as OpenStack API services. Select this configuration if you want to build a production-
grade OpenStack cloud with 6 nodes or more."
      },
      "multinode": {
        "description": "In this configuration the OpenStack controller is deployed separately
from the compute and cinder nodes. This mode assumes the presence of 1 controller node and 1
or more compute/cinder nodes. You can add more nodes to scale your cloud later."
      }
    },
    "id": 2,
    "name": "RHOS 3.0 for RHEL 6.4"
  },
  {
    "operating_system": "Ubuntu",
    "description": "This option will install the OpenStack Grizzly packages using Ubuntu as a
base operating system. With high availability features built in, you are getting a robust,
enterprise-grade OpenStack deployment.",
    "roles": [
```

```
      "monitoring",
      "compute",
      "swift_storage",
      "swift_proxy",
      "controller",
      "cinder",
      "empty"
    ],
    "state": "available",
    "version": "2013.1.4",
    "roles_metadata": {
      "compute": {
        "conflicts": [
          "controller,monitoring,empty,swift_proxy,swift_storage"
        ],
        "name": "Compute",
        "description": "A compute node creates, manages and terminates virtual machine instances."
      },
      "swift_storage": {
        "conflicts": [
          "controller,compute,monitoring,cinder,empty,swift_proxy"
        ],
        "name": "Swift Storage",
        "description": "It is a blob storage server that can store, retrieve and delete objects stored on local devices."
      },
      "monitoring": {
        "conflicts": [
          "compute,cinder,empty,swift_proxy,swift_storage"
        ],
        "name": "Monitoring",
        "description": "Monitoring GE (Cosmos and Context Broker) and XIMM (XIFI Infrastructure Monitoring Middleware)."
      },
      "swift_proxy": {
        "conflicts": [
          "compute,monitoring,cinder,empty,swift_storage"
        ],
        "name": "Swift Proxy",
        "description": "Swift Proxy handles all incoming API requests."
      },
      "controller": {
        "conflicts": [
          "compute,empty,swift_storage"
        ],
```



```
        "name": "Controller",
        "description": "The controller initiates orchestration activities and provides an external API.  Other components like Glance (image storage), Keystone (identity management), Horizon (OpenStack dashboard) and Nova-Scheduler are installed on the controller as well."
      },
      "cinder": {
        "conflicts": [
          "empty,swift_proxy,swift_storage"
        ],
        "name": "Cinder",
        "description": "Cinder provides an infrastructure for managing block storage volumes in OpenStack.  Block storage can be used for database storage, expandable file systems, or providing a server with access to raw block level storage."
      },
      "empty": {
        "conflicts": [
          "controller,compute,monitoring,cinder,swift_proxy,swift_storage"
        ],
        "name": "Empty",
        "description": ""
      }
    },
    "modes_metadata": {
      "ha_compact": {
        "description": "This configuration requires multiple OpenStack controllers (3+) and provides high availability for all OpenStack components, including MySQL/Galera, RabbitMQ, and Cinder, as well as OpenStack API services. Select this configuration if you want to build a production-grade OpenStack cloud with 6 nodes or more."
      },
      "multinode": {
        "description": "In this configuration the OpenStack controller is deployed separately from the compute and cinder nodes. This mode assumes the presence of 1 controller node and 1 or more compute/cinder nodes. You can add more nodes to scale your cloud later."
      }
    },
    "id": 3,
    "name": "Grizzly on Ubuntu 12.04"
  }
]
```


2.1.9 Installation Manual

ITBox is distributed as an ISO image that contains an installer for ITBox Master Server. The ISO can be installed in the same way, using a virtualization software package, such as VirtualBox, or a bare-metal server. The first solution is suggested for testing scopes, whereas the second solution is suggested for production environment.
Suggested minimum hardware requirements for installation in testing environment:

- Dual-core CPU
- 2+ GB RAM
- 1 gigabit network port

- HDD 80 GB with dynamic disk expansion

Suggested minimum hardware requirements for installation in production environment:

- Quad-core CPU
- 4+ GB RAM
- 1 gigabit network port
- HDD 128+ GB

Once the Master Server is installed, all other servers can be powered on, and the user can login into the ITBox User Interface (UI) using the default address http://10.20.0.2:8000/, or they can start using the command line interface [7]. The cluster's servers will be booted in bootstrap mode (CentOS based Linux in memory) via PXE. Thus, these servers will be seen by the system as "discovered", and user will see notifications in the user interface. At this point the user can create an environment, add servers into it and start with the configuration.

2.1.10 User Manual

When the user has completed the Master Node installation, they can access ITBox UI, visiting the default URL http://10.20.0.2:8000/, as depicted in Figure 3.

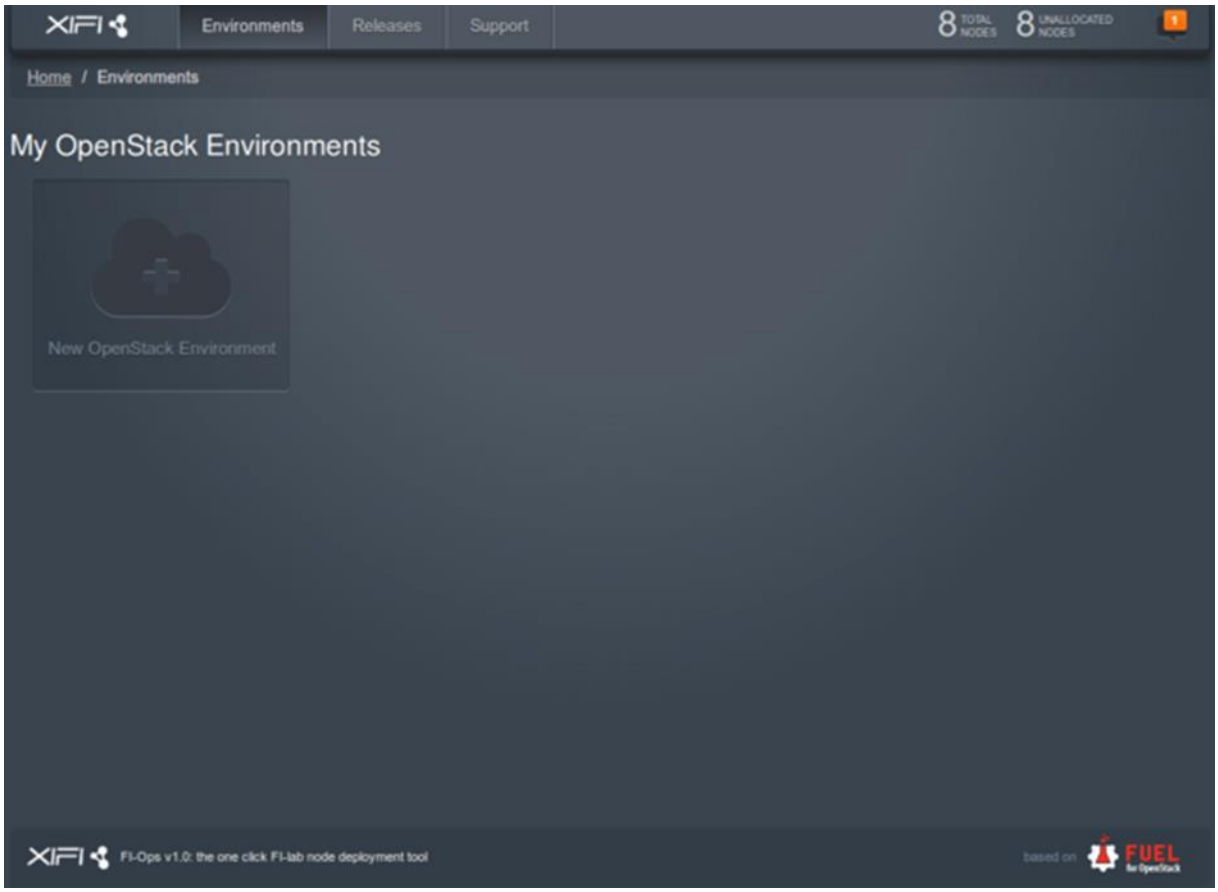

Figure 3: ITBox homepage

The user sets bare-metal servers to boot from network via PXE and power them on. They will start automatically with a bootstrap operating system, based on Centos. The ITBox will notify about discovered nodes on ITBox UI (see Figure 3in the upper right corner). At this moment, the user could create a new environment.

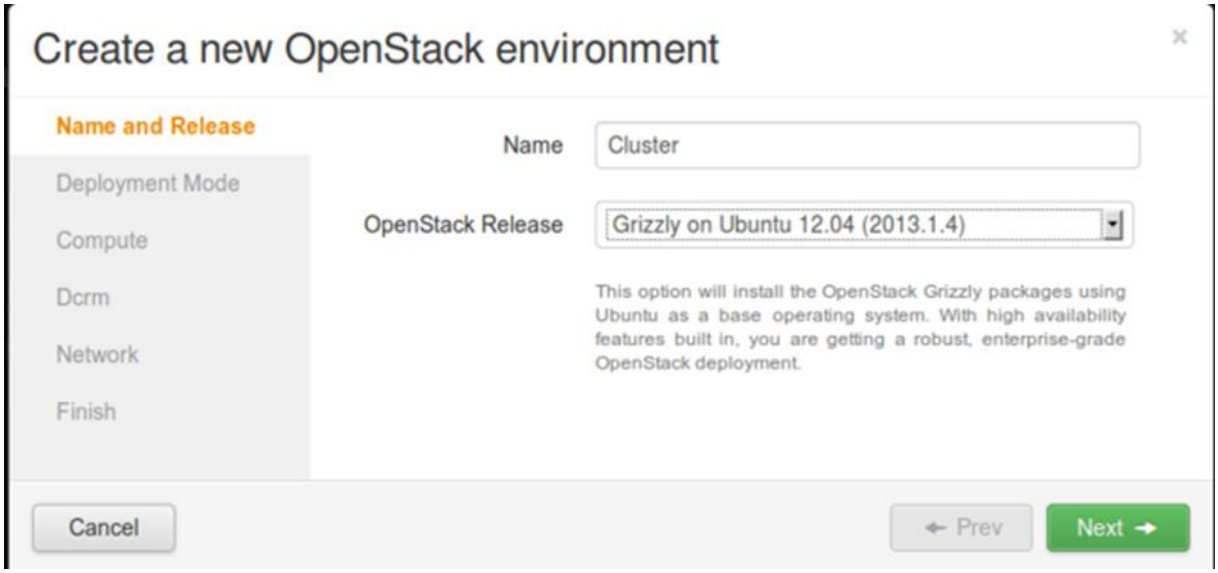

Figure 4: Creation of a new environment

The first step that involves the user is the "New OpenStack Environment" creation (Figure 4), where the user inserts such basic information about the environment as name, operating system, deployment mode (multi-node or multi-node with High Availability), hypervisor, DCRM GE with Pivot or Pulsar scheduler and network manager (Nova-Network, Neutron with GRE, Neutron with VLAN).
The DCRM GE installation (Figure 5) is a XIFI specific feature. If the user selects a DCRM GE scheduler, the ITBox will install all necessary packages and configure Pivot or Pulsar scheduler.
If the user skips this step, then the ITBox will install the FilterScheduler as default.

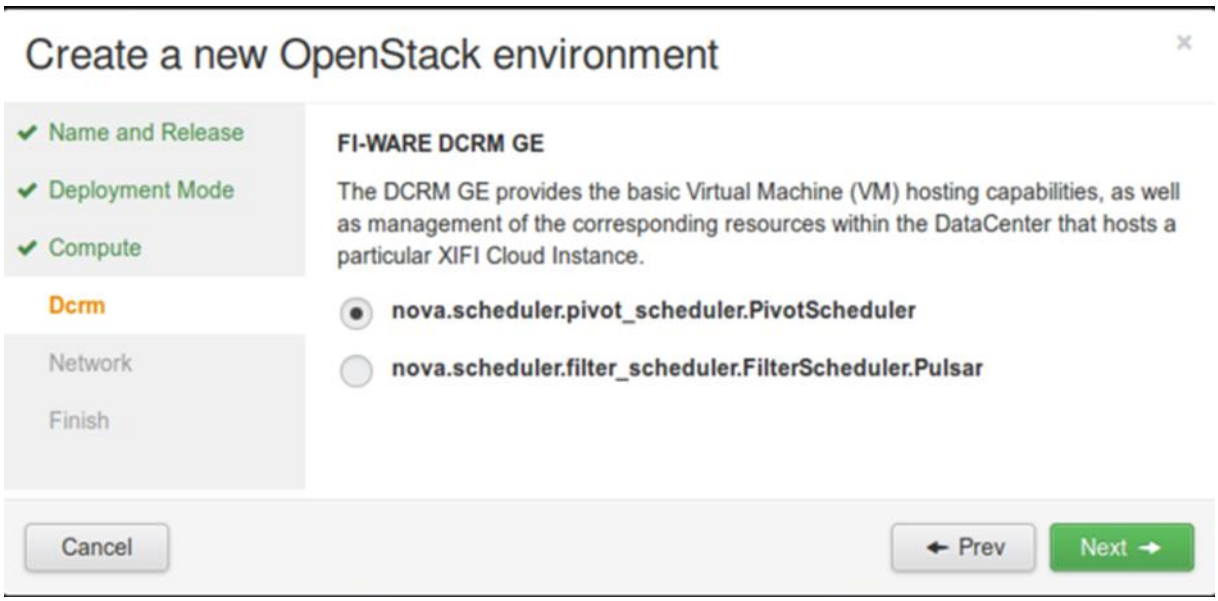

Figure 5: DCRM installation options

Now the environment is ready for deployment (Figure 6, Figure 7).

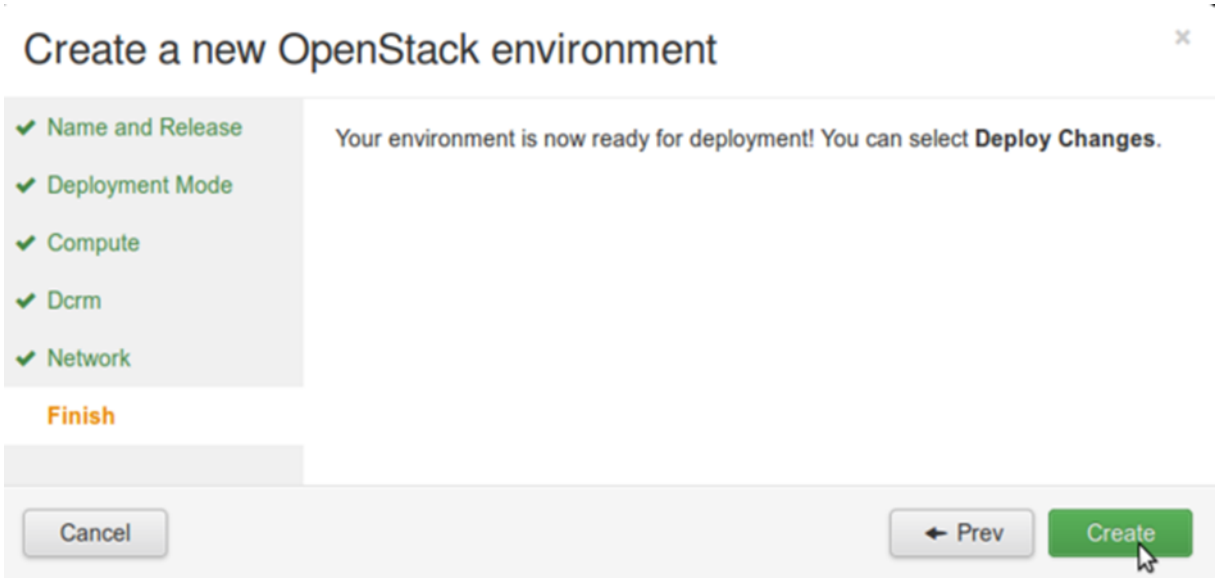

Figure 6: Final creation step

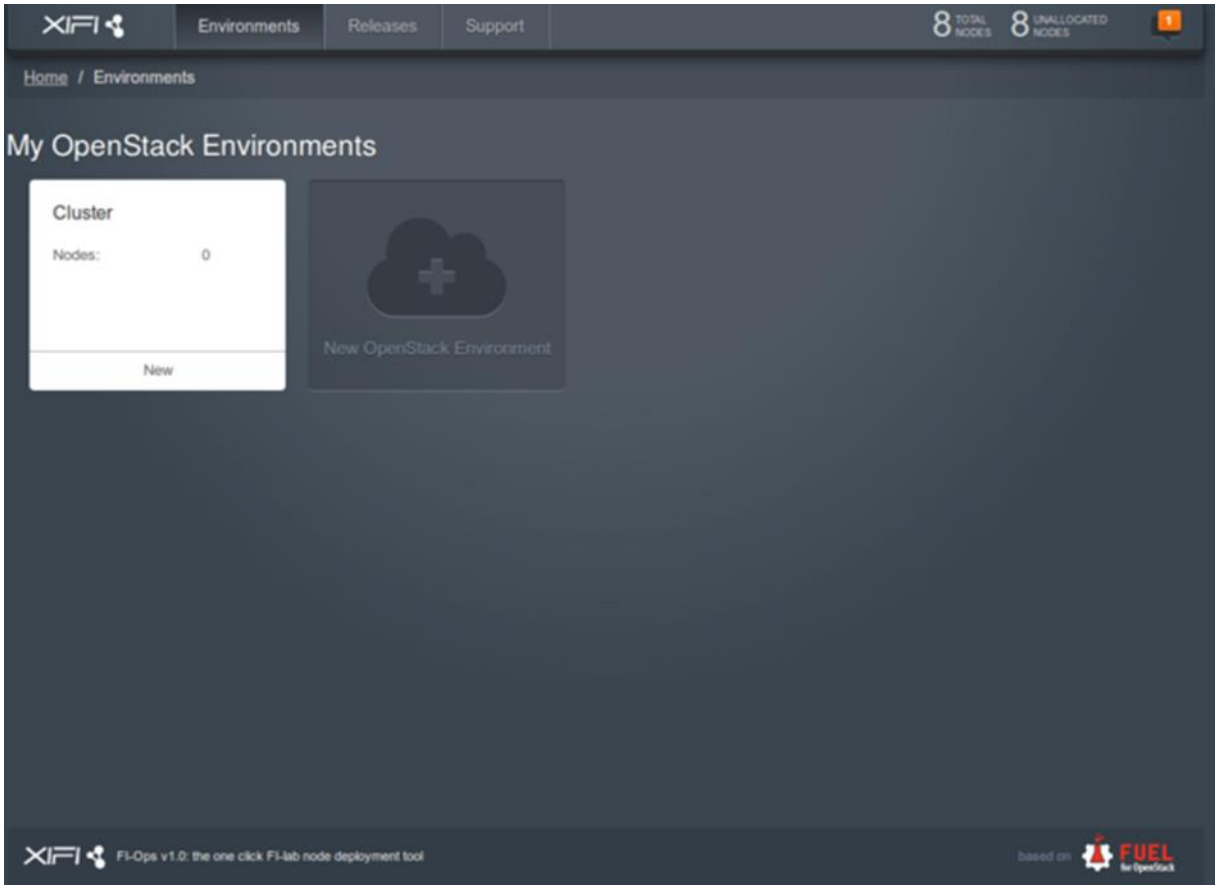

Figure 7: The page of the created environment

In the environment creation process the user should define the architecture of his cloud infrastructure. The user assigns a role to every server, configures the network, defines the space allocated to hard disks and sets other OpenStack options (Figure 8).

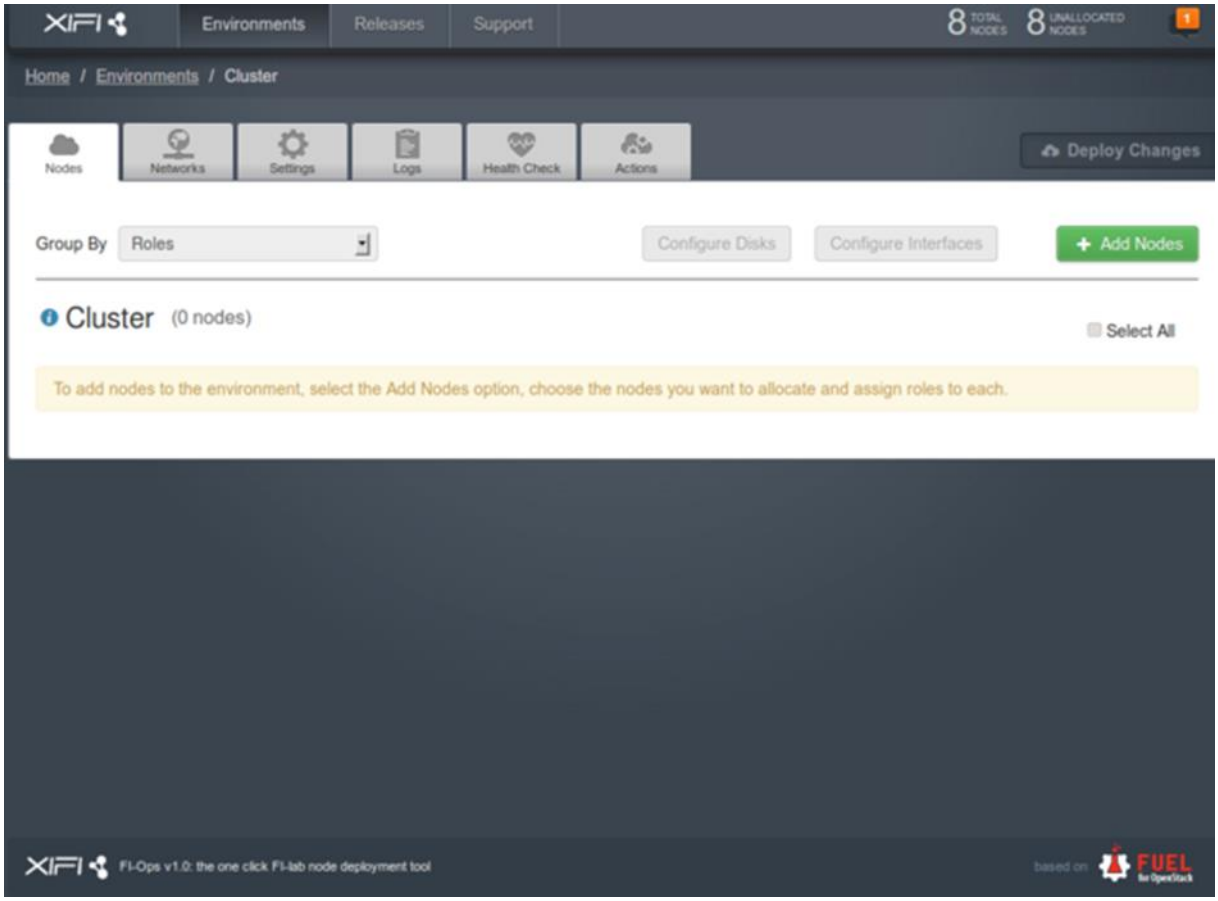

Figure 8:The definition of the environment

Giving roles to servers

In “Nodes” tab, the user can view the state of his environment and where the nodes are ordered by Roles. Thus, the user can view the node's details and configure it appropriately.

By clicking on the “Add Nodes” button, the ITBox shows users the list of available roles and the list of unallocated nodes. After selecting a role, other incompatible roles are automatically disabled. For example, a controller node cannot be together with a compute node on the same host, and so on.

Finally the user applies the changes (Figure 9).

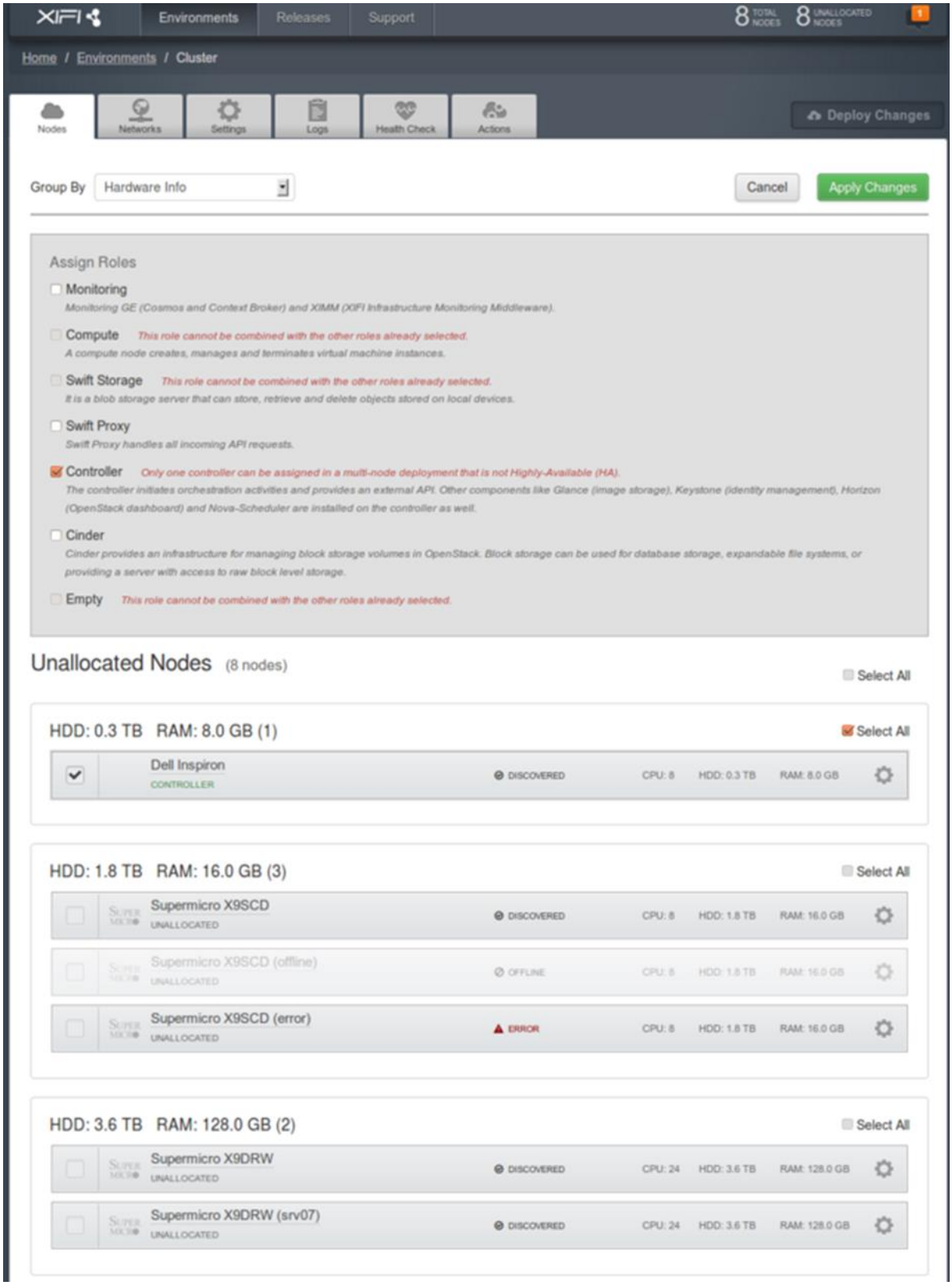

Figure 9: The list of the available servers

When the changes are applied, it is possible to tune the node, by clicking on the right button indicated by the gear icon. The ITBox shows a dialog where the user can configure network interfaces, defines

the space allocated to hard disks and views server information (e.g. Service tag, MAC addresses, hardware specifications, etc.) (Figure 10, Figure 11, Figure 12).

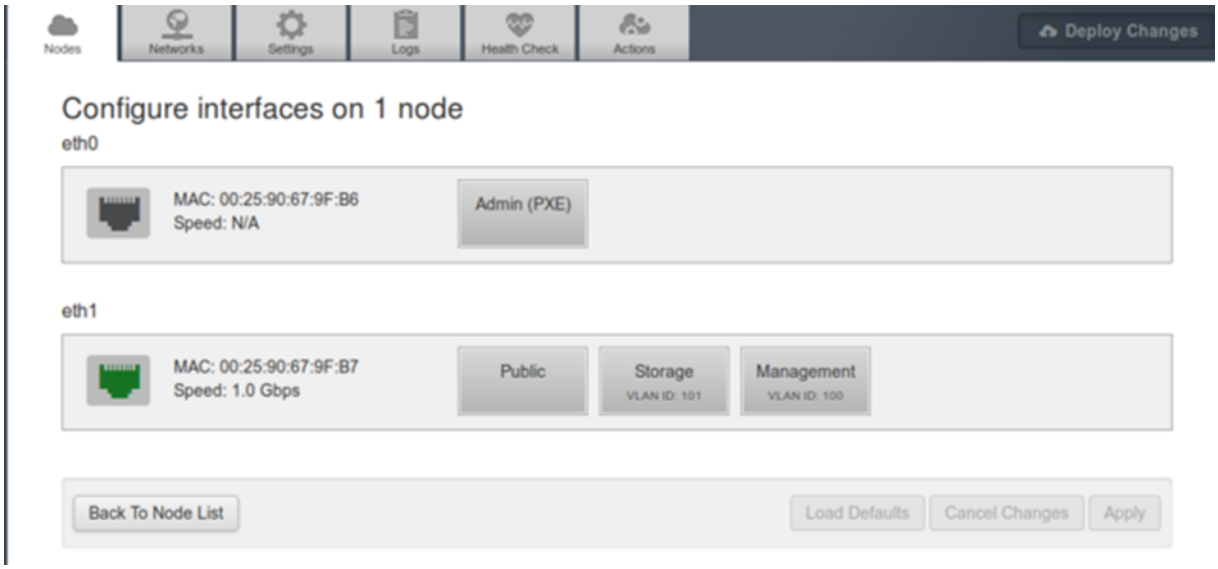

Figure 10: Network interfaces configuration

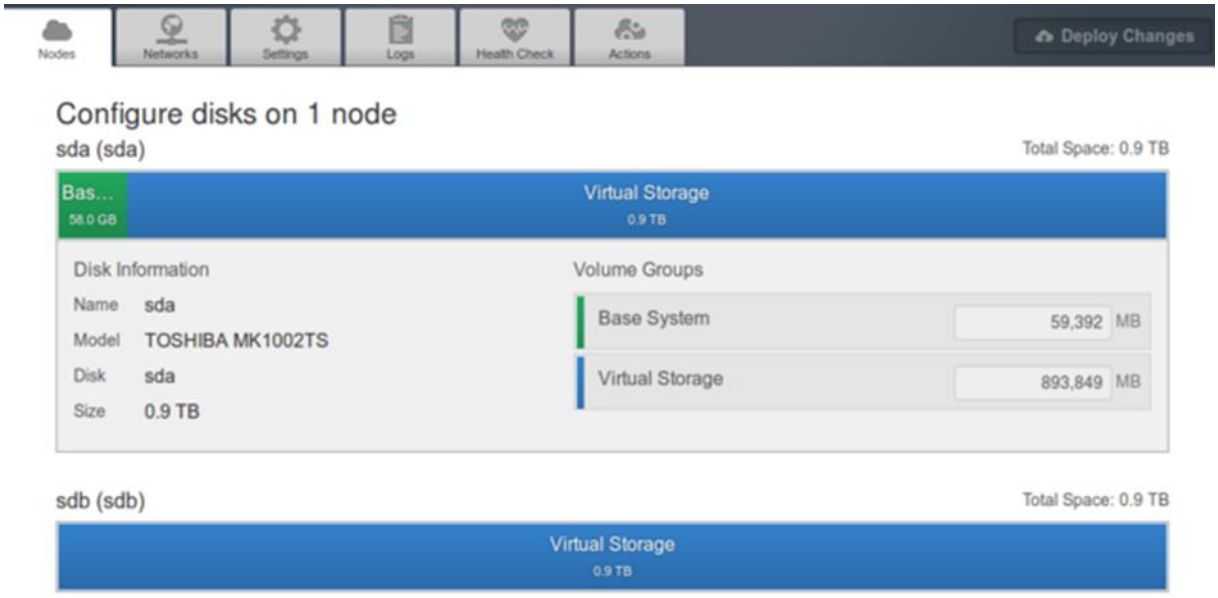

Figure 11: Hard disks configuration

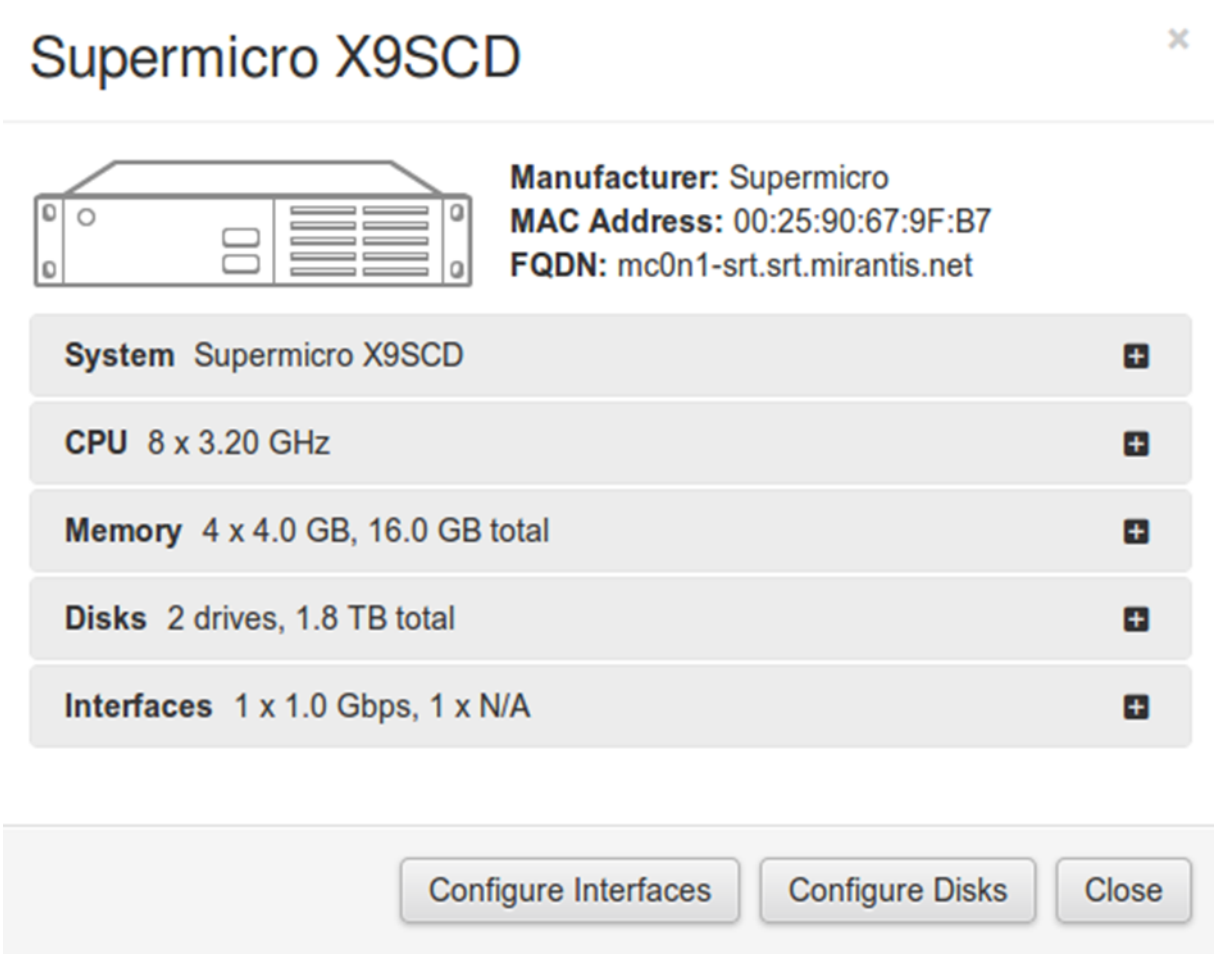

Figure 12: Detailed information about the selected server

Network settings

In the Network section, the user can manage configuration parameters. Based on the OpenStack network architecture, ITBox considers three networks: Public, Management and Storage. Management and Storage sections indicate the network subnet in CIDR notation and VLAN tags, whereas the Public section allows the user to set the IPs pool and its VLAN tag (Figure 13).

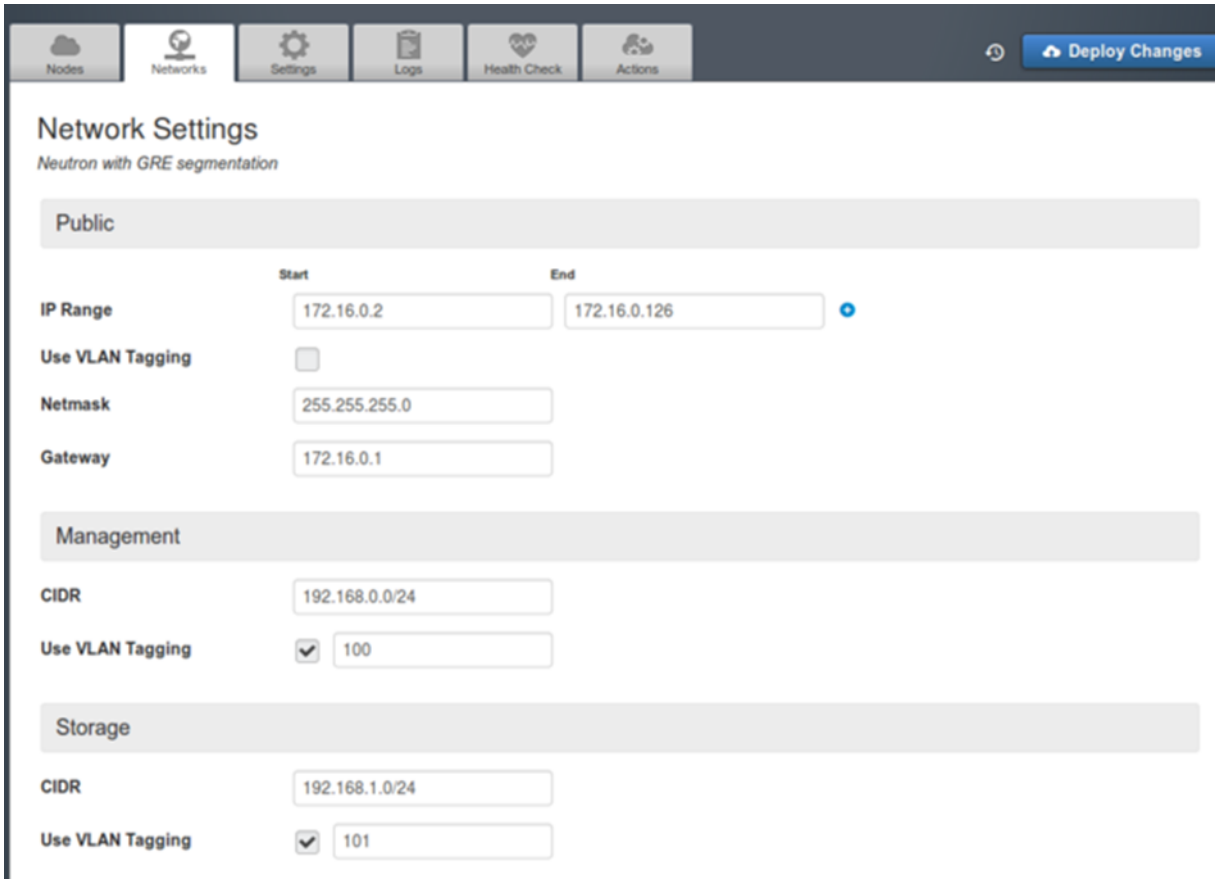

Figure 13: Infrastructure network settings

The ITBox gives the user the opportunity to manage the Neutron plugin and to define the L2 connection tunnel ID range and the L3 floating IP range. Furthermore, the user can verify the network configuration by clicking the "Verify Network" button, which checks for connectivity between nodes using the configured VLANs. It also checks if some external DHCP server might interfere with the current deployment (Figure 14).

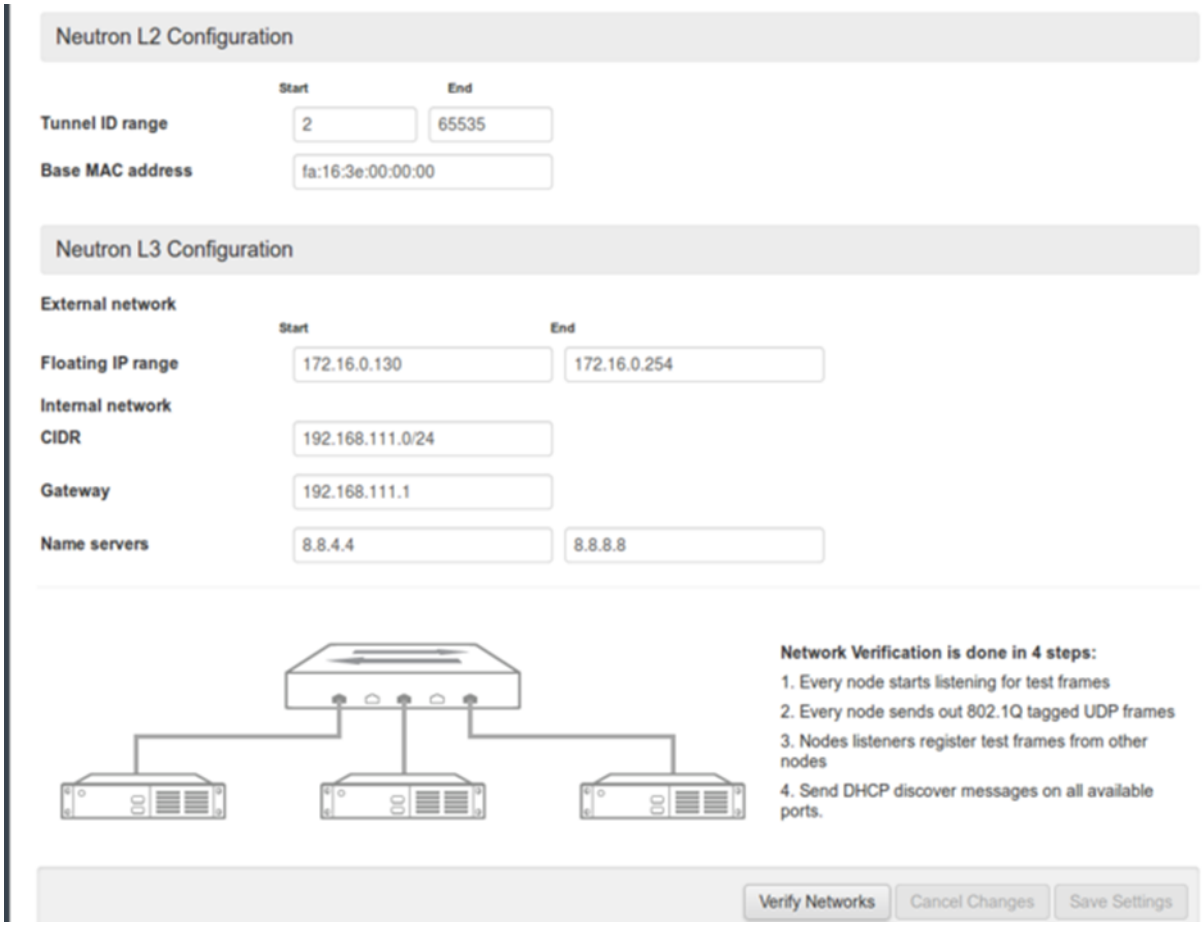

Figure 14: L2/L3 Neutron configuration

General Settings

The "Settings" tab contains useful options for managing the current environment. For example, the user can change the OpenStack admin account can change the hypervisor type or the scheduler driver. To make changes permanent it is necessary re-deploy the changes. (Figure 15, Figure 16).

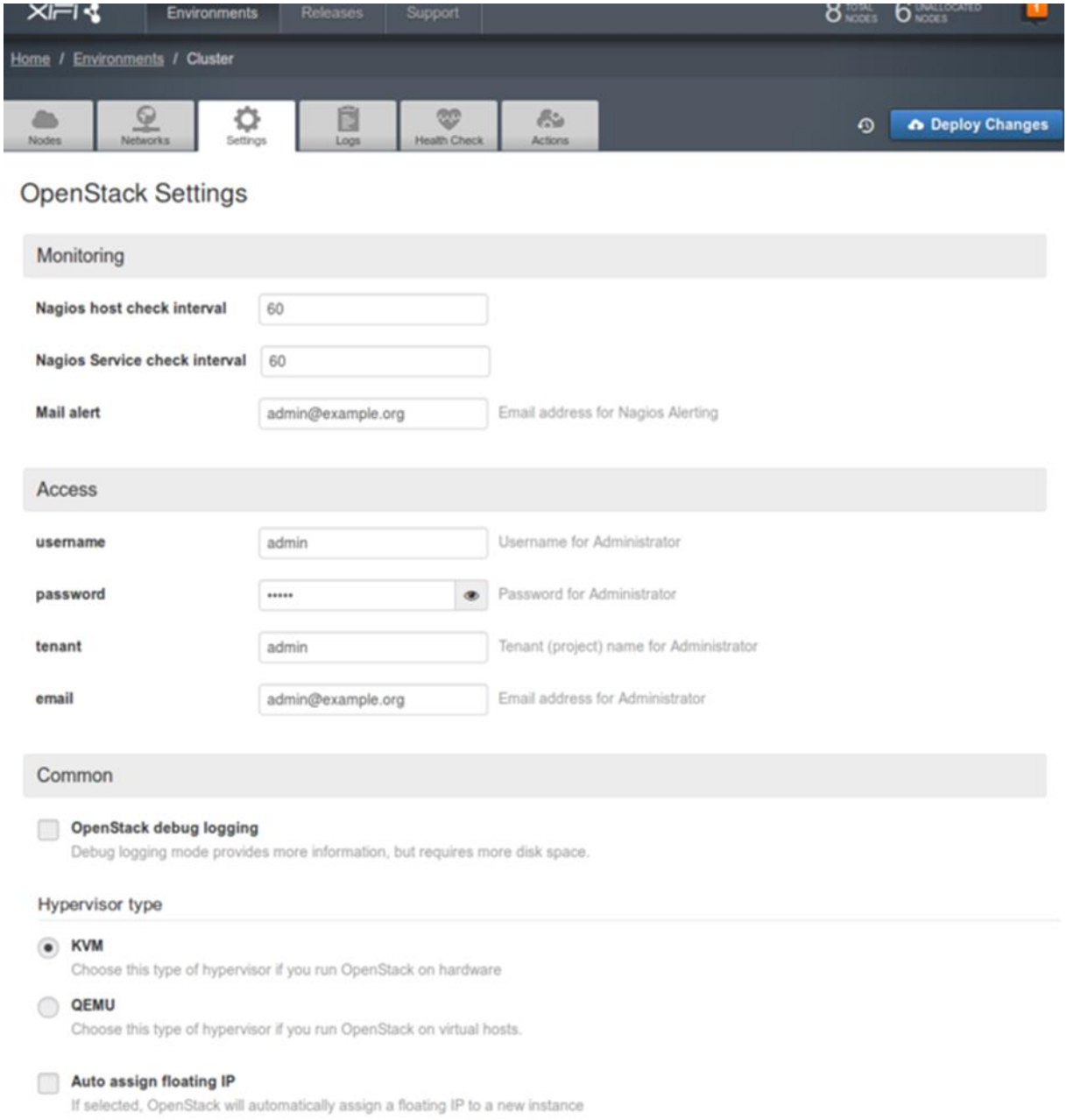

Figure 15: Infrastructure settings (monitoring, administrator account, common)

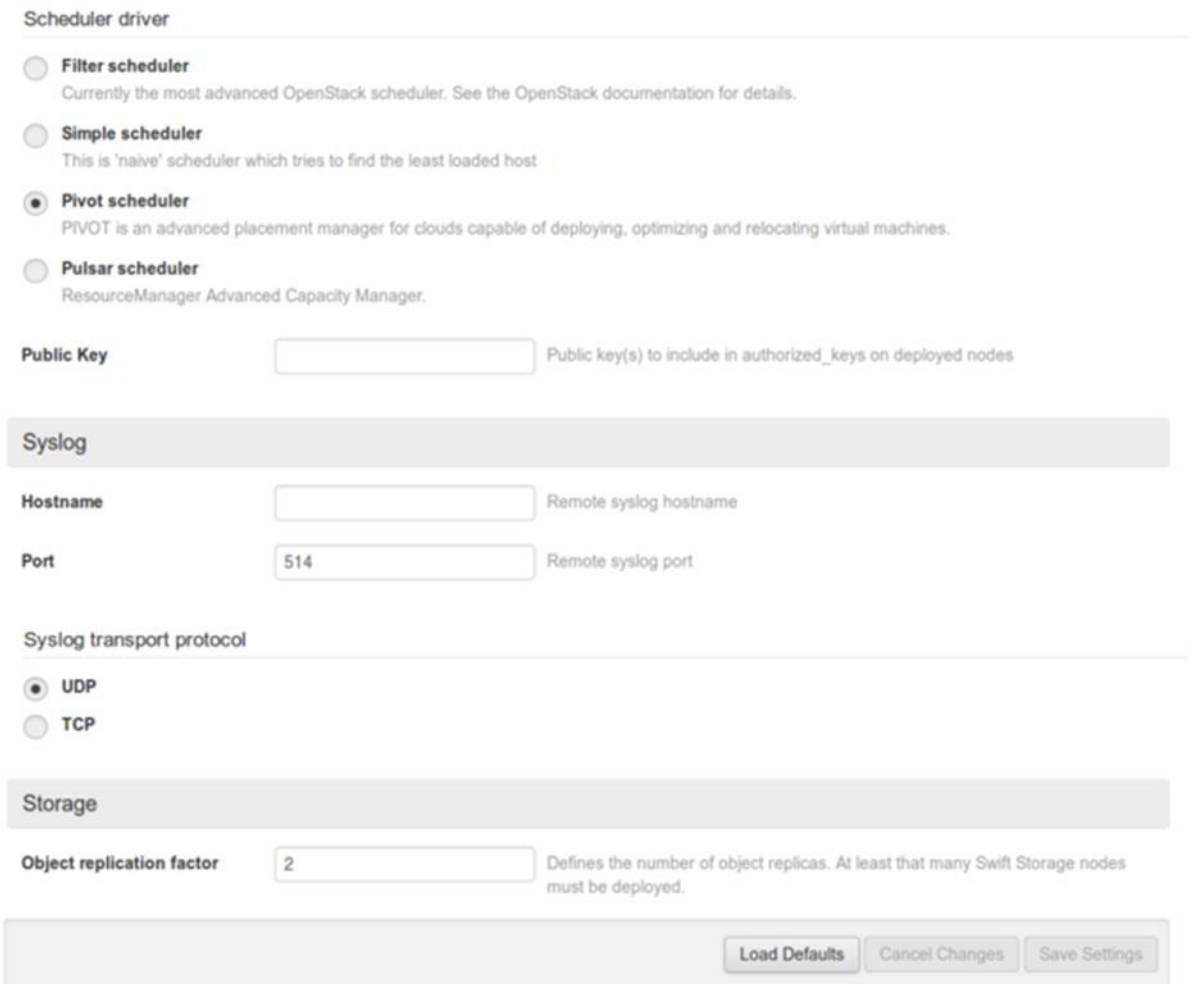

Figure 16: Infrastructure settings (scheduler drivers, syslog, storage)

Logs

The log section is designed to monitor the state of installation and support troubleshooting. The user can select the node to monitor, the log level and the generator source.

Health Check

It is very useful, running a post deployment test, to see if the installation process is correctly finished. The Health check process runs a set of tests, and when it is done, the user will see green Thumbs Up sign if it was correct and a red Thumbs Down sign if something went wrong (Figure 17).

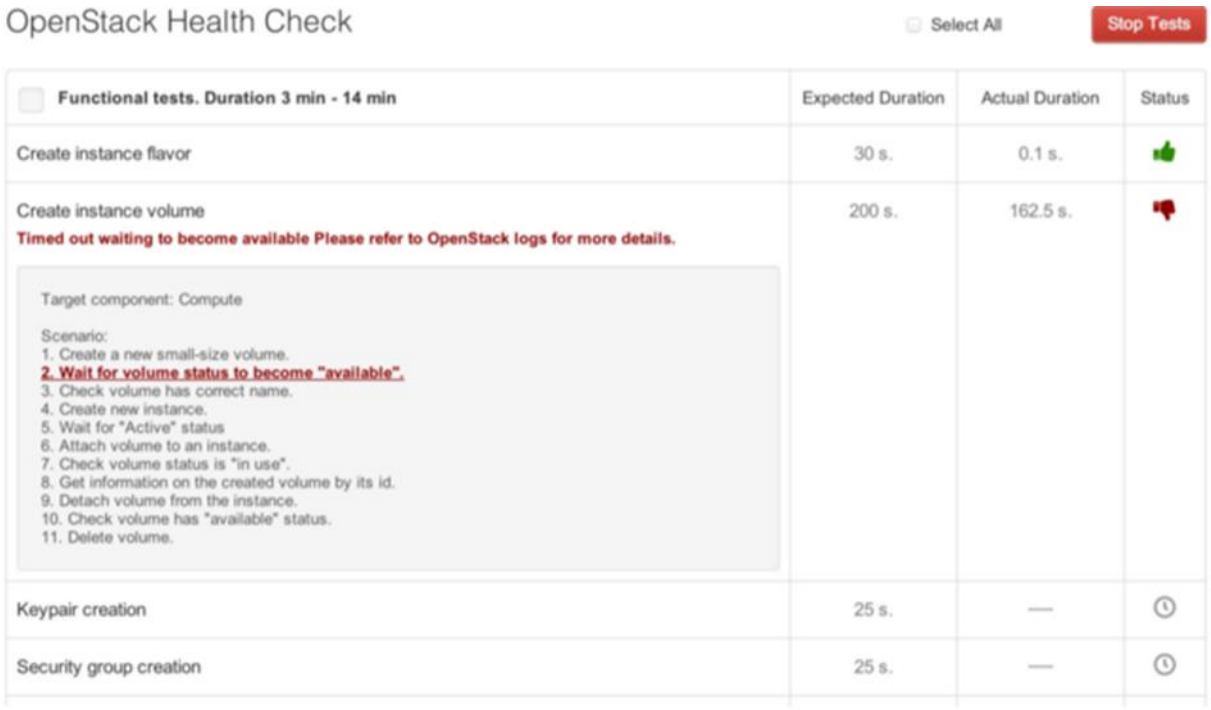

Figure 17: Health check result

Start deploy

When the user has finished setting the environment, they can start the deployment process, clicking on "Deploy changes" button (Figure 18).

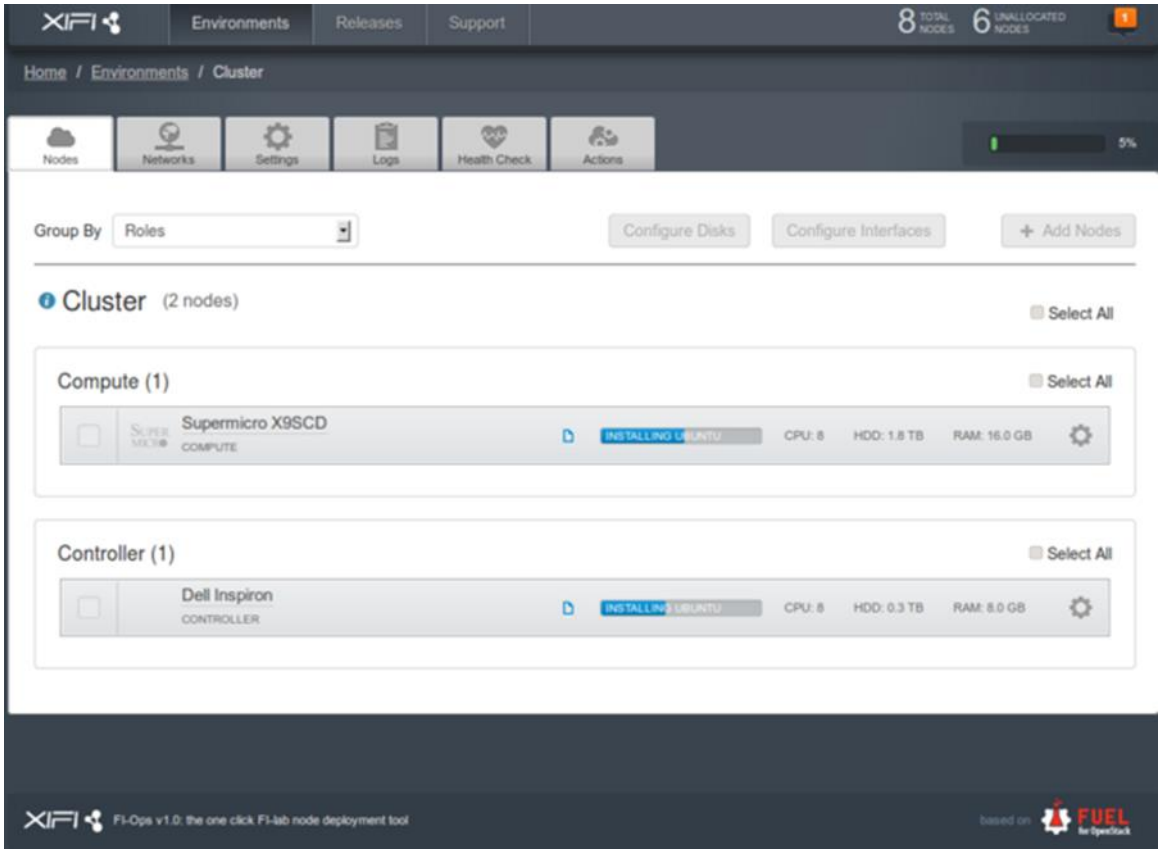

Figure 18: Installation in progress

2.1.11 Developer guide

The ITBox is based on Fuel by Mirantis released under Apache 2.0 license. The source code of ITBox can be found on https://github.com/SmartInfrastructures/ git repository:

- https://github.com/SmartInfrastructures/itbox-puppet
- https://github.com/SmartInfrastructures/itbox-web
- https://github.com/SmartInfrastructures/itbox-astute
- https://github.com/SmartInfrastructures/itbox-ostf
- https://github.com/SmartInfrastructures/itbox-main

The ITbox-main project contains the ISO build scripts so everyone can clone it and build an ITBox ISO. Requirements:

- Ubuntu 12.10

The steps are:

- Install the following required software: http://docs.mirantis.com/fuel-dev/develop/env.html#building-the-fuel-iso (add the following command to the list: sudo npm install -g requirejs)
- git clone https://github.com/SmartInfrastructures/itbox-main.git
- make iso

Personalized version of ITBox

A developer might be interested in producing a personalized ITBox version. In order to do so, they should clone (or fork) all repositories (https://github.com/search?q=@SmartInfrastructures%20itbox) and they should know some basic information on the ITBox project structure.

If the developer wants to add a new puppet module, they will make it in a deployment/puppet directory of the project https://github.com/SmartInfrastructures/itbox-puppet, which contains all puppet scripts. Finally, the new puppet module will included in /fuel/deployment/puppet/osnailyfacter/manifests/cluster_simple.pp and /fuel/deployment/puppet/osnailyfacter/manifests/cluster_ha.pp files.

For example:

```
include mypuppetscript
```

In order to build a new ISO, the developer will do the steps showed in previous section. Furthermore, they will update FUELLIB_REPO, NAILGUN_REPO, ASTUTE_REPO and OSTF_REPO fields in /fuel-main/config.mk with developer's repositories URL.

For more information see the Mirantis development documentation: http://docs.mirantis.com/fuel-dev/.

2.2 GE Deployment and Configuration Adapter (DCA)

2.2.1 Summary

The Deployment and Configuration Adapter (DCA) is the XIFI component that caters for the enhanced deployment functionality, as needed by the project users forming in parallel a Deployment Registry. Briefly the DCA provides:

(a) Deployment of multiple GEs and XIFI components upon XIFI infrastructure: The DCA supports the deployment and configuration of multiple GEs in a batch mode (as images, through recipes or in a combination), allowing the user to select details (including the sequential or parallel deployment and the notification capabilities). Such multi-GE deployment can take place in a single node or upon federated XIFI nodes. The DCA can also be used to deploy XIFI components upon the infrastructure.

(b) Check of Available Resources prior to the Deployment: The DCA performs check on the resources that are available to the user, prior to the deployment of one or multiple GEs and according to the documented hardware requirements of the GEs. This functionality can protect the user from receiving errors (by the platform) after invoking the deployment procedure. The resource availability check is performed considering the user's quota upon the XIFI infrastructure, the resources that have been already reserved by the user and the hardware needs of the GEs under deployment (currently quantified in CPU cores, memory and storage). The checks can be performed per node and / or federated nodes.

(c) Persistence of information related to the deployed GE instances: The DCA holds all pertinent information from the whole lifecycle of GE deployment. This includes the requests on behalf of the users (through the portal) and the system responses as related to the GE instances (going well beyond the typical awareness of the VM instances). This information is adapted and then exposed upon request to the components of WP4, through a set of meaningful queries.

The positioning of DCA in XIFI infrastructure is depicted in Figure 19.

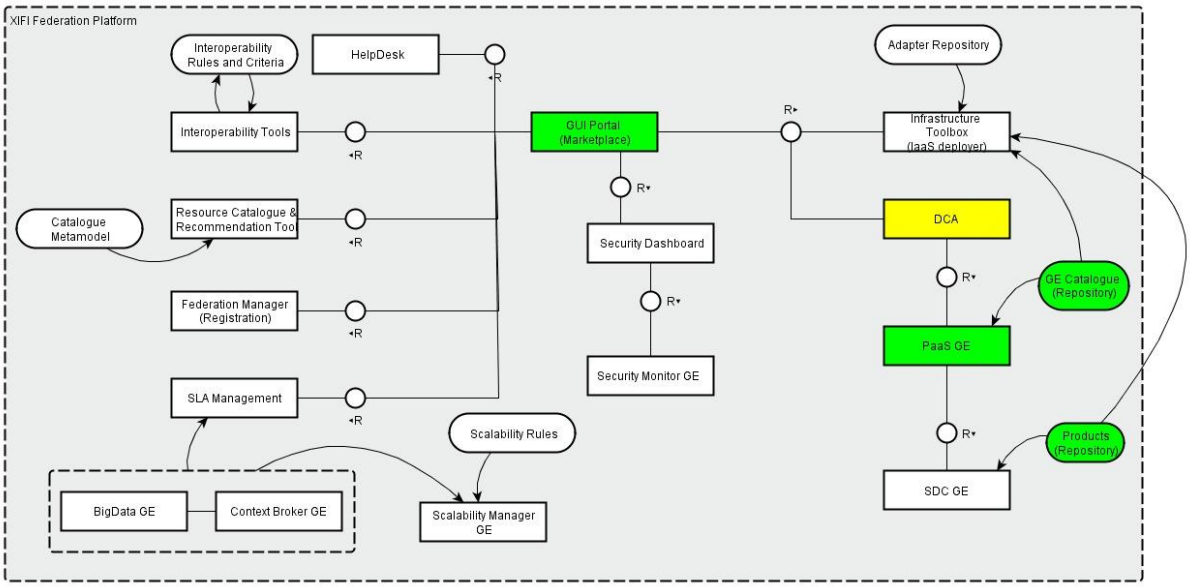

Figure 19: Location of the DCA in the XIFI Reference Architecture

DCA consists of

The DCA is composed of its internal information repository, the business logic and the set of API Handlers that allows for interconnection with other components

- The internal information repository handles the configuration details for the connection with the nodes and the GEs repository. It may also include (currently not externally available) the prerequisites (in terms of resources) for the deployment of GE
- The DCA component is used by the portal (or other client), which is responsible for orchestrating its functionality. Nevertheless the DCA includes a set of rules related to its operations. This includes first level checks (e.g. related to the necessary resources for GE deployment) and rules for the sequence of operations (e.g. check on resource availability prior to the deployment of a GE).
- The API Handlers are responsible for the interconnection with external components.

DCA depends on

The DCA depends on the following external entities:

- The Cloud management platform. In the first release of the DCA, we consider the OpenStack. In the second release it will be based on the DCRM.
- The repository of the Generic Enablers. This holds the Generic Enablers, (a) either as images or (b) as recipes combined with products.
- The Deployment and Configuration tool. This can be a third party tool, such as Chef, or FIWARE GEs, namely PaaS Manager and SDC.

Reference Scenarios	UC-2 - Setup and usage of a development environment
Reference Stakeholders	Developers: who want to deploy and configure (individual or groups) GEs upon a node or at a federation level and/or perform queries on the available GE instances. Infrastructure owners: who want deploy and configure (individual or groups) GEs upon a node or at a federation level (in a SaaS provision manner) and/or perform queries on the GE instances upon a node or the federation the infrastructure.
Type of ownership	Extension
Original tool	PaaS Manager GE
Planned OS license	PaaS Manager GE Terms and Conditions As the original tool.
Reference OS community	FI-WARE community.

2.2.2 Component responsible

Developers	Email	Company
P. Trakadas	ptrak@synelixis.com	SYNELIXIS
A. Papadakis	papadakis@synelixis.com	SYNELIXIS
A. Voulkidis	voulkidis@synelixis.com	SYNELIXIS
P. Karkazis	pkarkazis@synelixis.com	SYNELIXIS

2.2.3 Motivation

Since the XIFI project began, it was identified that the interconnection among the XIFI federated infrastructure, the UC projects and the FIWARE tools should be as smooth as possible, especially in the critical area of GE deployment and configuration. Our starting point has been based on the following assumptions:

- The users (application providers) have to be alleviated from as much overhead as possible when it comes to the deployment of their GEs, especially considering that deployment and configuration of GEs in groups is desired by UC projects
- The federated architecture multiplies possibilities for the users but also poses challenges for the deployment, configuration and management of the GEs; especially since the cloud hosting GEs have not catered (at least in their initial phases) for federated architectures.
- The XIFI project (even from in its Description of Work) has identified and described the need for an adaptor in the area of Enablers Deployment and Configuration, that will offer added value services (information) to the upper layer (mainly the portal) of the XIFI infrastructure and contribute to the deployment of the XIFI tools.

Indeed the Deployment and Configuration Adapter (DCA) has been specified since the beginning of the project (in D3.1) to fill this space. It also supports mainly the users and infrastructure owners. The functionality of the DCA is offered to users through the components of WP4 and specifically the portal and the individual modules that support its operations. The DCA leverages FIWARE functionality (mainly offered through the cloud hosting GEs) and offers a more flexible and user-friendly approach for special cases of GE deployment, configuration and management. Furthermore it offers a persistency layer for information related to the instances deployed upon the XIFI infrastructure. Finally, the DCA ensures compatibility with the:

- FIWARE cloud hosting GEs (including DCRM, and PaaS Manager)
- OpenStack
- 3rd party deployment tools (Chef)

2.2.4 User Stories backlog

User story id	User story name	Actors	Description
1	Create VM	User	The user creates a VM in the OpenStack (Grizzly) environment. This VM will accommodate the deployed GE(s). The result of the operation is stored locally (in the DCA).
2	Deploy and Configure a Product	User	The user deploys and configures a product (such as Apache Tomcat) in the OpenStack environment. The result of the operation is stored locally (in the DCA).
3	Deploy and Configure a GE using an image	User	The user deploys and configures a GE using an image in the OpenStack environment. The result of the operation is stored locally (in the DCA).
4	Deploy and Configure a group of GEs using images	User	The user deploys and configures GEs sequentially using images in the OpenStack environment. The result of the operation is stored locally (in the DCA).
5	REST interface	User	The information of the deployment and configuration results is offered through the REST interface (1st version of the REST

			interface).
6	Deploy and Configure a GE using a recipe / blueprint	User	The user deploys and configures a GE using a recipe / blueprint in the OpenStack environment. The result of the operation is stored locally (in the DCA).
7	Deploy and Configure a group of GEs using recipes	User	The user deploys and configures GEs sequentially using recipes / blueprints in the OpenStack environment. The result of the operation is stored locally (in the DCA).
8	Resource Availability Check	User	The user can ask the DCA to check for resource availability for the deployment of a specific GE upon a specific node.
9	Deploy and Configure a GE on multiple nodes	User	The user deploys and configures a GE using an image or a recipe/blueprint on multiple (more or equal to 2) federated XiFi nodes. These nodes have the OpenStack environment installed. The result of the operation is stored locally.
10	GE deployment Registry	User	The DCA holds the deployment and configuration information (successful and unsuccessful requests) in a Registry. The information can be offered through the DCA REST interface (version 2).
11	Query Information per GE and per node	User	The user can ask the DCA for the deployment and configuration requests (both successful and unsuccessful) per GE (or group of GEs) and per node (or group of federated nodes).
12	Complex Queries	User	The user can ask the DCA a set of complex queries, e.g. most frequent (successful, unsuccessful or both) GE deployment and configuration requests in a geographical location, in which nodes is (or is not) the CEP GE installed, which nodes have a combination of GEs, which are the infrastructure owners in Germany that use a specific GE.
13	Resource Availability Check	User	The user can ask the DCA to check for resource availability for the deployment of a specific GE upon the federation.
14	Full DCA functionality available	User	The full functionality of the DCA is available through the DCA API.

2.2.5 State of the Art

Current infrastructures empowered with virtualization capabilities and comprising multiple nodes call for robust mechanisms for the deployment of the functional elements (and specifically the Generic Enablers). In this section we perform a brief state of the art analysis and explain our selection. We initially refer to an IaaS deployment tool and then to two system configuration tools. We refer to FI-WARE pertinent Generic Enabler and then to our point of view.

CloudStack

Apache CloudStack is open source software designed to deploy and manage large networks of virtual machines as an Infrastructure as a Service (IaaS) cloud-computing platform [9]. It includes compute orchestration, Network as a Service, user and account management, resource accounting, an API and

UI. It supports VMware, KVM, XenServer and Xen Cloud Platform (XCP) hypervisors, with its API being compatible with AWS EC2 and S3.

Chef

Chef is a system configuration management tool, built on Ruby. It provides system configurations as "recipes" which describe the configuration of a series of resources (products) [10]. It can operate in a client server or standalone fashion. The configuration of Chef is based on Git. It has a stable and well-designed layout and according to [12], it can be natural fit for development-minded admins.

Puppet

Puppet as an open source configuration management tool (with Apache 2.0 license) built on Ruby [6]. Puppet provides similar functionality to Chef. Puppet is a mature solution. Chef and Puppet are the two basic options for software deployment and configuration in FI-WARE. According to [11], Puppet is appropriate for heterogeneous environments, with Ansible and Salt [12], [13] a better fit for larger but more homogenous infrastructures.

CloudBees

CloudBees offers a Platform as a Service (PaaS) to build, run, and manage web applications [14]. Its cloud services belong to two main categories: development services and deployment/management services web-based applications. Its orientation towards Java can create restrictions regarding other options.

FI-WARE PaaS Manager

According to FI-WARE, the PaaS Manager (Platform as a Service Manager) performs the installation and configuration of the whole software stack, taking into account the underlying virtual infrastructure. It enables multiple deployment architectures (based on single or multiple servers). The PaaS Manager wraps the functionality of the SDC (Software Deployment and Configuration) GE that is currently based upon Chef (with plans to support Puppet also).

Our Rationale and Selection

After this brief analysis we consider that the scope of the DCA is not to build from scratch another configuration management tool but to adapt the existing mechanisms to the needs of our users (developers and infrastructure owners and operators) as reflected in relative XIFI requirements. XIFI is tightly bound to FI-WARE handling federated, heterogeneous infrastructures. In this view we have opted for full compatibility with the FI-WARE and designed DCA as leveraging on PaaS Manager GE functionality with the target to provide additional offerings to the XIFI users (both developers and infrastructure owners).

2.2.6 Architecture Design

As described, the DCA provides an adaptation layer between the portal and the components that are responsible for the deployment and configuration of the Generic Enablers (GEs). In principle, we consider that a GE is deployed on a (single) VM. We use the off-the-shelf (third party) tools that have to be configured prior to the deployment of a GE (such as a DB, web server etc.). We assume the following:

- The prior deployment and configuration of the cloud hosting GEs (PaaS Manager, SDC, DCRM, SM)
- The availability of the recipes needed to deploy and configure a GE in an automated manner (as stated in the conditions and instructions of FIWARE)

The interaction of DCA with other XIFI components is depicted in Figure 20:

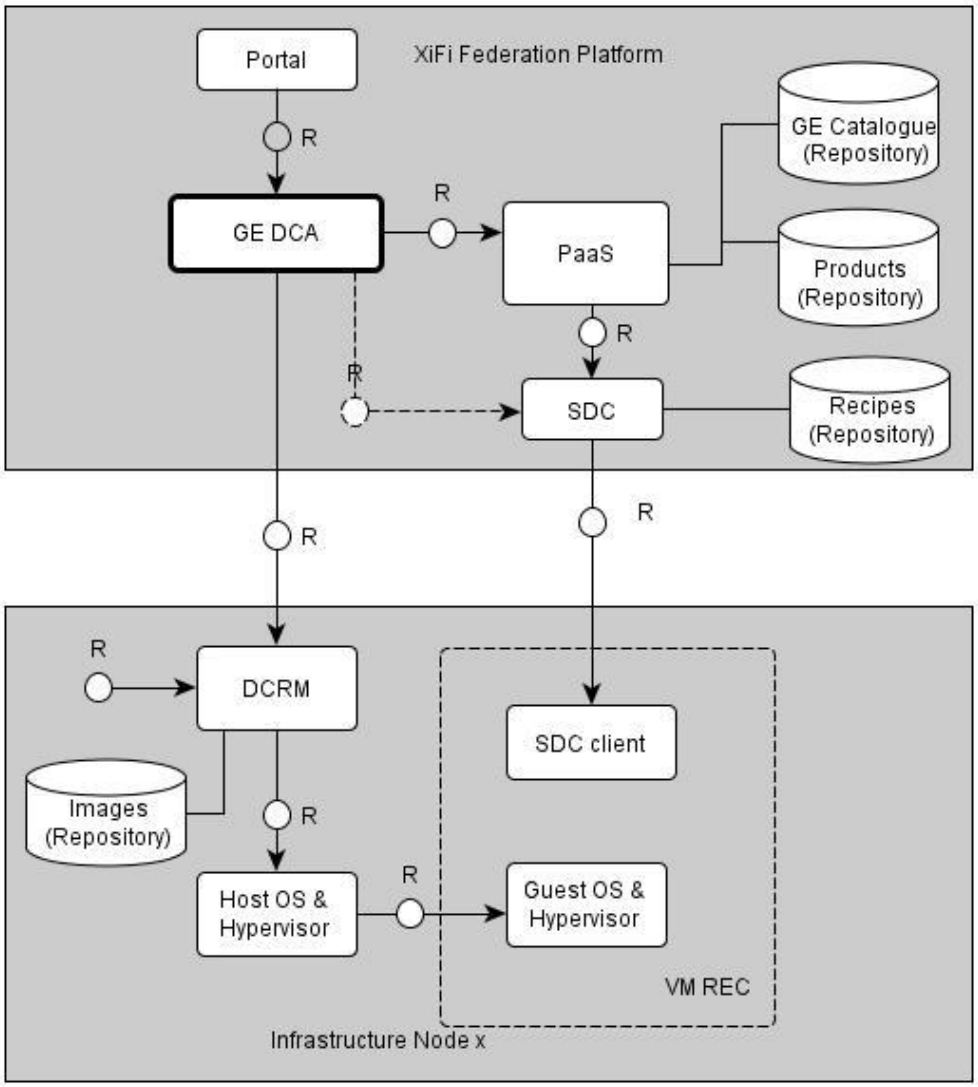

Figure 20: DCA positioning in the XIFI architecture

The DCA receives its instructions from the user through the XIFI portal. These instruction define the (group of) GE(s) to be deployed and the region for the deployment. The basis of the GEs grouping can be thematic (e.g. belonging to the same FI-WARE chapter) or of any arbitrary manner according to the application needs (for example an IoT application may need the IoT protocol adapter, the Context Broker and the Big Data Analysis GEs).

The DCA performs, interacting with the DCRM a check upon the necessary resources and then instructs the PaaS Manager for the deployment of the GE(s). One method invocation towards the DCA can be decomposed to multiple ones towards the PaaS Manager. The PaaS Manager utilizes the GE catalogue and the Products and Recipes repository to retrieve the necessary resources. The PaaS Manager instructs the SDC server (in the Master Node), which in turn instructs the SDC client, residing in the Infrastructure Node. In case the GE is deployed as an image the DCA interacts with the directly with the DCRM, which uses the image repository.

The DCA also deploys and configures GEs in a cluster of nodes. Node clustering is at the heart of federation and can be performed based on multiple criteria including the ownership by specific infrastructure owners, authentication / authorization rights, geographical location, resource availability, properties of the nodes, networking availability existence of GEs or any other arbitrary reasoning according to the application needs.

The DCA is responsible for storing the full set of information related to the exchanged requests and responses and relaying to the user.

The internal topology of DCA is depicted in Figure 21:

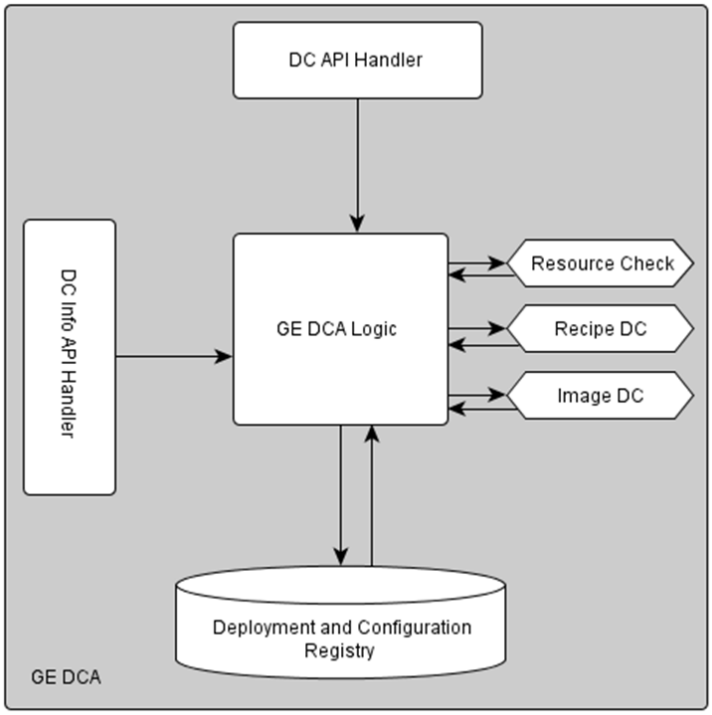

Figure 21: DCA internal topology

The internal architecture consists of the individual functional elements responsible for resource availability checking, and GE deployment and configuration, the API handlers (for deployment and configuration and information retrieval) and the Registry; all of them orchestrated by the internal DCA logic. The Repository holds the deployment and configuration information and exposes a set of queries, indicatively:

- Most frequent (successful, unsuccessful or both) GE deployment and configuration requests in a geographical location
- In which nodes is (or is not) a specific GE installed, e.g. CEP GE
- Which nodes have a combination of GEs
- Which are the infrastructure owners, e.g. in Germany, that use a specific GE
- Most frequent reasoning for unsuccessful deployment attempts

DCA architecture and internal design provides the following offerings:

- Verify the availability of resources that are necessary for the deployment of a GE
- Deploy and configure a GE or combination of GEs using a recipe and/or an image at a single node or a cluster of nodes
- Provide confirmation that the deployment and configuration of a GE has taken place properly
- Provide information (based on queries) on the GEs deployment and configuration requests on the VM in the node or node federation

2.2.7 Release Plan

Version Id	Milestone	User Stories
0.5	30.11.2013	1, 2
1.0	31.12.2013	3, 4, 5
1.5	31.03.2014	6, 7, 8
1.8	30.06.2014	9, 10, 11
2.0	30.09.2014	12, 13, 14

2.2.8 Test Cases

This section provides a guide for unit testing regarding the installation and configuration of the DCA and its required software components. This includes:

- Installation and configuration of Apache Tomcat,
- Installation and configuration of MySQL Server,
- Installation of the DCA binaries.

Prerequisites

For the node that will accommodate the DCA the prerequisites are the following:

- CentOS (6.2 and above) is already installed
- Oracle Java Runtime Environment (1.7.0 and above) is already installed
- Apache Tomcat (7.0.35 and above)
- The cloud-hosting Generic Enablers (GEs) that support the deployment and configuration of the GEs. Namely the PaaS Manager and the SDC.
- A persistency server (DCA currently tested with MySQL DB Server)

For the Infrastructure Node:

- The DCRM GE or OpenStack Grizzly

Test Case for Oracle Java installation

We verify that the right version of Oracle Java has been installed:

```
[root@dca ~]# java -version
java version "1.7.0_45"
Java(TM) SE Runtime Environment (build 1.7.0_45-b18)
Java HotSpot(TM) 64-Bit Server VM (build 24.45-b08, mixed mode)
```

Test Case for Apache Tomcat installation

We verify that Apache Tomcat has been installed visit from your browser the URL:

```
http://<dca-server-ip>
```

Figure 22 depicts the result of a successful Apache Tomcat installation:

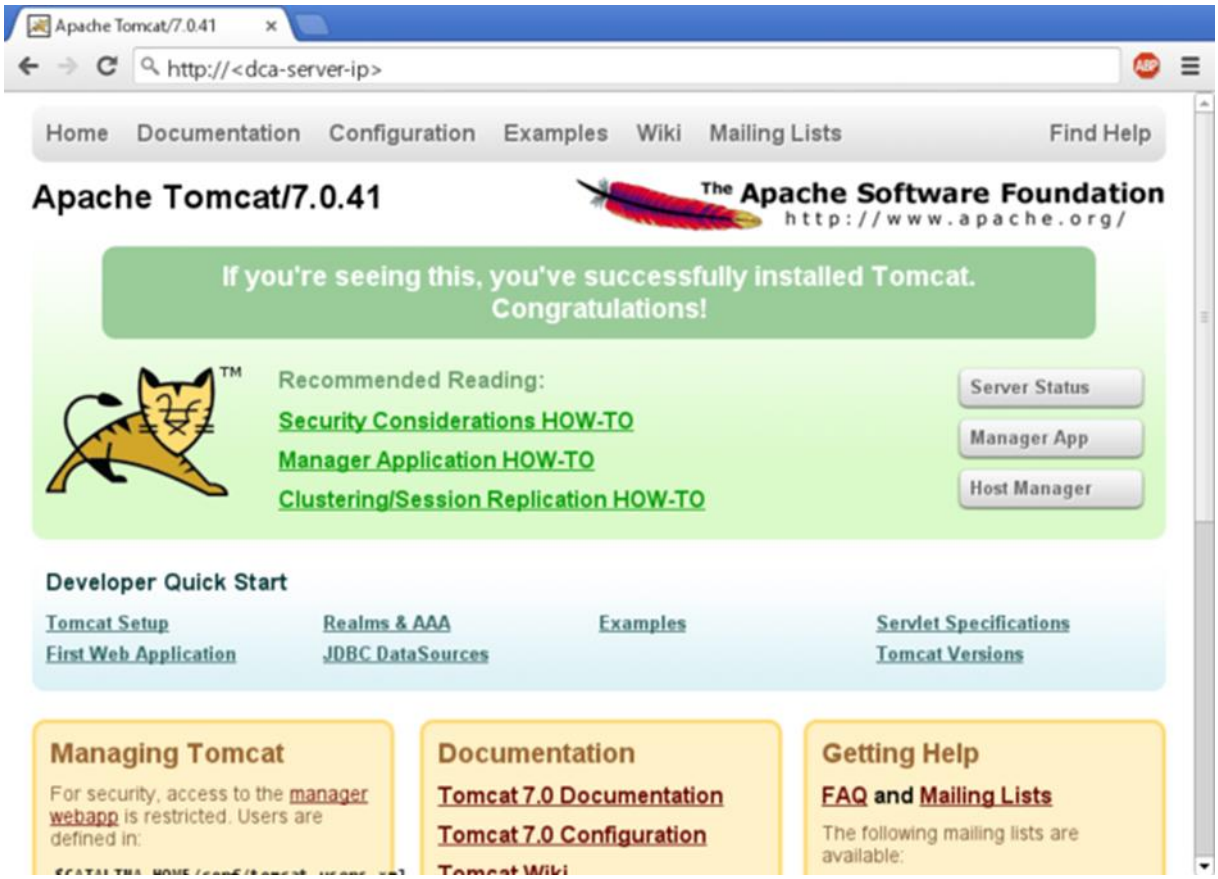

Figure 22: Verification of Apache Tomcat installation

Test case for MySQL and Java JDBC connector installation

We verify that MySQL server and the respective Java JDBC connector have been installed:

```
[root@dca opt]# service mysqld status
mysqld (pid  6557) is running...
[root@dca opt]# mysql_secure_installation
```

To verify that the database creation script has been executed, we execute the following:

```
[root@dca ~]# mysql -uroot -p -e 'show tables' dca
Enter password:
+-----------------+
| Tables_in_dca   |
+-----------------+
| ge              |
```

```
| hwrequirements |
| keystone       |
| request        |
| response       |
| server         |
+----------------+
```

We verify that the DCA binaries (dca.war) have been uploaded to the webapps directory of the Apache Tomcat installation and that a keystone instance has been initialized into the DCA platform. Supposing that the DCRM instance of the datacentre operator advertises its identity service at the URL http://hostname.example.com/keystone/v2.0 and is located in a region identified as A_Region, then, the following command should be issued:

```
curl http://<dca-server-ip>/dca/addEndpoint -X POST -H "Content-Type: application/json" \
       -d '{region:"A_Region", url:"http://hostname.example.com/keystone/v2.0"}'
```

The answer of the DCA server should be as follows:

```
{url:"http://hostname.example.com/keystone/v2.0", region:"A_Region"}
```

Then we verify the smooth operation of DCA invoking its operations as described in the User Manual. For example we list the available flavours in RegionThree.

```
curl http://localhost:8080/dca/flavors?region=RegionThree
```

Response

The available flavours are retrieved (tiny, small, medium).

```
{
   "flavors":[
      {
         "id":"1",
         "name":"m1.tiny",
         "vcpus":"1",
         "ram":512,
         "disk":"0",
         "swap":"",
         "links":[
            {
               "rel":"self",
               "href":"http://cloud.lab.fi-
ware.eu:8774/v2/00000000000000000000000000001807/flavors/1",
               "type":null
```

```
            },
            {
               "rel":"bookmark",
               "href":"http://cloud.lab.fi-
ware.eu:8774/00000000000000000000000000001807/flavors/1",
               "type":null
            }
         ],
         "public":null,
         "OS-FLV-EXT-DATA:ephemeral":0,
         "rxtx_factor":1.0,
         "OS-FLV-DISABLED:disabled":null,
         "rxtx_quota":null,
         "rxtx_cap":null,
         "os-flavor-access:is_public":null
      },
      {
         "id":"2",
         "name":"m1.small",
         "vcpus":"1",
         "ram":2048,
         "disk":"10",
         "swap":"",
         "links":[
            {
               "rel":"self",
               "href":"http://cloud.lab.fi-
ware.eu:8774/v2/00000000000000000000000000001807/flavors/2",
               "type":null
            },
            {
               "rel":"bookmark",
               "href":"http://cloud.lab.fi-
ware.eu:8774/00000000000000000000000000001807/flavors/2",
               "type":null
            }
         ],
         "public":null,
         "OS-FLV-EXT-DATA:ephemeral":20,
         "rxtx_factor":1.0,
         "OS-FLV-DISABLED:disabled":null,
         "rxtx_quota":null,
         "rxtx_cap":null,
         "os-flavor-access:is_public":null
      },
      {
```

```
          "id":"3",
          "name":"m1.medium",
          "vcpus":"2",
          "ram":4096,
          "disk":"10",
          "swap":"",
          "links":[
             {
                "rel":"self",
                "href":"http://cloud.lab.fi-
ware.eu:8774/v2/00000000000000000000000000001807/flavors/3",
                "type":null
             },
             {
                "rel":"bookmark",
                "href":"http://cloud.lab.fi-
ware.eu:8774/00000000000000000000000000001807/flavors/3",
                "type":null
             }
          ],
          "public":null,
          "OS-FLV-EXT-DATA:ephemeral":40,
          "rxtx_factor":1.0,
          "OS-FLV-DISABLED:disabled":null,
          "rxtx_quota":null,
          "rxtx_cap":null,
          "os-flavor-access:is_public":null
       }
   ]
}
```


2.2.9 Installation Manual

This section provides a step-by-step guide for the installation and configuration of the required software components in order to setup the DCA component in a particular node in XIFI federation.

In particular, guided manuals are foreseen for:

- Installation and configuration of Apache Tomcat,
- Installation and configuration of MySQL Server,
- Installation of the DCA binaries.

Additionally, configuration instructions are given for all the aforementioned software components.

Prerequisites

For the node that will accommodate the DCA (XIFI federation platform):

- CentOS (6.2 and above) is already installed

- Oracle Java Runtime Environment (1.7.0 and above) is already installed
- Apache Tomcat (7.0.35 and above)
- The cloud-hosting Generic Enablers (GEs) that support the deployment and configuration of the GEs. Namely the PaaS Manager and the SDC.
- A persistency server (DCA currently tested with MySQL DB Server)

For the Infrastructure Node:

- The DCRM GE or OpenStack Grizzly

Installation steps

After verifying that the right version of Oracle Java has been installed, we install Apache Tomcat as follows:

```
[root@dca ~]# cd /opt
[root@dca opt]# wget http://archive.apache.org/dist/tomcat/tomcat-7/v7.0.41/bin/apache-tomcat-7.0.41.tar.gz
[root@dca opt]# tar xzf apache-tomcat-7.0.41.tar.gz
[root@dca opt]# ./apache-tomcat-7.0.41/bin/startup.sh
```

Tomcat should be up and running. Optionally, you may want to redirect the traffic of port 8080 to port 80 using the following iptables rule:

```
[root@dca opt]# iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 8080
```

Upon successfully installing Apache Tomcat, MySQL server and the respective Java JDBC connector should be also installed by issuing the following commands:

```
[root@dca opt]# yum install mysql-server mysql-connector-java
[root@dca opt]# chkconfig --levels 235 mysqld on
[root@dca opt]# service mysqld start
```

Next, place the database creation script (dca.sql) in the root directory (/root) and issue the following commands:

```
[root@dca opt]# cd ~
[root@dca ~]# mysql -uroot -p < dca.sql
```

Next, the DCA binaries (dca.war) should be uploaded to the webapps directory of the Apache Tomcat installation (in the context of the present installation guide, this should be /opt/apache-tomcat-7.0.41/webapps). Apache Tomcat should then automatically deploy the war file and extract its contents under the webapps directory. In order to enable transactions, one should initialize a keystone instance into the DCA platform. Supposing that the DCRM instance of the datacentre operator advertises its identity service at the URL http://hostname.example.com/keystone/v2.0 and is located in a region identified as A_Region, then, the following command should be issued:

```
curl http://<dca-server-ip>/dca/addEndpoint -X POST -H "Content-Type: application/json" \
       -d '{region:"A_Region", url:"http://hostname.example.com/keystone/v2.0"}'
```

The answer of the DCA server is the following:

```
{url:"http://hostname.example.com/keystone/v2.0", region:"A_Region"}
```

2.2.10 User Manual

The user manual describes the requests and responses of the methods currently offered by the DCA (version 1.0). The interaction has been performed using two trial accounts in the FI-WARE

infrastructure (one in lab.fi-ware.eu and one in testbed.fi-ware.eu, regions RegionOne and RegionTwo, respectively), as well as a local (deployed in SYNELIXIS) OpenStack environment (RegionThree). This means that many of the (proprietary) fields are currently blank (appearing as having null values).

List flavours

We request the flavours available in RegionThree.

Request

```
curl http://localhost:8080/dca/flavors?region=RegionThree
```

Response

The available flavours are retrieved (tiny, small,medium).


```
{
   "flavors":[
      {
         "id":"1",
         "name":"m1.tiny",
         "vcpus":"1",
         "ram":512,
         "disk":"0",
         "swap":"",
         "links":[
            {
               "rel":"self",
               "href":"http://cloud.lab.fi-
ware.eu:8774/v2/00000000000000000000000000001807/flavors/1",
               "type":null
            },
            {
               "rel":"bookmark",
               "href":"http://cloud.lab.fi-
ware.eu:8774/00000000000000000000000000001807/flavors/1",
               "type":null
            }
         ],
         "public":null,
         "OS-FLV-EXT-DATA:ephemeral":0,
         "rxtx_factor":1.0,
         "OS-FLV-DISABLED:disabled":null,
         "rxtx_quota":null,
         "rxtx_cap":null,
         "os-flavor-access:is_public":null
      },
      {
         "id":"2",
         "name":"m1.small",
         "vcpus":"1",
         "ram":2048,
```



```
         "disk":"10",
         "swap":"",
         "links":[
            {
               "rel":"self",
               "href":"http://cloud.lab.fi-
ware.eu:8774/v2/00000000000000000000000000001807/flavors/2",
               "type":null
            },
            {
               "rel":"bookmark",
               "href":"http://cloud.lab.fi-
ware.eu:8774/00000000000000000000000000001807/flavors/2",
               "type":null
            }
         ],
         "public":null,
         "OS-FLV-EXT-DATA:ephemeral":20,
         "rxtx_factor":1.0,
         "OS-FLV-DISABLED:disabled":null,
         "rxtx_quota":null,
         "rxtx_cap":null,
         "os-flavor-access:is_public":null
      },
      {
         "id":"3",
         "name":"m1.medium",
         "vcpus":"2",
         "ram":4096,
         "disk":"10",
         "swap":"",
         "links":[
            {
               "rel":"self",
               "href":"http://cloud.lab.fi-
ware.eu:8774/v2/00000000000000000000000000001807/flavors/3",
               "type":null
            },
            {
               "rel":"bookmark",
               "href":"http://cloud.lab.fi-
ware.eu:8774/00000000000000000000000000001807/flavors/3",
               "type":null
            }
         ],
         "public":null,
         "OS-FLV-EXT-DATA:ephemeral":40,
```

```
            "rxtx_factor":1.0,
            "OS-FLV-DISABLED:disabled":null,
            "rxtx_quota":null,
            "rxtx_cap":null,
            "os-flavor-access:is_public":null
         }
      ]
}
```


List the Keypairs

We request the user key pairs on RegionThree.

Request

```
curl http://localhost:8080/dca/keypairs?region=RegionThree
```

Response

The response is retrieved.


```
{
 "keypairs":[
      {
         "name":"panos",
         "fingerprint":"22:ec:58:0a:c5:f5:c6:d4:7b:91:64:26:5f:17:e9:8b",
         "user_id":null,
         "public_key":"ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAAAgQDBgi/eZqP7IKqygkvTn2pkrlPn3LKg57SU8jyRQNxMq37fG6RZUtfTSZsaJs0lnQ
TQvFhfuuzRXs4/9eYQ2CD82BcFOqQr6p2CyhgYzMnEv4xXIz53fUFXGqSjsvUb+YbR6fbSibl3OaXLkNhg4FQH4lGQHDEV
QEABCyagyQTuPQ== nova@gcsic001.ifca.esn",
         "private_key":null
      }
   ]
}
```


List the Security Groups

Request

The security groups available to the user on RegionThree are requested.

```
curl http://localhost:8080/dca/securitygroups?region=RegionThree
```

Response

The security groups are retrieved.


```
{
   "security_groups":[
      {
         "id":2372,
         "name":"context_broker",
```

```
        "description":"context broker security group",
        "rules":[
           {
              "id":4211,
              "name":null,
              "group":{
                 "name":null,
                 "tenant_id":null
              },
              "parent_group_id":2372,
              "from_port":1026,
              "to_port":1026,
              "ip_protocol":"tcp",
              "ip_range":{
                 "cidr":"0.0.0.0/0"
              }
           },
           {
              "id":4212,
              "name":null,
              "group":{
                 "name":null,
                 "tenant_id":null
              },
              "parent_group_id":2372,
              "from_port":8,
              "to_port":0,
              "ip_protocol":"icmp",
              "ip_range":{
                 "cidr":"0.0.0.0/0"
              }
           },
           {
              "id":4213,
              "name":null,
              "group":{
                 "name":null,
                 "tenant_id":null
              },
              "parent_group_id":2372,
              "from_port":0,
              "to_port":0,
              "ip_protocol":"icmp",
              "ip_range":{
                 "cidr":"0.0.0.0/0"
```



```
            }
         }
      ],
      "links":null,
      "tenant_id":"00000000000000000000000000001807"
   },
   {
      "id":2078,
      "name":"default",
      "description":"default",
      "rules":[
         {
            "id":3425,
            "name":null,
            "group":{
               "name":null,
               "tenant_id":null
            },
            "parent_group_id":2078,
            "from_port":22,
            "to_port":22,
            "ip_protocol":"tcp",
            "ip_range":{
               "cidr":"0.0.0.0/0"
            }
         }
      ],
      "links":null,
      "tenant_id":"00000000000000000000000000001807"
   },
   {
      "id":2079,
      "name":"monitoring",
      "description":"For FI-WARE monitoring",
      "rules":[
         {
            "id":4137,
            "name":null,
            "group":{
               "name":null,
               "tenant_id":null
            },
            "parent_group_id":2079,
            "from_port":1026,
            "to_port":1026,
```

```
                "ip_protocol":"tcp",
                "ip_range":{
                   "cidr":"0.0.0.0/0"
                }
             },
             {
                "id":4141,
                "name":null,
                "group":{
                   "name":null,
                   "tenant_id":null
                },
                "parent_group_id":2079,
                "from_port":80,
                "to_port":80,
                "ip_protocol":"tcp",
                "ip_range":{
                   "cidr":"0.0.0.0/0"
                }
             }
          ],
          "links":null,
          "tenant_id":"00000000000000000000000000001807"
       }
   ]
}
```


List Images

Request

The already available images on RegionThree are requested.

```
curl http://localhost:8080/dca/images?region=RegionThree
```

Response

The response is retrieved. The images are presented along with the related details.


```
{
   "images":[
      {
         "status":"active",
         "name":"CentOS_6.4",
         "deleted":false,
         "container_format":"bare",
         "created_at":"2013-11-04T19:13:46",
         "disk_format":"iso",
```



```
            "updated_at":"2013-11-04T19:14:01",
            "id":"79e9245c-3a02-48af-8dd2-ada0d893331e",
            "min_disk":0,
            "protected":false,
            "min_ram":0,
            "checksum":"4a5fa01c81cc300f4729136e28ebe600",
            "owner":"e9312e85dde04636a63c1b340f89242a",
            "is_public":true,
            "deleted_at":null,
            "properties":{

            },
            "size":358959104
        },
        {
            "status":"active",
            "name":"Cirros 0.3.1",
            "deleted":false,
            "container_format":"bare",
            "created_at":"2013-10-23T14:28:21",
            "disk_format":"qcow2",
            "updated_at":"2013-10-23T14:28:22",
            "id":"c4c6463f-0acb-4e06-8051-1e14070c154d",
            "min_disk":0,
            "protected":false,
            "min_ram":0,
            "checksum":"d972013792949d0d3ba628fbe8685bce",
            "owner":"e9312e85dde04636a63c1b340f89242a",
            "is_public":false,
            "deleted_at":null,
            "properties":{

            },
            "size":13147648
        },
        {
            "status":"active",
            "name":"orion",
            "deleted":false,
            "container_format":"bare",
            "created_at":"2013-11-20T18:06:47",
            "disk_format":"qcow2",
            "updated_at":"2013-11-20T18:12:22",
            "id":"791c1279-4d38-4f89-a33b-a3a93a29e75c",
            "min_disk":0,
```

```
            "protected":false,
            "min_ram":0,
            "checksum":"d60b7cfc2f87a9b69095cbd627805bf0",
            "owner":"e9312e85dde04636a63c1b340f89242a",
            "is_public":false,
            "deleted_at":null,
            "properties":{
               "instance_uuid":"a9259820-dd5e-4fb4-9581-09acaddf199b",
               "image_location":"snapshot",
               "image_state":"available",
               "instance_type_memory_mb":"2048",
               "instance_type_swap":"0",
               "instance_type_vcpu_weight":"None",
               "image_type":"snapshot",
               "instance_type_id":"5",
               "ramdisk_id":null,
               "instance_type_name":"m1.small",
               "instance_type_ephemeral_gb":"0",
               "instance_type_rxtx_factor":"1",
               "kernel_id":null,
               "instance_type_flavorid":"2",
               "instance_type_vcpus":"1",
               "user_id":"752627b3561f46b08bc27726ce2630ce",
               "instance_type_root_gb":"20",
               "base_image_ref":"79e9245c-3a02-48af-8dd2-ada0d893331e",
               "owner_id":"e9312e85dde04636a63c1b340f89242a"
            },
            "size":358875136
         }
      ]
}
```


List Networks

Request

The networks available to the user on RegionThree are requested.

```
curl http://localhost:8080/dca/networks?region=RegionThree
```

Response


```
{
   "networks":[
      {
         "status":"ACTIVE",
```



```
            "subnets":[
               "3b873329-6469-4d44-9814-93be7b6d7eb2"
            ],
            "name":"net-0",
            "id":"a714bca6-7f2d-4181-91fb-44e6b603a7e4",
            "shared":"false",
            "provider: physical_network":null,
            "admin_state_up":true,
            "tenant_id":"e9312e85dde04636a63c1b340f89242a",
            "provider: network_type":"gre",
            "router: external":"false",
            "provider: segmentation_id":"1"
         },
         {
            "status":"ACTIVE",
            "subnets":[
               "ed1fa327-81ba-40ca-b523-aa64e45ba091"
            ],
            "name":"ext_net",
            "id":"c3a72055-71ac-4cc7-afad-4869dc602b19",
            "shared":"false",
            "provider: physical_network":null,
            "admin_state_up":true,
            "tenant_id":"e9312e85dde04636a63c1b340f89242a",
            "provider: network_type":"gre",
            "router: external":"true",
            "provider: segmentation_id":"2"
         }
      ]
}
```


List Servers

Request

The already deployed (in total) GEs are requested.

```
curl http://localhost:8080/dca/servers/ge
```

Response

The response is retrieved. The servers are presented along with the related details.


```
[
   {
      "id":"6fda6be8-0e70-4d08-9731-c858d6a27f50",
      "name":"test-xifi-proton",
      "imageRef":"90d4865d-5e7b-4d95-af2c-69753e1740d6",
```

```
      "flavorRef":"2",
      "keyName":"xifi",
      "securityGroups":"default,cep",
      "created":1390051880000,
      "status":null,
      "tenantId":"00000000000000000000000000000101",
      "userId":"artemis-voulkidis",
      "region":"RegionOne"
   },
   {
      "id":"a335265d-777e-42fa-8f5a-355160bc93cc",
      "name":"test-xifi-proton-r1",
      "imageRef":"90d4865d-5e7b-4d95-af2c-69753e1740d6",
      "flavorRef":"2",
      "keyName":"xifi",
      "securityGroups":"default,cep",
      "created":1390135132000,
      "status":null,
      "tenantId":"00000000000000000000000000000101",
      "userId":"artemis-voulkidis",
      "region":"RegionOne"
   },
   {
      "id":"29c1ee5e-fac8-451d-85d9-963ed7ae2386",
      "name":"test-xifi-orion",
      "imageRef":"02fdb0bc-6b47-4af4-ab13-95508033cdb4",
      "flavorRef":"2",
      "keyName":"xifi",
      "securityGroups":"default",
      "created":1390135136000,
      "status":null,
      "tenantId":"00000000000000000000000000000158",
      "userId":"artemis-voulkidis",
      "region":"RegionTwo"
   },
   {
      "id":"1ae2d8d8-e8ed-4e4e-88bd-4209b808eeda",
      "name":"test-xifi-proton-r2",
      "imageRef":"7b001833-5eaa-4a4d-84fa-803e3c59377d",
      "flavorRef":"2",
      "keyName":"xifi",
      "securityGroups":"default,cep",
      "created":1390052553000,
      "status":null,
      "tenantId":"00000000000000000000000000000158",
```

```
      "userId":"artemis-voulkidis",
      "region":"RegionTwo"
   }
]
```


By issuing relevant API calls containing HTTP parameters more complex queries are also supported. For example, the CEP instances deployed in RegionOne could be discovered as follows:

Request

```
curl http://localhost:8080/dca/servers/ge?desc=cep&region=RegionTwo
```

Response


```
[
   {
      "id":"1ae2d8d8-e8ed-4e4e-88bd-4209b808eeda",
      "name":"test-xifi-proton-r2",
      "imageRef":"7b001833-5eaa-4a4d-84fa-803e3c59377d",
      "flavorRef":"2",
      "keyName":"xifi",
      "securityGroups":"default,cep",
      "created":1390052553000,
      "status":null,
      "tenantId":"00000000000000000000000000000158",
      "userId":"artemis-voulkidis",
      "region":"RegionTwo"
   }
]
```


Additionally, the DCA component supports active VM (GE) listing, where the data is requested directly by the DCRM instances of the federation. For example, an infrastructure owner may be interested on checking the GEs that are deployed on its own infrastructure, located at RegionOne:

Request

```
curl http://localhost:8080/dca/servers/live?region=RegionOne
```

The response of the DCA module could be as follows (the full listing has been suppressed for reasons of brevity):

Response


```
{
   "list":[
      {
         "id":"a335265d-777e-42fa-8f5a-355160bc93cc",
         "name":"test-xifi-proton-r1",
         "addresses":{
```



```
            "addresses":{
               "private":[
                  {
                     "macAddr":null,
                     "version":"4",
                     "addr":"10.0.1.202",
                     "type":null
                  }
               ]
            }
         },
         "links":[
            {
               "rel":"self",
               "href":"http://cloud.lab.fi-
ware.eu:8774/v2/00000000000000000000000000000101/servers/a335265d-777e-42fa-8f5a-
355160bc93cc",
               "type":null
            },
            {
               "rel":"bookmark",
               "href":"http://cloud.lab.fi-
ware.eu:8774/00000000000000000000000000000101/servers/a335265d-777e-42fa-8f5a-355160bc93cc",
               "type":null
            }
         ],
         "image":{
            "id":"90d4865d-5e7b-4d95-af2c-69753e1740d6",
            "links":[
               {
                  "rel":"bookmark",
                  "href":"http://cloud.lab.fi-
ware.eu:8774/00000000000000000000000000000101/images/90d4865d-5e7b-4d95-af2c-69753e1740d6",
                  "type":null
               }
            ]
         },
         "flavor":{
            "id":"2",
            "links":[
               {
                  "rel":"bookmark",
                  "href":"http://cloud.lab.fi-
ware.eu:8774/00000000000000000000000000000101/flavors/2",
                  "type":null
               }
            ],
```

```
            "public":null
         },
         "accessIPv4":"",
         "accessIPv6":"",
         "configDrive":"",
         "status":"ACTIVE",
         "progress":0,
         "fault":null,
         "tenantId":"00000000000000000000000000000101",
         "userId":"artemis-voulkidis",
         "keyName":"xifi",
         "hostId":"c0d2f2faa797bfb8be05680a6d2f31ba4553f7219e35a13011cb22d1",
         "updated":"2014-01-19T12:39:05Z",
         "created":"2014-01-19T12:38:52Z",
         "metadata":{
         },
         "powerState":"1",
         "vmState":"active",
         "host":"gcsic022.ifca.es",
         "instanceName":"instance-00000ad2",
         "hypervisorHostname":null,
         "diskConfig":"MANUAL",
         "availabilityZone":null,
         "uuid":null,
         "adminPass":null
      }
   ]
}
```


Combinatorial requests are also supported. For example, assume that it is of interest to know on which XIFI nodes have instances of CEP and Marketplace GEs simultaneously deployed:

Request

```
curl http://localhost:8080/dca/servers/ge/multiple?desc1=cep&desc2=marketplace
```

Assuming that in the present state of the XIFI federation, only RegionOne has these two GEs deployed, the answer would have been:

Response

```
[
   "RegionOne"
]
```

Single Server Deployment

Request

In this request we ask for the deployment of a single CEP instance on RegionOne. Necessary information is provided through the POST method. The response is condensed for reasons of brevity.

```
curl http://localhost:8080/dca/servers -X POST -H "Content-Type: application/json" -d '{
         "name":"test-3",
         "imageRef":"90d4865d-5e7b-4d95-af2c-69753e1740d6",
         "flavorRef":"2",
         "keyName":"xifi",
         "securityGroups":[
            {
               "name":"default"
            },
            {
               "name":"cep"
            }
         ],
         "region":"RegionOne"
      }'
```

Response

The response is retrieved and it shows that the server is created.

```
{
   "id":"ffae6295-9aba-48a8-908d-4eecb71bd79c",
   "name":null,
   "addresses":null,
   "links":[
      {
         "rel":"self",
         "href":"http://cloud.lab.fi-
ware.eu:8774/v2/00000000000000000000000000000101/servers/ffae6295-9aba-48a8-908d-
4eecb71bd79c",
         "type":null
      },
      {
         "rel":"bookmark",
         "href":"http://cloud.lab.fi-
ware.eu:8774/00000000000000000000000000000101/servers/ffae6295-9aba-48a8-908d-4eecb71bd79c",
         "type":null
      }
   ],
   "diskConfig":"MANUAL",
   "adminPass":"hRmXkm97DRqY"
}
Multiple Servers Deployment Request
In this request we ask for the deployment of a CEP instance, destined for RegionOne, and an
Orion CPB instance, destined for RegionTwo.
curl http://localhost:8080/dca/multiple -X POST -H "Content-Type: application/json" -d '{
```



```
   "list":[
      {
         "name":"test-xifi-proton-r1",
         "imageRef":"90d4865d-5e7b-4d95-af2c-69753e1740d6",
         "flavorRef":"2",
         "keyName":"xifi",
         "securityGroups":[
            {
               "name":"default"
            },
            {
               "name":"cep"
            }
         ],
         "region":"RegionOne"
      },
      {
         "name":"test-xifi-orion-r2",
         "imageRef":"02fdb0bc-6b47-4af4-ab13-95508033cdb4",
         "flavorRef":"2",
         "keyName":"xifi",
         "securityGroups":[
            {
               "name":"default"
            }
         ],
         "region":"RegionTwo"
      }
   ]
}'
```


Response

The response is retrieved and proves the creation of the two requested servers (VMs).


```
[
   {
      "id":"a335265d-777e-42fa-8f5a-355160bc93cc",
      "name":null,
      "addresses":null,
      "links":[
         {
            "rel":"self",
            "href":"http://cloud.lab.fi-
ware.eu:8774/v2/00000000000000000000000000000101/servers/a335265d-777e-42fa-8f5a-
355160bc93cc",
            "type":null
```

```
            },
            {
               "rel":"bookmark",
               "href":"http://cloud.lab.fi-
ware.eu:8774/00000000000000000000000000000101/servers/a335265d-777e-42fa-8f5a-355160bc93cc",
               "type":null
            }
         ],
         "diskConfig":"MANUAL",
         "adminPass":"ZGs4QxAvZc8c"
      },
      {
         "id":"29c1ee5e-fac8-451d-85d9-963ed7ae2386",
         "name":null,
         "addresses":null,
         "links":[
            {
               "rel":"self",

"href":"http://130.206.80.11:8774/v2/00000000000000000000000000000158/servers/29c1ee5e-fac8-
451d-85d9-963ed7ae2386",
               "type":null
            },
            {
               "rel":"bookmark",

"href":"http://130.206.80.11:8774/00000000000000000000000000000158/servers/29c1ee5e-fac8-451d-
85d9-963ed7ae2386",
               "type":null
            }
         ],
         "diskConfig":"MANUAL",
         "adminPass":"P8inbZ7LERa8"
      }
]
```


Delete Server

Request

We request the deletion of a specific server.

```
curl -X DELETE http://localhost:8080/dca/servers/d5c22170-38d6-4344-9d62-c9770e550cee
```

Response

The response is retrieved and shows the deletion of the server.


```
{
        "deleted":"d5c22170-38d6-4344-9d62-c9770e550cee"
```



```
}
```

Note that the region of the server is inferred through the DCA persistency layer, described later.

2.2.11 Developer Guide

The code of DCA is available in the SVN repository of the project in the WP3, DCA software trunk (https://xifisvn.res.eng.it/wp3/software/DCA).

DCA API is documented here: http://docs.dca.apiary.io/

The project can be built and managed using the Apache Maven software project management tool (http://maven.apache.org/index.html). The steps are:

- The developer retrieves the POM file that describes the build procedure of OpenStack Java SDK and places it in the parent directory of DCA. The POM is retrieved from: https://raw.github.com/woorea/openstack-java-sdk/master/pom.xml
- Then uses maven for building the project.

The commands are the following:

- wget https://raw.github.com/woorea/openstack-java-sdk/master/pom.xml
- cd dca/
- mvn --also-make --projects dca install

The dca.war file is then created in the target directory.

3 CONCLUSIONS

This document has described the first version of the design and development of tools and adapters to facilitate the management of the XIFI federation and FI enablers.

The first component, ITBox, provides a toolbox that eases and automatizes the process of deploying the appropriate tools and GEs to a bare-metal infrastructure, becoming fully compliant to the XIFI federation requirements. On the other hand, the second component, DCA, provides a management layer that caters for the deployment of multiple GEs to be utilized by application developers, offering collected data to the federation and thus the interested users.

These components have been developed based on the tools which has been described in the sections of this deliverable and offered as autonomous components to interested XIFI partners, infrastructure owners and application developers through FI-Ops. Apart from the source code, the components described herein provide extended documentation for the installation, testing and extension of provided APIs in the XIFI repository.

All in all, the achievements of this deliverable can be summarized as follows:

- ITBox will significantly assist XIFI sustainability by offering a toolbox that will take care of the processes required for a new (bare-metal) infrastructure owner to become IaaS/PaaS compatible with the XIFI federation infrastructure.
- DCA will support the extensibility of XIFI infrastructure and simplify the management of GEs deployment and configuration by providing APIs and data on federation level that is of high interest of application developers.

The development of the ITBox and DCA components is progressing according to release plan. This document describes the developments included in version 1.0 of the components. The plan of future developments and extensions to ITBox and DCA can be found at http://wiki.fi-xifi.eu/Xifi:Wp3:Components:InfrastructureToolbox and http://wiki.fi-xifi.eu/Xifi:Wp3:Components:DCA.

Due to the living nature of the FI-WARE components (i.e. the functional enhancement to the existing GEs), we should continuously monitor and adjust the specifications of the components in order to guarantee their added value and seamless interoperability.

REFERENCES

[1] Software Deployment and Configuration (SDC) GE - https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Cloud.SDC

[2] Fuel by Mirantis - http://fuel.mirantis.com/

[3] Tuskar - http://github.com/tuskar/tuskar

[4] Foreman - http://theforeman.org/

[5] Rackspace Private Cloud - http://www.rackspace.com/knowledge_center/product-page/rackspace-private-cloud

[6] Puppet Labs - http://puppetlabs.com/puppet/what-is-puppet

[7] Fuel command line - http://docs.mirantis.com/fuel/fuel-3.2.1/user-guide.html#understanding-environment-deployment-with-fuel-cli

[8] XIFI Deliverable D3.1 "XIFI infrastructure adaptation components API open specification", http://wiki.fi-XIFI.eu/Public:D3.1

[9] Apache Cloudstack, http://cloudstack.apache.org/

[10] Chef configuration management tool, http://www.getchef.com/chef/

[11] P. Venezia "Review: Puppet vs. Chef vs. Ansible vs. Salt", http://www.infoworld.com/d/data-center/review-puppet-vs-chef-vs-ansible-vs-salt-231308

[12] Ansible configuration management tool, http://www.ansibleworks.com/

[13] Salt Open Source Software Project, http://www.saltstack.com/

[14] CloudBees Platform as a Service, http://www.cloudbees.com

