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

eXperimental Infrastructures for the Future Internet

D2.4: XIFI Handbook v2

Revision: v1.6

Work package	WP 2
Task	Task 2.1
Due date	30/09/2014
Submission date	10/10/2014
Deliverable lead	ENG
Authors	Maria Antonietta Di Girolamo (ENG), Ian Millis (WIT), Susana Gonzales (ATOS), Joaquin Iranzo Yuste (ATOS), Silvio Creti (CNET), Alessandro Martellone (CREATE-NET), Cyril Dangerville (THALES), Daniel Nehls (TUB), Jose Gonzales (UPM-SSR), Alvaro Alonso (UPM-DIT)
Reviewers	Riwal Kerherve (ILB), Glenn Wearen (Heanet)

Abstract	The XIFI Handbook describes the processes and tools for the Installation and configuration of a new XIFI node from the infrastructure owner stand point, including the procedures and the tools of the FIWARE services Configuration and FIWARE app development.
Keywords	OpenStack, ITBox, sub-systems, XIFI node, deployement, federation, XIFI components, set-up, configuration and installation.

Document Revision History

Version	**Date**	**Description of change**	**List of contributor(s)**
V0.0	09.09. 2014	First completion of deliverable	Maria Di Girolamo (ENG), Ian Millis (WIT), Susana Gonzales (ATOS), Joaquin Iranzo Yuste (ATOS), Silvio Cretti (CNET), Cyril Dangerville (THALES), Daniel Nehls(TUB), Jose Gonzales (UPM-SSR), Alvaro Alonso (UPM-DIT)
V1.0	15-09-2014	Added section “Integration and Validation check”	Maria Di Girolamo (ENG)
V1.1	15-09-2014	Updated ”Security Probe (SIEM agent)” section	Susana Gonzales (ATOS)
V1.1	15-09-2014	Integrated Quick Onli Test	Ian Millis (WIT)
V1.1	15-09-2014	Updated “How a region will deploy a FI services”	Joaquin Iranzo Yuste (ATOS)
V1.1	15-09-2014	Updated “Federation Access control GE”	Cyril Dangerville (THALES)
V1.2	16-09-2014	Integrated the sections about the set-up and test installation processes for the Federation Monitoring and OpenStackDataCollector components	Silvio Cretti (CREATE-NET)
V1.2	16-09-2014	Updated the sections set-up and installation and validation check for the “DCRM GE” and “Object storage GE” components	Alessandro Martellone (CREATE-NET)
V1.2	16-09-2014	Updated “Installation and Validation check” for Registration at Cloud Portal and Federation Manager component.	Daniel Nehls (TUB)
V1.2	16-09-2014	Updated the section “Installation and Validation check” for Quick Online Test component.	Ian Millis (WIT)
V1.2	17-09-2014	Updated the section XIMM monitoring (functional description and set-up processes). Added for this component the section Installation and Validation check	Jose Gonzales (UPM-SSR)
V1.2	17-09-2014	Updated section “How a region will deploy a FI services”.	Joaquin Iranzo Yuste (ATOS)

Version	**Date**	**Description of change**	**List of contributor(s)**
		Added section “Installation and validation check” for the Resource Catalogue and SLA Manager	
V1.3	23-09-2014	Added functional description for Federation Manager after internal review	Daniel Nehls (TUB)
V1.3	25-09-2014	Added functional description for GUI-Portal and update the SLA Manager, Cloud Portal , Security Dashboard and “How a region will deploy a FI services” section after internal review	Susan Gonzales
V1.3	25-09-2014	Updated the manual installation of a new node (implementation of internal reviewer’s comments)	Maria Di Girolamo (ENG)
V1.3	25-09-2014	Updated the section Federation Access Control (implementation of internal reviewer’s comments)	Cyril Dangerville (THALES)
V1.3	26-09-2014	Updated the section Interoperability Tool and Deployment and Configuration Adapter (implementation of internal reviewer’s comments), PaaS and SDC GE	Maria Di Girolamo (ENG), Pablo Rodriguez Archilla (TID), Paul Grace (IT-INNOV), Panos Trakadas (SYNELIXIS)
V1.3	26-09-2014	Updated the section related the functional description for Security Proxy and Monitoring Dashboard and their installation and configuration on a new XIFI node (implementation of internal reviewers’ comment)	Alvaro Alonso (UPM-DIT)
V1.3	26-09-2014	Deleted the section related to the Quick Online Test,since it was integrated within the Federation Manager Component	Ian Millis (WIT), Maria Di Girolamo (ENG)
V1.4	29-09-2014	Overall final check	Laura Pucci (ENG)
V1.5	30-09-2014	Implementation of the comments after the final check by Laura Pucci (ENG)	Maria Di Girolamo(ENG), Susana Gonzales(ATOS)
V1.6	10-10-2014	Final Version	

Project co-funded by the European Commission in the 7th Framework Programme (2007-2013)		
Nature of the Deliverable:	R	
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

This document provides the final version of XIFI Handbook (D2.1 – XIFI Handbook v1) [4].

The XIFI Federation identifies two types of roles for nodes:

- Master node including also federation services ;
- Slave node including only local resource management.

The scope of this final version of the Handbook is to provide the complete procedure for the installation of XIFI both slave and master node and how to set up the connectivity with the XIFI Federation. The main changes with respect to the previous version of this Handbook are:

- To include and complete the list of components, and describe procedure to set up a new XIFI node (slave node);
- To identify the list of components and describe their procedures to set up a new XIFI node (master node);
- To describe how a region deploys the needed FI Services, including how to publish the SE and the non-conventional services (these parts were not included in the previous XIFI Handbook).

This Deliverable is the result of a cross WPs/Tasks cooperation[2]:

- In WP2 were identified the software package components, the federation components, and first results of test components (D2.3- Test and Validation report v1[6]);
- In WP3 were identified the description of monitoring components[9][9] and the node deployment tools and their usage;
- WP4 provided the description of marketplace components[11], baseline service[12] and tools for the XIFI federation[11];
- WP1 provided the component for the Use Cases and the identification of the model logical architecture (as described in the D1.5 – Federated Platform Architecture Version 2 [3][3]);
- WP5 provided, in collaboration with WP2, the definitions of sub-system, the definition of joining federation and the resolution of some issues [D5.4 – XIFI federation extension and support[16]and Appendix].

The architecture of XIFI relies on a number of FIWARE GEs, thus, part of the Handbook is based on the FIWARE GEs documentation (with a dedicated section in this Deliverable). Rather than duplicating information related to each component, which was already published in related documentation, the Handbook provides the overall picture and the references to that documentation.

Intended audience for the handbook includes: all users that for the first time approach the installation of a new XIFI node, all XIFI users (as XIFI infrastructure services owners, stakeholders) and all developers that provide FIWARE Generic and Specific Enablers.

This documentation provides all information on how to setup a) a slave node (cf. section Installing XIFI node (Slave Node)), b) a master node (cf. section Installing a XIFI Node (Master Node)) and the documentation that the user should be read before to start with the setup of a new XIFI node (cf. -FAQ).

For the definition of slave and master node refer to the section IntroductionScope of this document.

[2] Note: The partners involved both WP2 and in other technical WPs as WP3, WP4 , WP5

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

CI	Continuous Integration
DCA	Deployment and Configuration Adapter
DCRM	Datacenter Resource Management Generic Enabler
DEM	Data Center & Enablers Monitoring
FI	Future Internet
FM	Federation Manager
GEs	Generic Enablers
HA	High Availability
IdM	Identity Management
IO	Infrastructure Owner
IP	Internet Protocol
FI	Future Internet
FM	Federation Manager
FM App	Federation Manager Application
FAQ	Frequently Asked Questions
N/A	Not Available
NAM	Network Active Monitoring
NPM	Network Passive Monitoring
PaaS	Platform as a Service
PXE	Preboot eXecution Environment
QOT	Quick Online Test
SaaS	Software As A Service
SDC	Software Deployment and Configuration
SE	Specific Enabler
SLA	Service Level Agreement
SSO	Single Sign On
SVN	Version Control System
TCP	Transmission Control Protocol
TUI	Text User Interface
UI	User Interface
WP	WorkPackage
XIMM	XIFI Infrastructure Monitoring Middleware

1 INTRODUCTION

1.1 Scope

The scope of the document is:

- To update the procedures – already described in the previous Handbook version - for installing and deploying a new XIFI node (slave role) ;
- To describe the same processes for a master node too;
- To provide the procedures and tools for the configuration of FIWARE services and for FIWARE apps development.

A slave node provides three main categories of components grouped for their functionalities: a) cloud computing, b) monitoring and c) security (refer to D1.5 – Federated Platform Architecture Version2 - Architecture of a generic XIFI Node [3]).

A master node is a slave node with the following additional categories of components grouped for their functionalities: a) User oriented services and tools, b) Service and tools supporting the setup, deployment and operation of the Federation and c) Federation Security tools (refer to D1.5 – Federated Platform Architecture Version2 - Architecture of a XIFI Master Node [3])

For a definition of master and slave node refer to D1.1.b – XIFI Core Concepts, Requirements and Architecture draft [1].

The logical architecture model (figure below) provides the list of components for the setup of a new XIFI generic node (slave and master node). This model is defined within WP1 and described in the deliverable D1.5 – Federated Platform Architecture Version 2 [3].

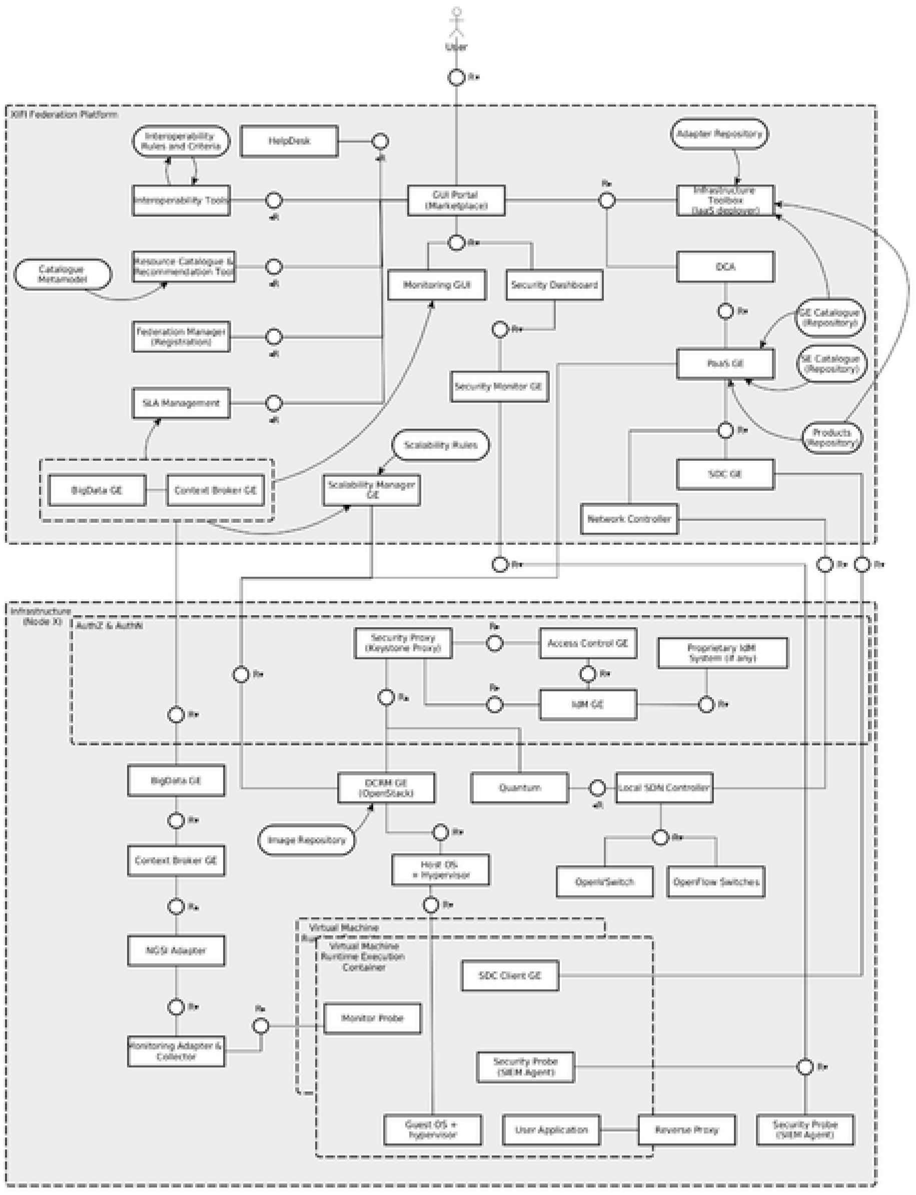


Figure 1: Logical Architecture for XIFI components

The XIFI components described in this XIFI Handbook are:

- Infrastructure Toolbox
- XIFI Infrastructure Monitoring Middleware (XIMM) included NAM adapter, NPM adapter, DEM adapter
- DCRM GE,
- Object Storage GE,
- Identity Management GE,
- Federation Monitoring,
- Federation Access Control GE,
- Resource Catalogue and Recommendation Tool,
- Security Dashboard

- Security Monitoring
- Security Probe
- Security Proxy
- Quick Online Test
- Federation Manager
- Infographics and Status pages
- Cloud Portal
- Monitoring Dashboard
- Deployment and Configuration Adapter
- Interoperability Tools
- PaaS GE
- SDC GE

1.2 Intended audience

The intended audience of this document includes:

- users of the XIFI federation;
- users of the XIFI infrastructure services;
- stakeholders that provide FIWARE Generic and Specific Enablers as well as FI-PPP Use Cases;
- infrastructure administrators and owners approaching to the XIFI federation.

1.3 How to read

This Handbook is structured as follows:

1. *Chapter One* provides information on the scope of the document, to which this manual is useful, on how to read this manual and what are the changes introduced with respect to the previous release.
2. *Chapter Two* provides the description of the XIFI processes to be followed by a new user who wants to participate or to become a member of the XIFI federation. These processes are identified as: "join the federation" and "deploy the needed FI services".
3. *Chapter Three* describes at a high level the functionalities of all needed XIFI components, including those components not yet described in the previous release (as the GUI component). The list of components identified is based on:
    - The logical architecture model, as described in the deliverable D1.5 – Federated Platform Architecture Version 2 - Updates on Federation Architecture [3],
    - Sub-systems definition as described in the deliverable D2.3 – Test and Validation Report v1- Chapter 2 Test purposes – paragraph 2.3 Testable sub-systems[6].
    - How all components interact with each other, more details about how the XIFI components interact among them are reported in D2.5, APIs and Tools for Infrastructure v2 – Chapter 2 The notion of sub-systems in XIFI – paragraph 2.1 Definition of sub-systems, table "Sub-systems under consideration" provided in the D2.5 – APIs and Tools for Infrastructure v2 Chapter 2 The notion of sub-systems in

XIFI – paragraph 2.2 Updates on Federation Architecture[7].

4. *Chapter Four* is the core of this document .It described steps by steps, component per component how to install a new XIFI node: slave and master node (to difference from slave and master node refer to Introduction - Scope page 12 of this manual.
5. *The Annex* provides additional information with a F.A.Q. section that collects the questions (and answers) coming from IO registered during the XIFI lifecycle to the project RedMine tool and related to the setup of a new XIFI node.

1.4 Changes and additions in the new version

Basically the main changes with respect to the previous version of XIFI Handbook [4] are:

1. The procedures of installation, configuration and deployment of the slave node were updated and same procedures were added for the master node (the previous version provided these procedures only for a slave node). The list of these components is in the session "XIFI component- functional description"(Chapter 3), table "Table 1: List of XIFI component.
2. It was introduced, as example, a real use case process describing the process of joining the federation and FI services, based on real experience of a new user that joined the XIFI federation.
3. A FAQ section was introduced in Appendix A, providing information about general or technical questions (including possible issues and their resolution). Those F.A. Questions are based on the real experience of XIFI partners (IO, developer Owners and new users) by collecting tasks registered by means of the RedMine project tool.

2 XIFI PROCESSES – USE CASES

This section describes the XIFI processes, adopted by Work Package 4 and Work Package 5, about how a region can join the federation and deploy the needed FI services. The following XIFI processes are identified:

- Joing the federation;
- Deploy the needed FI services (included SE and non-conventional services).

2.1 Joining the federation

The process to join the federation can be split into three sequential blocks:

- Registration at Cloud Portal;
- Infrastructure Setup with the Infrastructure Toolbox;
- Finalization of Registration at the Cloud Portal.

The first block describes the initial registration process for an Infrastructure Owner interested in participating in the XIFI Federation. The second block concerns the installation and configuration of the needed software stack enabling the infrastructure to take part in the XIFI federation. Finally, the third block explains the process of remote verification and setup of the services that the node wants to expose to the members of the federation.

2.1.1 Step One: Registration at Cloud Portal

When a non registered infrastructure owner (IO) visits the Federation Manager website (FM GUI) will be asked if the IO and the organization he/she is acting on behalf on are already registered to the federation identity management system (IDM).

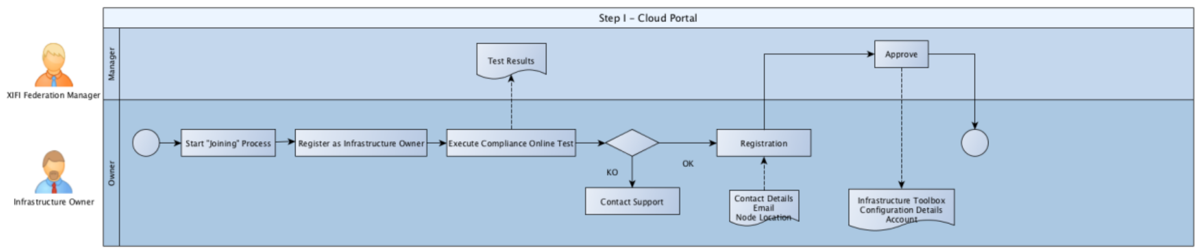


Figure 2: Registration Process – Step One

When denied, the visitor is redirected to the IDM to complete this mandatory registration. After registration at the IDM, the IO has to answer the Quick Online Test.

During the test the IO has to provide information regarding legal, technical and operational compliance, as well as information about the organization, which will operate the node.

The data is transmitted to the Federation Manager (FM), who can then add the organization to the list of authorized entities for the Federation Manager application (FM App).

The node owner will then be informed and can proceed in the process. The IO can now login to the FM GUI, add information about the infrastructure and download the Infrastructure Toolbox software package.

2.1.2 Step Two: Infrastructure Setup

This process describes the initial software installation and setup of an infrastructure by means of utilization of the Infrastructure Toolbox. In the first step of the installation process, the OpenStack layer is deployed on the hardware.

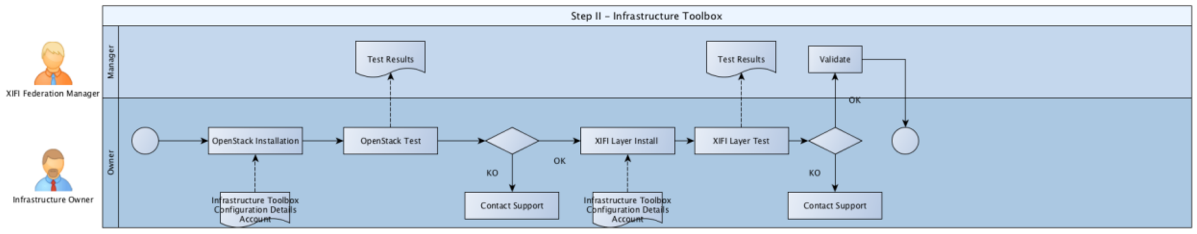


Figure 3: Registration Process – Step Two

After the installation, a verification test is run to ensure proper functioning of the basic OpenStack. If the test passes, the IO can proceed with the installation of the XIFI specific software and this is done by means of the Infrastructure Toolbox as well. The XIFI software layer includes the XIFI adapters and the basic Federation GEs, such as PaaS, SDC and so forth.

Again, after installation a test procedure is executed to ensure proper function of the newly installed XIFI software layer. The results of the tests are transmitted to the FM, who will validate them.

2.1.3 Step Three: Finalization of Registration

This process describes the enablement of services by the new node.

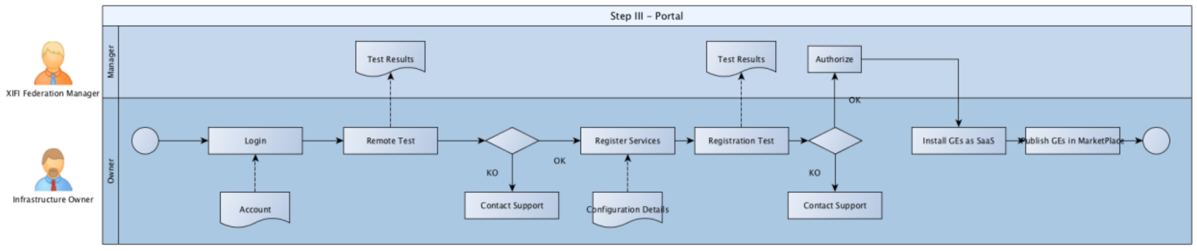


Figure 4: Registration Process – Step Three

When logging into the portal the first time, after the software installation process, the IO has to trigger remote connectivity and functionality tests. If these tests pass, the node is ready to deploy services for the members of the XIFI federation.

The IO has to provide configuration details for the services that shall be exposed. Prior to be published to the Federation Marketplace these services are tested remotely to verify the interface compliance (in case of GE services) and connectivity. After the authorization by the FM the services are installed as SaaS and published to the Marketplace.

2.2 How a region will deploy the needed FI services

Once a Future Internet (FI) Developer agrees on XIFI Federation terms and conditions, the following procedure must be followed to select the resources required by the developer from the ones offered by the infrastructure owners and deploy experimenting innovative applications in the XIFI platform.

This procedure can be summarized with three steps:

1. Single Sign On in the XIFI Portal;

2. Browse Resource Catalogue;
3. Selection and deployment of resources.

The XIFI components involved in this process are the following:

- Resource Catalogue and Recommender
- Federation Monitoring
- SLA Manager
- Cloud Portal
- Federated Identity Management

More roles were also introduced, in order to manage these processes. The following roles can be considered, which interact with the Marketplace and the Resource catalogue:

- FI Developers: this role is assigned to users that want to interact with the XIFI Portal to find and use the services. They have to be registered in the IdM.
- Infrastructure Owners: this role identifies the nodes which want to be federated. They can publish both the specific enablers and the non-conventional services. They have to be identified by the IdM as organization and validated by the federation.
- Software providers: this role identifies organizations that want to expose their own specific enablers, for example SME, Technological Providers, Use Case Projects. They cannot publish non-conventional service since only the Infrastructure Owner can provide this kind of resources. They must be identified by the IdM as organizations.
- Federator: this role indicates the XIFI federation that is composed by all those responsible to maintain the federated platform.

2.2.1 Step I: Single Sign On in the XIFI Portal

Any FI developer who wants to make use of the capabilities offered by the XIFI platform needs to authenticate through the XIFI portal.

The access is Single Sign On (SSO), so once logged into the platform, the developer will not be required to introduce again his/her credentials, which might also be associated to privacy/business terms, in order to access to other administrative domains. Hence, the authentication of all the users, who want to use the federation environment, is delegated to the Federated Identity Management.

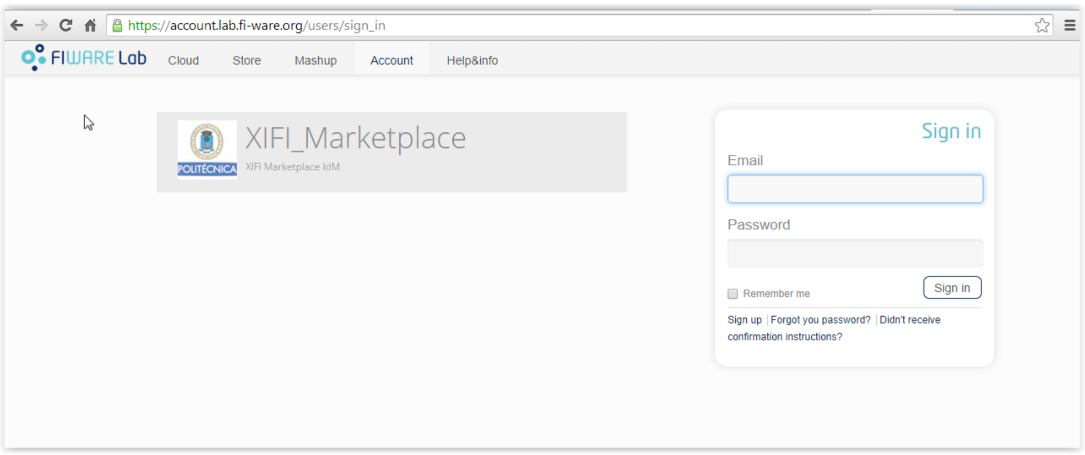


Figure 5: Login of the XIFI Portal

After the identification, the FI developers can see and use the available services in the federation. The user has been identified and the XIFI Portal shows the user the initial page.

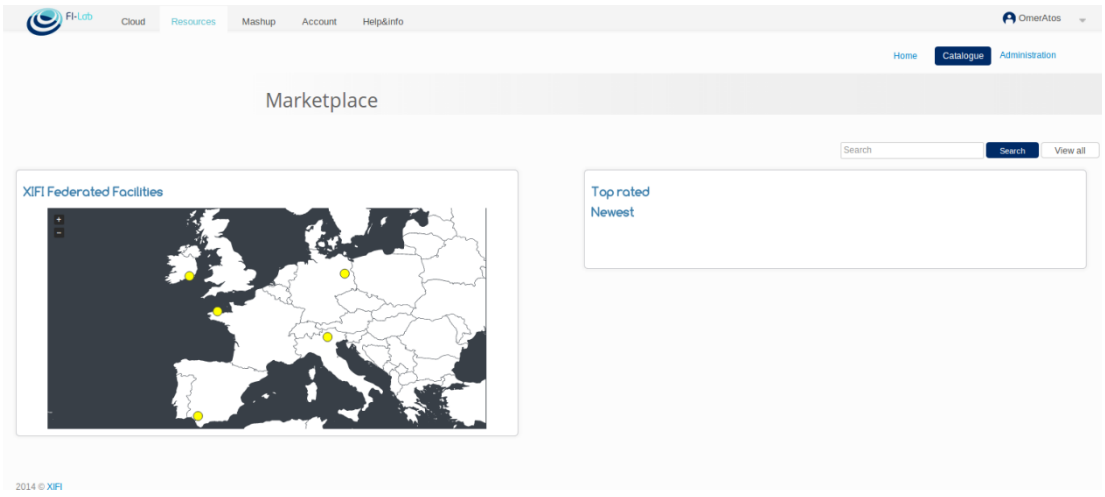
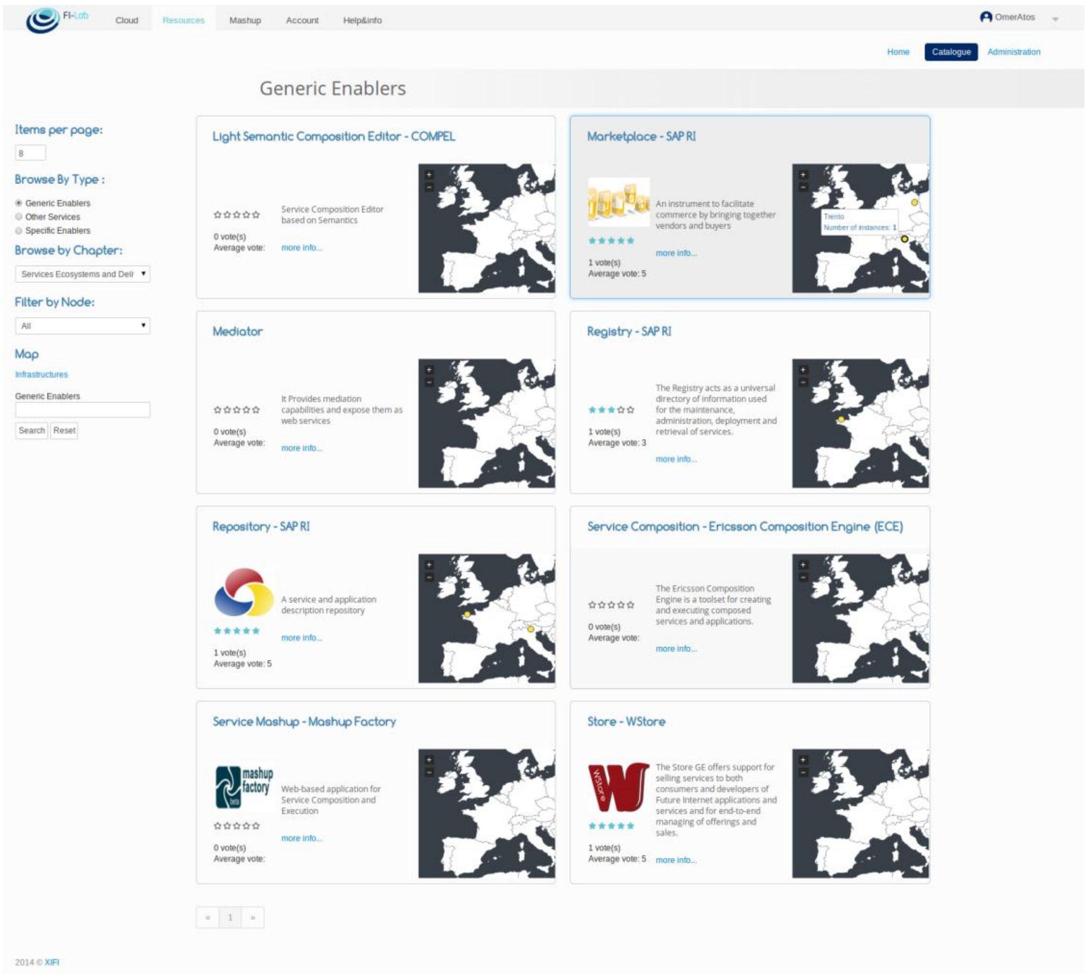


Figure 6: Initial page after the user has been identified

2.2.2 Step II: Browse Resource Catalogue

FI developers can search through the Resource Catalogue included in the Portal, the services and resources made available by the GEs instances (GIs) and the infrastructure capabilities available and published by the Infrastructure Owners. They can either browse the structure shown or introduce several search criteria to make a first filtering and, according to the type of resource, developers can obtain detailed information like the number of instances for every node and see their description. They cannot publish services, they only can use them.



Generic Enablers

Items per page: 8

Browse By Type :
- Generic Enablers
- Other Services
- Specific Enablers

Browse by Chapter: Services Ecosystems and Deli

Filter by Node: All

Map

Infrastructures

Generic Enablers

Light Semantic Composition Editor - COMPEL
Service Composition Editor based on Semantics
0 vote(s)
Average vote:
more info...

Marketplace - SAP RI
An instrument to facilitate commerce by bringing together vendors and buyers
1 vote(s)
Average vote: 5
more info...
Trento
Number of instances: 1

Mediator
It Provides mediation capabilities and expose them as web services
0 vote(s)
Average vote:
more info...

Registry - SAP RI
The Registry acts as a universal directory of information used for the maintenance, administration, deployment and retrieval of services.
1 vote(s)
Average vote: 3
more info...

Repository - SAP RI
A service and application description repository
1 vote(s)
Average vote: 5
more info...

Service Composition - Ericsson Composition Engine (ECE)
The Ericsson Composition Engine is a toolset for creating and executing composed services and applications.
0 vote(s)
Average vote:
more info...

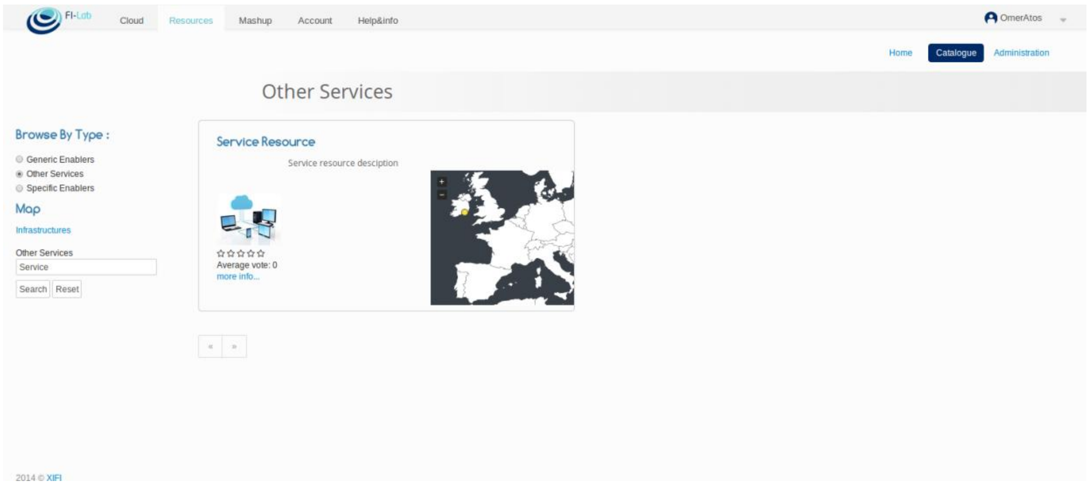
Service Mashup - Mashup Factory
Web-based application for Service Composition and Execution
0 vote(s)
Average vote:
more info...

Store - WStore
The Store GE offers support for selling services to both consumers and developers of Future Internet applications and services and for end-to-end managing of offerings and sales.
1 vote(s)
Average vote: 5
more info...

2014 © XIFI

Figure 7: List of GE and the instances per node.

For the non-conventional services, the FI-Developer can see the description of them; identifying the place and all the necessary information to use it.



Other Services

Browse By Type :
- Generic Enablers
- Other Services
- Specific Enablers

Map

Infrastructures

Other Services: Service

Service Resource
Service resource desciption
Average vote: 0
more info...

2014 © XIFI

Figure 8: List of Other Services (Non-conventional Services).

FI developers can also use the Recommendation Tool to know the opinions provided by other developers or to get a recommendation from the system that better matches their preferences and needs and which is based on functional or technical parameters. The Federation Monitoring tool can be used in this procedure to check the QoS level of deployed services before selecting them.

2.2.3 Step III: Selection and deployment of resources

FI developers have to select in the Resource Catalogue web interface the GE instance (GI) where the service is hosted and the specific node in case several nodes have it deployed from the ones showed. Once the node is selected, the authorization key to use the service can be obtained through the Access Control system. At this stage, the Service Level Agreement (SLA) between the developer and the selected service provider is defined and the accounting and business terms are agreed. Finally, the use of the service is granted and the developer is redirected to its implementation.

FI developers who want to access to a PaaS resource, need to access the Cloud Portal (PaaS Manager) in order to browse and select in the Catalogue the available GEs which can be deployed in the platform, then select the node, the number and type of the virtual machines where the deployment will take place.

FI Developers are able to identify their own services from the federation across XIFI federated nodes. Besides, they are able to identify their registered (SaaS) and deployed (PaaS) services on a facility and see their details.

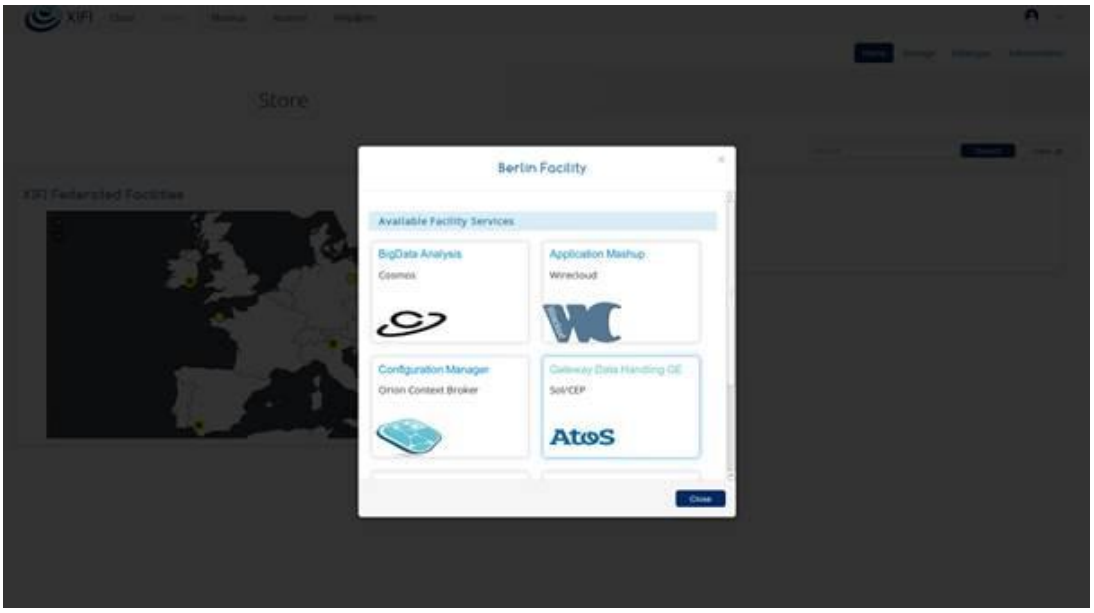


Figure 9: List of GE instances available in Resource Catalogue (mock up)

More details are described in the User manual of the Resource Catalogue and Recommender component [84] or in the deliverable D4.4 BaseLine Tools v2 [81].

2.3 Service monitoring provided by XIFI infrastructures

Two service monitoring procedures are considered in XIFI to be used by Infrastructures owners and FI developers:

- Analysis of metrics;
- Check SLAs.

2.3.1 Analysis of metrics

The Monitoring Management Tool, included in the XIFI Portal, gives access to a graphical interface where the following metrics generated by the Federation Monitoring GE are provided:

- Network and Datacenter Performance;

- GEs Performance.

2.3.2 Check SLAs

FI developers and Infrastructure owners can use the SLAs Dashboard included in the XIFI Portal to monitor the Service Level Agreement (SLA) established between them during the resource provisioning. The QoS metrics included in the SLA contract will be recovered from the Federation Monitoring GE and checked with the negotiated values to detect any violation of the SLA. More details are available at the SLA Manager section included in the deliverable D4.2[12].

2.4 Provision of feedback and comparison of services/resources

FI developers can enhance the use of the XIFI platform, and in particular of the available services/resources, providing feedback to the Resource Catalogue based on their experience through the Recommendation Tool or comparing resources between them. Below the steps to be followed in both cases are described.

2.4.1 Resource recommendation

At first the developer must select the resource to be recommended, which can be a GE/SE provided as a SaaS or a non-conventional service. Developers can provide feedback directly related to a GE/SE type to be stored and used later by other developers, or about the use of a particular instance of a GE/SE. In this case, it must be selected the endpoint associated to the infrastructure where the service is deployed.

In any of those levels, developers can provide reviews through comments or suggestions or evaluate the service in terms of rates. Since non-conventional services cannot be automatically provisioned, for this type of resources only comments or suggestions are allowed.

2.4.2 Comparison of resources

Developers can compare GE/SE types considering a general description of their characteristics or compare particular instances of a same GE/SE.

In order to compare GE/SE types, it is important that they have been described using common features, so that the system can return a table with the comparison. When GE/SE instances are compared, the Service Level Agreement can be compared too.

2.5 How to publish the SE and non-conventional services.

Before the FI Developers can find and use the services, the software providers need to publish them. The following workflow details how to publish the SE and the non-conventional services. Remark that the publication and validation of the Generic Enablers are responsible of the FIWARE Catalogue [84], hence, the XIFI Portal only need to collect this information but not to maintain it. For more details about the publication process refer to [91][91].

Step One: Identification of the provider in the XIFI Portal through the Federation Identity Manager.

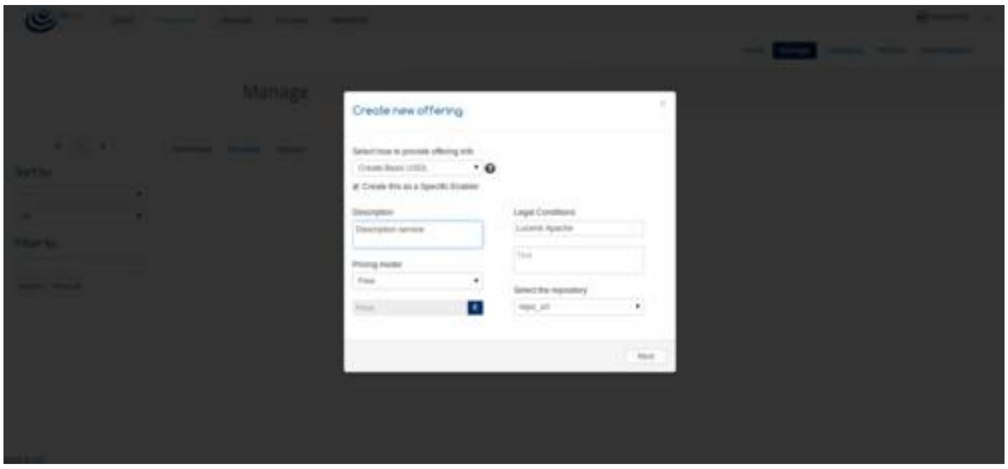
The service provider needs to be identified in the federation. Only, if the user has the role of the Software Provider or the Infrastructure Owner can continue with the process.

Step Two: Register the service.

It is possible to register two types of services: i) Specific Enablers and ii) Advanced/Non-Conventional Services.

- Specific Enabler can be published by Software Providers and Infrastructure Owners. They need to register these SE images in the Cloud Portal, to be deployed automatically for the stakeholders who want to use and monitoring.
- Advanced/non-conventional capabilities (e.g. sensor networks) can be published by the Infrastructure Owners. These don't need to be registered in the Cloud Portal, since it is only necessary to provide their properties and characteristics. This allows obtaining the contact information of the infrastructure owner in order to ask or negotiate with the IO the offer details.

In both cases, register the service description through the USDL


Create new offering

Select how to provide offering info: Create Basic USDL

✓ Create this as a Specific Enabler

Description: Description service

Legal Conditions: License Apache

Pricing model: Free

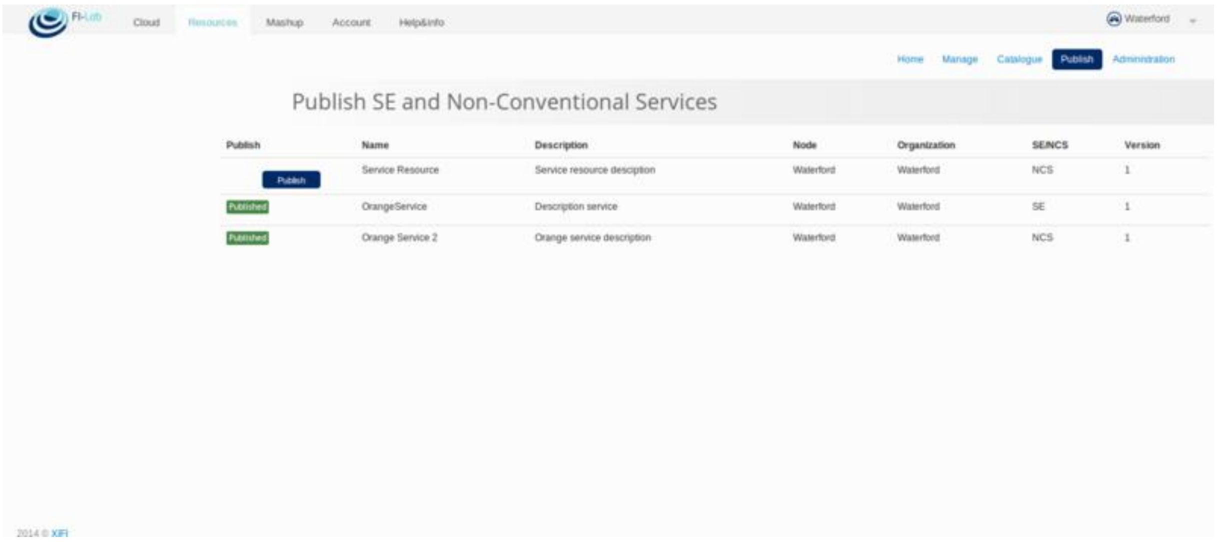
Select the repository: repo_uri

Next

Figure 10: Description of the services that will be included in the Catalogue.

Step Three: Validate the services.

- The federator role needs to validate that all the contents are correct, before to confirm the publication in the catalogue.



Publish SE and Non-Conventional Services

Publish	Name	Description	Node	Organization	SE/NCS	Version
Publish	Service Resource	Service resource desciption	Waterford	Waterford	NCS	1
Published	OrangeService	Description service	Waterford	Waterford	SE	1
Published	Orange Service 2	Orange service description	Waterford	Waterford	NCS	1

2014 © XIFI

Figure 11: List of services pending to be validated.

- After the Federator has approved them, the services and its status will be available through the XIFI Portal.

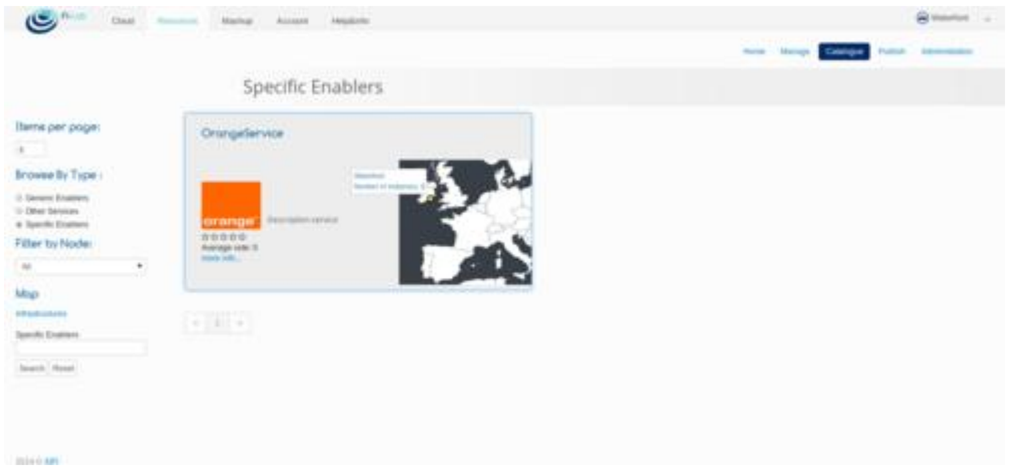


Figure 12: Specific Enabler available in Catalogue

More details are described in the User manual of the XIFI Wiki area Resource Catalogue and Recommender module[84] or in the deliverable D4.4 BaseLine Tools v2 [81].

3 XIFI COMPONENTS – FUNCTIONAL DESCRIPTION

This section is an overview of the components that the users have to install and configure on the new XIFI node (slave and master).

The list of components identified within this Handbook is based on:

- The logical architecture model, as described in the deliverable D1.5[3];
- The definition of sub-system, as defined in deliverable D2.3[6];
- How the XIFI components interact each other, as described in the session “Definition of sub-systems” of deliverable D2.5 [7] and table “Sub-systems under consideration” [7].

Below is reported the definition of each sub-system, as defined in the D2.3[6]:

- Monitoring Sub-system: this sub-system consists of the basic real-time monitoring data collection and of an extension that allows persistence of monitored data;
- Security Sub-system: this sub-system consists of two independent 'flavors', namely the identity management and the security monitoring;
- Deployment and Operations Sub-system: this sub-system consists of the Infrastructure Toolbox and the deployment and configuration adapter;
- User oriented and GUI Sub-system: this sub-system consists of the cloud portal and a number of dashboards accessible through the cloud portal namely info graphics, monitoring, security, SLA management, and the federation management. These heavily depend on other sub-systems to implement meaningful test cases.

The following table provides the list of components, grouped per sub-systems, which can be installed on a new XIFI node (slave and master node):

Sub-system name	Component Name	Described in the deliverable	Pre-requisites	Mandatory
Monitoring Sub-systems	Federation Monitoring (Context Broker GE, Big Data GE)	D2.2[5]	XIMM	Yes – node can work without it but no monitoring information will be available
	XIFI Infrastructure Monitoring Middleware (NAM Adapter, DEM Adapter, Openstack Data Collector, NGSI Adapter)	D3.2[9], D3.5[82]	Monitoring probes, OpenStack	Yes – node can work without it but no monitoring information will be available
Security Sub-system	Federation Identity Management GE	D2.5[7]	Open Stack IdM GE	Yes
	Federation Access Control GE	D2.5[7], FI-WARE D8.1.2b	IdM GE	No
	Security Proxy	XIFI public area wiki[79]	IdM KeyRock	Yes
	Security Monitoring	D4.2[12]	Security Probes	Yes – node can work without it but no security monitoring information will be available

Sub-system name	Component Name	Described in the deliverable	Pre-requisites	Mandatory
User Oriented & GUI Sub-systems	Monitoring Dashboard	N/A. The details will be public in the Wiki area [79]	Cloud Portal , NAM	No
	Security Dashboard	D4.2[12]	IdM GE, Security Monitoring GE, Access Control GE	Yes (master node) - node can work without it but no security monitoring information will be available
	Infographics and Status Pages	D4.3[13]	N/A	N/A
	Cloud Portal	D4.3[13]	Openstack, Security Proxy, IdM Keyrock	Yes
	SLA Manager	D4.2[12]	IdMGE, Federation Monitoring	Yes
	Federation Manager	D2.5[7]	IdMGE	N/A

Sub-system name	Component Name	Described in the deliverable	Pre-requisites	Mandatory
	Interoperability Tool	D4.2[12]	IdM GE	N/A
	Resource Catalogue & Reccomendation tool	D4.2[12]/D4.3[13]	Repository GE, IdM GE and DCA	Yes
Deployment & Operations Sub-systems	Infrastructure Toolbox	D3.3Errore. L'origine riferimento non è stata trovata.	Bare-metal servers and network connectivity	No- users can manually install OpenStack
Infrastructure Components	DCRM (including DOVE) GE	FIWARE documentation[17]	OpenStack	Yes
	Object Storage	FIWARE documentation[17]	OpenStack	Yes

Table 1: List of XIFI component

3.1 Infrastructure Toolbox

ITBox is a deployment tool that helps an Infrastructure Owner to quickly deploy a FIWARE Lab node.

More details about this component are available here [79].

In order to install a XIFI node it's necessary to install many packages and GEs coming from different projects, each one with its own requirements, installation procedures and configuration management. To assist the installation, updating and managing of XIFI nodes, a modular tool has been developed, the Infrastructure Toolbox (ITBox), which provides a ready-to-use server starting from a bare metal server. At the end of installation process, every server will have installed the operating system, the hypervisor and the cloud management software and it will be ready to connect to the federation. The target users are Infrastructure owners and System administrators.

The main functionalities that Infrastructure toolbox provides are:

- Installation of operating system, hypervisor and cloud software through the Preboot eXecution Environment (PXE) on servers.
- Ability to support the deployment process among several pre-defined popular OpenStack configurations (e.g. simple cluster or a high availability cluster).
- Discovering and specifying the server "role": controller; storage; compute; monitoring; etc.
- Ability to test the deployment, so to verify that everything has been correctly installed.

3.2 DCRM GE

The generic enabler (GE) DCRM is a component of the FIWARE Cloud-Hosting Architecture,

more details are available at [70].

The Datacenter Resource Management Generic Enabler or DCRM-GE as detected in the official guide provides the basic Virtual Machine (VM) hosting capabilities, as well as management of the corresponding resources within the Datacenter that hosts a particular FIWARE Cloud Instance.

The DCRM offers all the capabilities that OpenStack natively provides to cloud hosting users and cloud hosting providers, plus some unique extended capabilities [70].

3.3 Object Storage GE

The Object Storage GE [71] is a generic enabler provided by the FIWARE Cloud-Hosting Architecture [72].

Objects can be files, databases or other datasets that need to be archived. Objects are stored in named locations known as “containers”. Containers can be nested, thus objects can be stored hierarchically. The users of the Object Storage Generic Enabler include both FIWARE Cloud Instance Providers and FIWARE Cloud Instance Users [90]. The elements or generic objects, as images and videos, can be stored as metadata and their equipment and life cycle can be managed by this generic enabler [73][58].

3.4 Identity Management GE (federated)

The goal of the Identity Management GE is to identify who has access to the resources such as services, resources and applications of the cloud computing infrastructure. In the context of the XIFI federation the Identity Management will interact with many of these resources in order:

- to allow the user to access the Portal Services;
- to allow the user to access the external (registered) Application;

- to allow the user to access the OpenStack capabilities;
- to allow the user to authenticate on other external and trusted identity managements;
- to provide one single identity for the authentication system.

All those functionalities are achieved by adopting standard protocols, which were identified within the XIFI federation context:

- OAuth that can address almost all the functionalities;
- SAML that provides interoperability between XIFI Identity Management and other Identity Managements that already use this protocol.

The above mentioned functionalities could be divided in two main groups: those related to the federation context and those related to the delegation mechanism.

In the Federation context, this component allows users to login-in through the Identity Management component of a specific domain and then access to the protected resources available into another specific domain.

As for the Delegation, this mechanism involves at least one application server that accesses the resource on behalf of a recognized/authorized user. This application server can be hosted into its domain or into a different domain [53][52].

The new version of this component provides:

- SAML 2.0 support, which allows users' authentication through other Identity Providers, implementing the SAML 2.0 protocol such as OpenAM.
- Implementation through FIWARE KeyStone Proxy of the Security Proxy.
- The Security Proxy component, as described in the D2.5 – APIs and Tools for Infrastructures v2 [7], provides proxy authentication to DCRM GE services toward the IdM GE federation.
- Implementation of HA to have IdM in high availability, both backend and frontend.
- KeyRock IdM GEi architecture based on Galera, to make compatibly the database cluster with it.
- A wrep API implementation for MySQL is provided in order to make database cluster compatibly with the KeyRock IdM GEi (architecture and based on Galera).
- Implementation of the Virtual IP service through the use keepalived of a load balancing and high-availability IP service for Linux.

More details are available in the D2.5 – APIs and Tools for Infrastructure v2 [7].

3.5 XIFI Infrastructure Monitoring Middleware (XIMM)

The XIFI Infrastructure Monitoring Middleware (XIMM) is a concept that has been already tackled in prior XIFI documentation, including the D2.1 - XIFI Handbook v1 [4]. Nonetheless, due to the fact that this abstraction layer has been upgraded, a proper up-to-date review is required. For a more detailed description, please check deliverable D3.5 - Infrastructures monitoring and interoperability adaptation components API v2 [82].

XIMM layer aims at providing standardized monitoring services by means of collecting and handling network and datacenter-based performance data from the infrastructures of the federation. In other words, considering the variety of monitoring solutions deployed by the infrastructures, XIMM is the abstraction layer responsible for collecting, standardizing and publishing monitoring information to the Federation Monitoring. Therefore, each single infrastructure node, independently from its assigned role, shall deploy this framework.

3.5.1 NGSI Adapter

NGSI Adapter is the component in charge of parsing multiple sources of data into a common format. It adapts raw monitoring data into NGSI-ready context information, notifying the Context Broker embedded in the Federation Monitoring. NGSI Adapter itself is completely agnostic to the monitoring tools used to gather data, and the exact format of the data originated by their probes. It delegates the specific adaptation mechanism into an extensible set of dynamically loadable parsers, which receive raw data from the probe (sent to NGSI Adapter as part of a HTTP request) and return the corresponding NGSI context.

3.5.2 Network Active Monitoring - NAM Adapter

Network Active Monitoring (NAM) Adapter is the component in charge of handling cross-domain active measurements between the XIMM instances, providing a standard multi-domain monitoring mechanism able to handle latency and bandwidth-related tests. Historical measurements represent results of regularly scheduled tests and cover one-way delay, jitter, one-way packet loss, and achievable throughput for a path. Nevertheless, NAM Adapter also offers the possibility to request an on-demand measurement of achievable throughput or one-way latency measurement between endpoints.

A single NAM Adapter instance comprises two types of module, Monitoring Probes and Measurement Collectors, pushing the data towards the NGSI Adapter.

- Monitoring Probes: tools used to perform measurement tests between given endpoints. Probes provide the Measurement Collectors with the raw network active monitoring data.
- Measurement Collectors: the core modules of the adapter responsible for collecting the data generated by the probes, processing and forwarding it to the upper layer via a REST-based Web Service API.

3.5.3 Datacenter & Enablers Monitoring - DEM Adapter

The Datacenter and Enablers Monitoring (DEM) Adapter is responsible for collecting and publishing monitored data from physical and virtual servers. Thus, this adapter is of major interest for both infrastructure owners and developers, since it enables to check host resources, such as processor load, RAM utilization, disk usage and more, as well as GEs performance.

The architecture of this component has been significantly modified since the release of previous documentation. The most meaningful update is the specification of two versions of the Adapter:

- DEM Active Adapter is the version which leverages on a Network Management System to handle the monitoring data, in this case Nagios.
- DEM Passive Adapter is embedded in a virtual host to check the performance of a given VM/GE. Hence, this version does not require to be invoked by a monitoring system to perform the measurements.

Further details will be found in the deliverable D3.5 - Infrastructures monitoring and interoperability adaptation components API v2 [82].

3.5.4 Network Passive Monitoring - NPM Adapter

This component is the responsible entity for handling passive monitoring of network devices in the XIMM. Its specification has not been upgraded since the first release of D2.1[4].

This component is the responsible entity for handling the Network Passive Monitoring (NPM) in the XIMM. Passive monitoring of network devices provides information, which is extremely valuable in trouble-shooting as real traffic is monitored. This type of monitoring does not imply an increase of load by injecting artificial traffic. The data is periodically collected through external means, the measurement tools, and saved into data storage. Unlike the NAM Adapter, NPM Adapters work independently from each other, since it is not required an end-to-end communication. This component only collects information about single specific points within the network.

Further details related to this component can be found in the specific deliverable D3.2 [9].

3.6 OpenStack Data Collector – ODC

ODC aims at collecting capacity information from OpenStack installation such as the number of cores installed, size of RAM and disk, number of virtual machines deployed, number of users created.

After the previous release of this handbook, a new user story/requirement has been added to this component in order to add a functionality to collect detailed information about the VM images and the running VMs. Nevertheless the architecture and the manual related to this component remain unchanged from the previous version. A detailed description of this component can be found at [20] or in the deliverable D3.5 [82][82].

3.7 Federation Monitoring

Federation Monitoring aims at providing monitoring information at federation level: data collected by XIMM modules on each node of the federation are elaborated and aggregated by means of some calculations (sum, average on time, average on similar resources - e.g. all the hosts of a given region) and provided through a set of APIs to the consumers (other XIFI components like monitoring dashboard GUI for example) requesting such type of data. Calculations and aggregations are executed in a distributed manner on each node of the federation for the data pertaining that node, so as to have a scalable architecture. Nevertheless raw monitoring data are also directly forwarded from slave nodes to master nodes in order to avoid any possible delay in presenting real time data to the consumers.

For a detailed description of this component and of the installation procedure, see deliverable D2.5 [7].

An overview of the architecture is provided, in order to contextualize this component in the XIFI Federation. The following figure shows the Federation Monitoring component architecture.

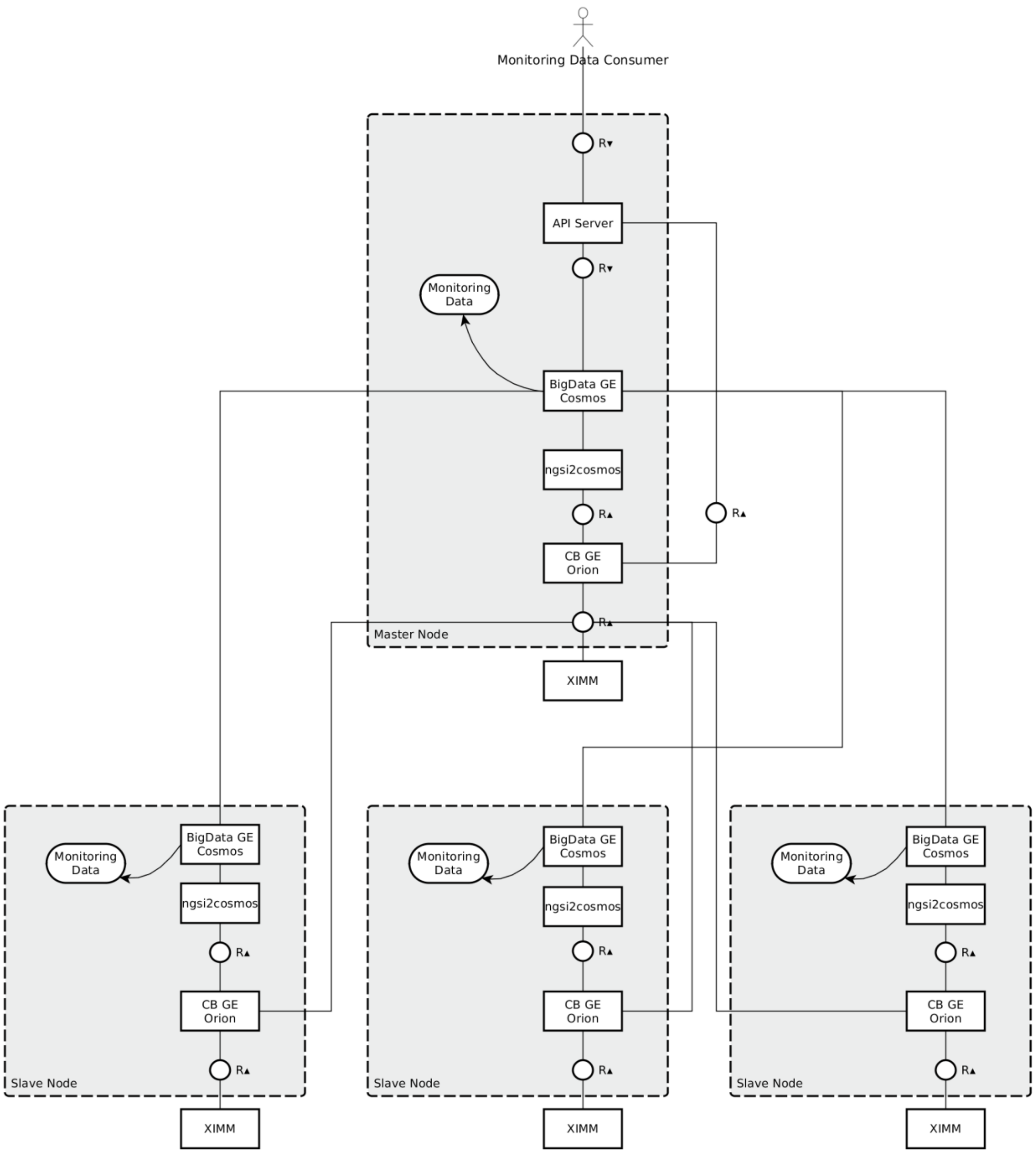


Figure 13: Overview of Federation Monitoring GE

As already mentioned, each node of the federation will gather, elaborate and store the monitoring data collected by XIMM modules on the monitored resources. The Context Broker Orion [60] and the Big Data GEs (COSMOS) [60] provide the functionalities needed on each node, whereas on the master nodes contain an additional component:

- API Server in order to expose APIs for accessing monitoring data.

Raw data collected on slave nodes is also directly forwarded to the master node Context Broker in order to bypass the Big Data GE and provide it to the consumers in real time without any delay.

For an up to date and detailed description of this component refer to the public link [92] or to the deliverable D2.5[7][3].

3.8 Infographics and Status pages

The Infographics and Status Pages component provides graphical information on the infrastructure capacities and status of FIWARE Lab infrastructure services. The service is mainly intended for Developers (that want to know about capacities of FIWARE Lab infrastructure and status of FIWARE Lab infrastructure services) and Federation Manager (that needs to verify the status of FIWARE Lab infrastructure and status of FIWARE Lab infrastructure services).

The Infographics page presents an innovative and intuitive way to publish high-level live information on the available infrastructure capacities in FIWARE Lab. Overall data are displayed in single sections (e.g. Users, Organizations, Regions, CPU, Ram, VM, etc.). Opening each section, it is possible to see more detailed node data, also thanks to the use of maps.

The Status page provides information on the status of the FIWARE Lab infrastructure services (e.g. Nova, Neutron, Cinder, Glance and others for a given node) and offers Jira support both for a specific node and for general portal issues. Even if the Status page is a service normally offered by cloud providers and other services providers, there was not such service available as opensource for OpenStack related infrastructures.

All data are collected from Federation Monitoring API.

A detailed description of this component can be found in the D2.5 – APIs and Tools for Infrastructure v2 [7].

3.9 Resource Catalogue and Recommendation Tool

Resource Catalogue and Recommendation Tool, integrated in the XIFI portal, helps the stakeholders to discover and compare the resources (enablers and advance capabilities) that are available in the XIFI federation environment, it is complementing with the recommendation of the set of resources/services or software components that best match the user preferences. On the other hand, the federation members (infrastructure owners) and the Technological Providers can publish/advertise their resources/services in order to allow the user to find and access the XIFI federated resources/services. The federation members can publish non-conventional services and specific enablers, meanwhile the Technological Providers only can publish specific enablers.

In order to allow these different behaviours, the following roles have been defined for the component in the Federation Identity Manager GE:

i) Developers, they are identified as simple users that want to use the platform as an individual relationship (developers, FI Experimenters, end users, etc.);

ii) Infrastructure Owners, they are federation members that provide their infrastructures and publish their resources/services (Trento, Lannion, Berlin, etc.);

iii) Technological Providers, those who want to publish/share part of their portfolio in the XIFI federation environment (GE/SE developers, SMEs, services providers, etc.);

iv) Federator, it is the responsible to the maintenance of the component and the validation of the catalogue publication.

More details can be found in D4.2 – Baseline tools v1 [12] and in the current description of this component [84].

[3] D2.5 will be published here: http://wiki.fi-XIFI.eu/Main_Page#Technical_Deliverables

3.10 GUI Portal (Marketplace)

The Marketplace Portal provides a single graphical entry point integrated with the Federated Identity Management for accessing to the different services offered by the XIFI federation including: Monitoring Dashboard, SLA and Accounting Dashboard, Resource Catalogue and Recommendation tool, Security and Privacy Dashboard, Interoperability Tools, Infographics and Status Pages and Cloud Portal. More details can be found in D4.3 - XIFI Marketplace Implementation [12]

3.11 SLA Manager

SLA Management, based on SLA management standards and integrated in the XIFI portal, covers the SLA lifecycle and allows the direct interaction among the different actors through the graphical user interface. It supports these actions: i) the providers to indicate the QoS metrics that can be monitored on their infrastructures; ii) the users to view the available metrics for the service and decide the boundaries which should be monitored.

There are two main roles: users (developers, End User, SMEs) and providers (Infrastructure Owners, technological providers). They can use the SLAs Dashboard, included in the SLA Manager component, to monitor the Service Level Agreement (SLA) established between them during the resource provisioning. The QoS metrics included in the SLA agreement will be recovered from the Federation Monitoring GE and checked with the negotiated values to detect any violation of the SLA.

More details are available at the SLA Manager section included in the deliverable D4.2 – Baseline Tools v1 [12] and in the current description of this component [85].

3.12 Security Dashboard

The Security Dashboard, integrated in the XIFI Portal, aims at providing a graphical user front-end to help in the analysis and assessment of security and accountability information collected throughout the federation by the Security Monitoring GE.

Two roles are defined for the Security Dashboard application into the XIFI Identity Manager GE: administrator (for Federation Managers) and IO (for Infrastructure Owners). Based on the assigned user role and thanks to the integration of this component with the Identity Management GE, it is controlled that the security and accountability data is only accessible by authorized users. This means that each infrastructure owner will have permissions only to visualize information generated in the resources of its own node but not in the rest of federated nodes. On the other hand, the Federation Managers will have access to visualize all the alarms generated and events obtained from the different nodes in the federation as well as defining the policies and security directives to be considered.

Finally, the Security Dashboard also enables to get security and statistical reports and help to take the required mitigation actions when a risk or threat is detected in the federation.

More details can be found in the deliverable D4.4 - Baseline Tools v2 [81].

3.13 Security Monitoring GE

The Security Monitoring GE is one of the main Generic Enablers provided by the FIWARE Security Architecture which, in the context of the XIFI Federation, allows not only to Federation Managers but also to Infrastructure Owners, to identify and asses attacks and security risks that can impact on their federated resources. Besides, in conjunction with the XIFI Security Dashboard and due to its integration with the Identify Management GE, it offers an efficient and user-oriented monitoring from the security perspective.

In particular, the component of the Security Monitoring GE used in the XIFI Federation is the Service Level SIEM (Security Information and Event Management), working in a distributed way where Security Probes with SIEM agents are installed on the slave nodes for the collection of logs and normalization into security events, whereas the SIEM server is installed on the master nodes for the analysis and correlation of those security events received from the different nodes to ensure the compliance with the security directives established in the XIFI Federation.

More details can be found in D4.2 – Baseline tools v1[12].

3.14 Federation Access Control GE

The Access Control GE provides XACML-standard-compliant authorization services for each XIFI node. The role and location of the GE in the global XIFI architecture is described in deliverable D1.5 – Federated Platform Architecture Version 2[3]. This component is useful when dynamic attribute-based access control matters, i.e. when authorization depends on attributes at the time of the access request, such as the requester (subject)'s attributes, the requested action, the requested resource, and possibly environment attributes. The component may be used in various ways to control access to critical infrastructure resources in the nodes, but also applications deployed by developers from use case projects. It provides APIs to XIFI stakeholders for managing ABAC and/or RBAC policies, and getting authorization decisions for these policies. It may be combined with the Security Proxy playing the role of PEP (Policy Enforcement Point), in which case the Access Control GE plays the role of PDP (Policy Decision Point). Last but not least, Access Control logs also benefit to the Security Monitoring with help of Security Probes to monitor access events in the XIFI federation. More details can be found in XIFI deliverables D1.5 – Federated Platform Architecture Version 2 [3] and D2.5 – APIs and Tools for Infrastructure v2[7].

3.15 Federation Manager

The Federation Manager governs the “Join-the-Federation” process. As central registration point for new infrastructures and their services this component provides a set of APIs and GUI for both infrastructure owners and administrators of the XIFI federation. More details can be found in the deliverable D2.5 [7].

3.16 Cloud Portal

The Cloud Portal provides a support for the users of the cloud infrastructure and platform to manage their services and resources deployed in cloud. It is implemented as a Web GUI; following the example of the portals that the most common cloud infrastructure managers (like Amazon EC2, Eucalyptus, Cloud Sigma, Rackspace, etc.) have today. In concrete it bases its design principles on the OpenStack Horizon Dashboard. The basic objective of the user portal is to facilitate the user of the cloud to perform operations over the underlying infrastructure. This includes performing actions such as: create user; manage projects; lunch instances on a base of images; create images in the image repository; retrieve flavours from the resource; etc. Moreover the portal facilitates management of a Virtual Data centers (VDCs), Services and Service Data Centers (SDCs), PaaS management and will offer monitoring statistics of the physical and virtual machines.

More details can be found in the deliverable D4.4 – Baseline Tools v2 [81].

3.17 Security Proxy

Security Proxy is the component responsible for authenticating requests that are sent from/to Cloud services. It generates special tokens for representing users acting on behalf of organizations, and these tokens are later used manage different services in the Cloud infrastructure. Federated Security Proxy aims at providing federated access to Cloud resources for members of XIFI federation. It is distributed on all the nodes of the federation and stores authentication tokens. The Federated Keystone Proxy extends the FIWARE Keystone Proxy component to support XIFI requirements. More information can be found in D2.2- APIs and Tools for Infrastructure Federation v1 [5].

The figure below shows how this architecture will be implemented for different nodes in XIFI:

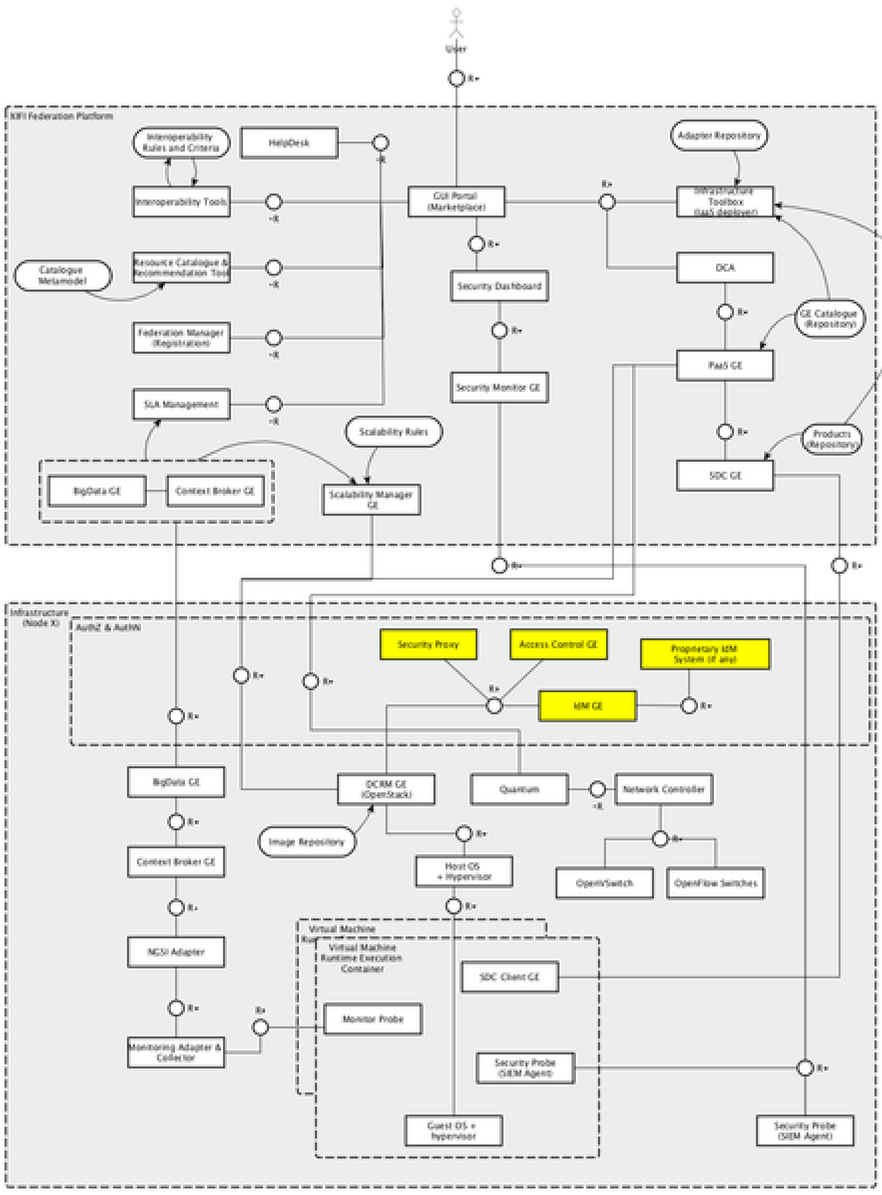


Figure 14: Security Proxy Architecture

3.18 Monitoring Dashboard

The Monitoring Dashboard is a mash up of different monitoring tools for some of the XIFI components currently deployed. Nowadays it consists of two components: VMs monitoring tool and NAM monitoring tool.

VMs monitoring tool (integrated in the Cloud Portal) is in charge of the Virtual Machines monitoring. It gets data from Federating Monitoring API in order to show the user the CPU, Memory and Disk status of its deployed Virtual Machines.

NAM monitoring tool is oriented to XIFI IO and provides them information about the Network status of the connections between XIFI nodes. They can obtain monitoring data about bandwidth, latency and packet loss in two ways: real data information and historical data information.

More details can be found in the deliverable D4.4 - Baseline Tools v2 [81].

3.19 Deployment and Configuration Adapter

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:

- 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.
- Check of Available Resources prior to the Deployment. The DCA performs a check of 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.
- Persistency 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).

More details can be found in the deliverable D3.1 [8].

3.20 Interoperability tools

The Interoperability Tool is a software engineering tool that supports development and testing of FIWARE based applications and services. In particular, the tool focuses on interoperability problems. The main users (i.e. most frequent) of the tool are Future Internet Developers [86]. The tool aids these stakeholders in two different ways:

(1) The developers can test whether their application interoperates with one or more GE instances deployed across XIFI (that is compliant to the open specifications).

(2) During the design and the implementation of new applications and services, developers can use the tool to observe and learn common usages of GEs; therefore, they can quickly build their applications based upon these established patterns of behaviour.

The later version of the tool contains additional features to directly support a further XIFI stakeholder: Infrastructure Owner [86]. Here, interoperability conformance of deployed GEs can be tested. The tool provides suites of executable patterns that can determine the extent to which a single GE can interoperate with applications and other GEs.

More details can be found in D4.4 - Baseline Tools v1 [81].

3.21 PaaS GE

The PaaS Manager helps FI developers to automatize deployment of GEs available at the Catalogue across the XIFI federated infrastructure. Developers access the Cloud Portal and interact with PaaS Manager by means of some forms to define blueprint templates describing the location (XIFI node) and deployment details of the different GEs comprising their applications.

Cloud Portal serves as GUI for PaaS Manager, which is accessed through a RESTful interface. Once blueprint templates are defined, FI developers use them to start the deployment: instances are created at the specified nodes (PaaS Manager interacts with DCRM) and GEs are installed and configured (PaaS Manager relies on SDC GE for that, see below).

More detail about the functionality and how PaaS Manager interacts with DCRM and SDC can be found in the deliverable D2.5 – APIs and Tools for Infrastructure v2[7].

3.22 SDC GE

The SDC (Software Deployment and Configuration) is a GE used to support PaaS Manager in the automated deployment (installation and configuration) of software on the instances launched at XIFI nodes as defined by the blueprint templates created by FI developers. SDC is responsible for applying “recipes” (Chef and/or Puppet) to configure such software, as requested by PaaS Manager through SDC Client Interface.

More details can be found in the deliverable D2.5[7].

4 HOW TO DEPLOY A NEW XIFI NODE

4.1 Overview

This section describes the installation and deployment processes for a generic XIFI node. It is divided into two parts: one for the master node and another for the slave node (the installation procedure for the basic components of a slave node was provided in the previous release of this manual D2.1[4]). More details about the role and the definition of master and slave nodes are documented in the deliverable D5.2[15]. Based on the architectural logical model provided in D1.1.1 – XIFI Core Concepts, Requirement and Architecture draft [1], a list of components is identified for slave and master nodes. For all the components is also provided information on how to validate the installation.

The minimal hardware requirements for basic and high availability physical deployment are provided in the deliverable D5.4 – XIFI federation extension and support - paragraph Basic Physical Deployment [16](figure" Basic Physical Deployment " and table 3" Hardware recommendations for high availability physical deployments").

The technical requirements (software, hardware and connectivity network) for a new XIFI node are provided in the deliverable D5.4[16], in the section "Constraints and Recommendations for the new nodes- Technical Requirements".

In the D2.3 – Test and Validation Report v1 – chapter 2 Test purposes- paragraph Testable sub-system introduced and identified the concept of testable sub-system. A sub system is defined as set of components grouped on basis their functionalities-The list of these sub-systems can be found at the following link [78].

For virtualization, XIFI project uses the KVM hyper-visor. More details about the procedures and protocols for XIFI federation are provided in deliverable D5.1[14]. For the documentation and the packages version of the OS Ubuntu LTS 12.04 refer to the official site http://www.ubuntu.com/. For the documentation and the packages version of KVM, refer to the official site at http://www.linux-kvm.org/page/Main_Page.

Although it is not in the scope of this document to provide information about the test of the installation and configuration process for the deployment of a new XIFI node, a smoke test is provided at the end of each component section, to verify the correctness of the deployment processes for each component More details about the deployment testing are provided in D2.3 [6] or D2.6 – Test and Validation Report v2 [102].

As described in the architectural model (Figure 1: Logical Architecture for XIFI components), the “Identity Management” and the “KeyStone proxy” should be installed on all XIFI slave nodes but, actually, they are installed only on the master node, due to legal issues related to the user account management.

Before starting the installation of a new XIFI node, read the documentation listed in the Appendix A of this document.

4.2 Installing XIFI node (Slave Node)

The list of components (grouped on the basis of their functionalities) needed for the slave node is based on the architectural model provided in D1.5[3]:

- Cloud Computing:
  - DCRM GE
  - SDC Client GE
- Monitoring:
  - Federation Monitoring

    - XIMM component (NGSI Adapter, DEM Adapter, NAM Adapter, NPM Adapter, OpenStackDataCollector)
- Security:
    - Federation Access Control GE
    - Proprietary IdM GE
    - Security Probe

Steps to be followed for the installation of a slave node

- Installation of OpenStack (eventually including the Object Storage GE component to offer this service);
- Installation of the DRCM GE component;
- Installation of the IdM GE and registration of the OpenStack services on IdM GE component (as mentioned before, the "Identity Management" is actually installed only on the master node)
- Installation of the XIMM component (NGSI adapter, NAM adapter, DEM adapter, NPM adapter);
- Installation of Openstack Data Collector;
- Installation of the Federation Monitoring component;
- Installation of Federation Access Control GE;
- Installation of Security Proxy (as mentioned before, the "KeyStone proxy" is actually installed only on the master node);
- Installation and Configuration of Security Probe.

4.2.1 Installing OpenStack and Cloud Service

As described in D5.1 [14], an OpenStack installation requires the definition of different roles for the servers. These roles correspond to a set of services to be installed on each server.

In a flat configuration there are only three main roles:

- *controller*, which includes nova, cinder, glance, neutron, keystone and IdM services;
- *compute*, which includes virtualization and networking capacities (nova, cinder and neutron services);
- *object storage*, which includes the swift component;

In the figure (taken from deliverable D5.1 – Procedures and Protocols for XIFI Federation – paragraph 3.3.2.1 Basic physical deployment - Figure 4

Basic physical deployment page 31 [14]) below are represented the three main roles listed above:

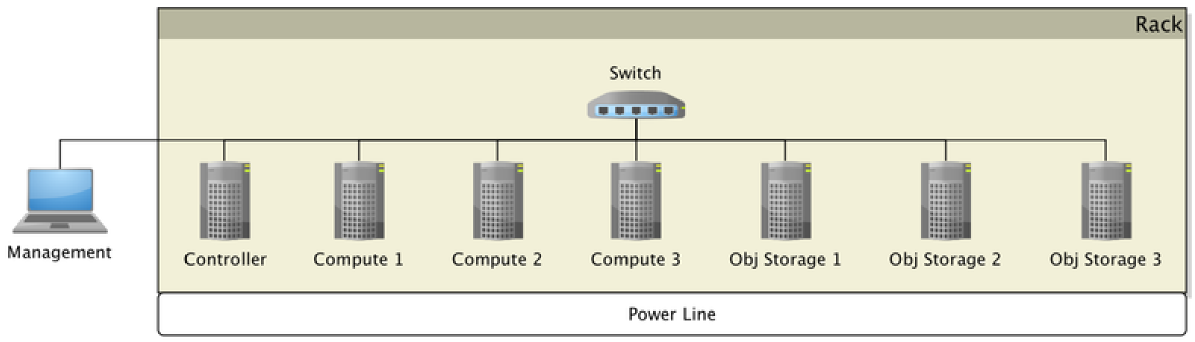


Figure 15: Basic physical deployment

As mentioned in the previous paragraph, the Infrastructure Toolbox provides support for automated installation of OpenStack (excluding Object Storage in v1.1). It is recommended using the Infrastructure Toolbox to install OpenStack. In some cases, there are needs or constraints that require the manual installation of OpenStack (e.g.: some drivers required for a specific server type, are included in the bootable image available in the Infrastructure Toolbox).

4.2.2 Manual Installation

OpenStack can be installed in many different ways, relying on many different products such as OpenStack, Red Hat Cloud Infrastructure, and Fuel Mirantis. In the XIFI federation context and in particular for the installation of XIFI node (slave), it will be used the official installation guide of OpenStack[4].

[4] OpenStack is an opensource cloud management platform designed to be scalable. It is divided in different components integrated each other and through APIs offered and consumed by each service.

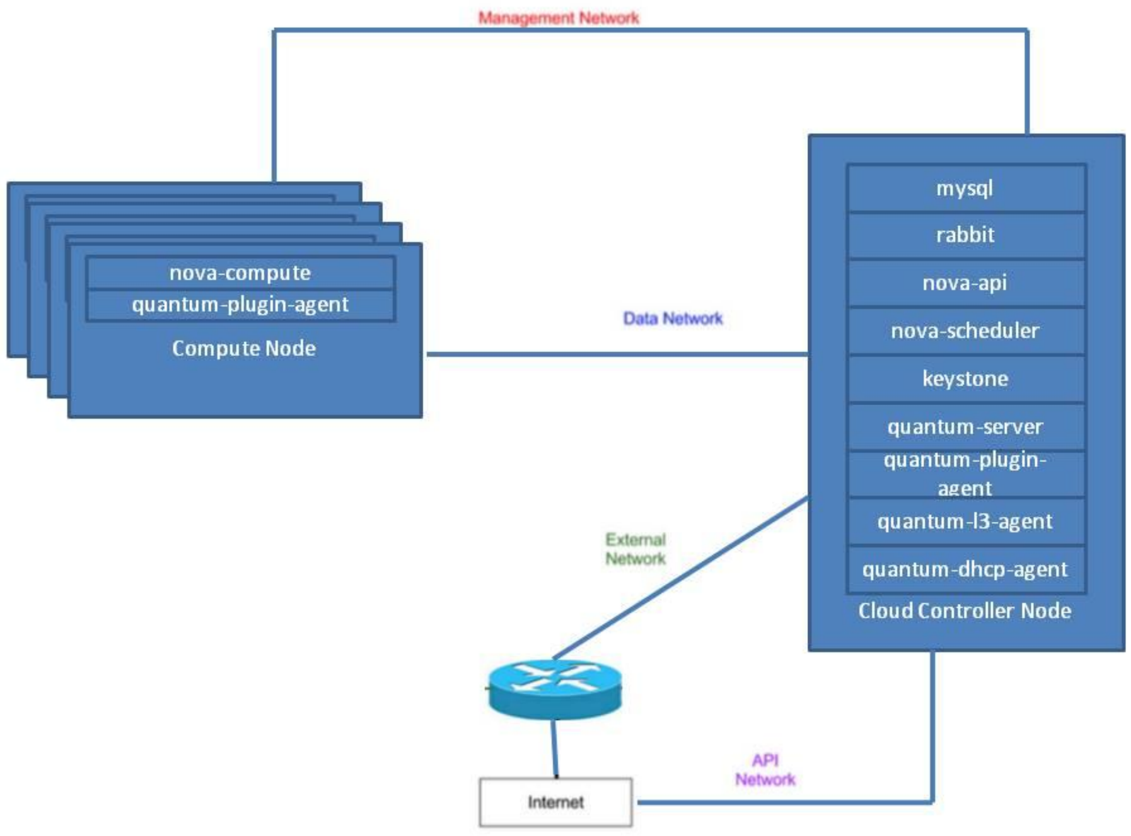


Figure 16: OpenStack Architecture

The official installation guide of OpenStack [21] considers four basic nodes for its architecture:

- Controller node [43][43]: it manages the OpenStack environment (adds new nodes, configures networking, and so on).

  The services, APIs and database hosted by this node are, for example, MySQL (for database), RabbitMQ (for message queue service), Keystone (for authentication and authorization services), Glance (for image management service), Horizon (for user dashboard), and API endpoints and furthermore it schedules compute[21].

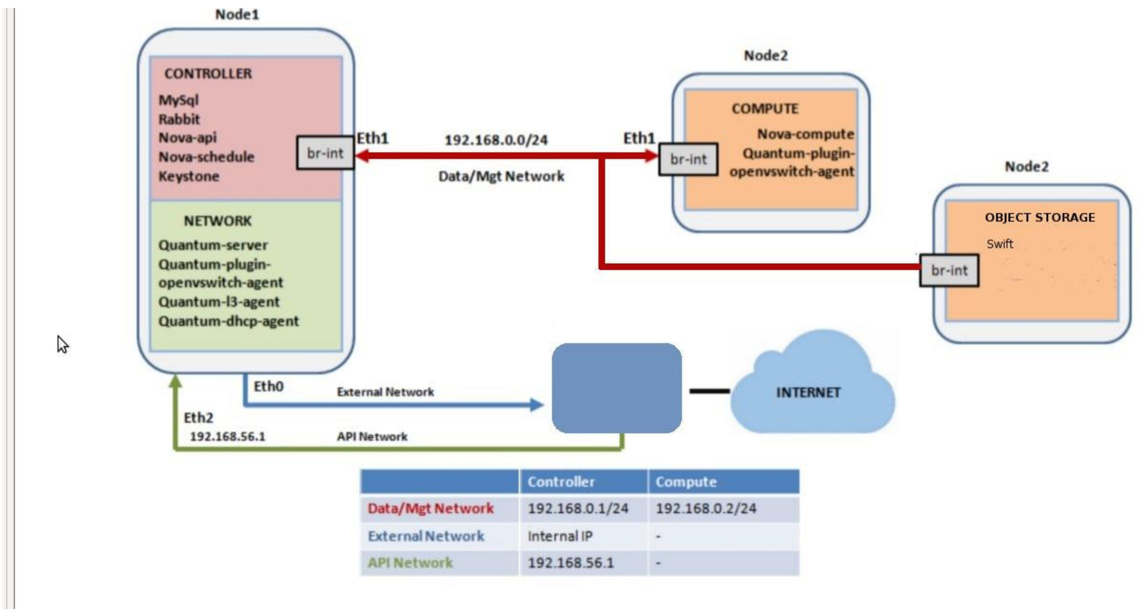
- Compute node [45]: it is a server with a hypervisor installed (KVM [19]) where virtual machines can run. These nodes are the servers on which users will create their virtual machines and host their applications.

- Object Storage node [21]: it is ideal for cost effective, scale-out storage. It provides a fully distributed, API-accessible storage platform that can be integrated directly into applications or used for backup, archiving and data retention. Block storage allows block devices to be exposed and connected to compute instances for expanded storage. Optionally it can be deployed a separate object storage node, this is recommended in high availability environments.

- Network node [44]: it provides an API-driven system to suit the needs of network management. It creates and manages network topologies, manages IP addresses (e.g. static IPs or DHCP) and configures VLANs [44] .

Note: In XIFI the Networking functionalities are included in the controller node, so there is no dedicated network node.

Figure below exemplifies the network deployment of a XIFI node.



	Controller	Compute
Data/Mgt Network	192.168.0.1/24	192.168.0.2/24
External Network	Internal IP	-
API Network	192.168.56.1	-

Figure 17: Network deployment for a new XIFI node

There should be four different physical networks (more details are available in the "OpenStack Operations Guide"[23]):

- Management network is used for internal communication between OpenStack components. The IP addresses on this network should be reachable only within the Datacenter.
- Data Network (or VMs Network) is used for VM data communication within the cloud deployment.
- External Network is used to provide VMs with Internet access in some deployment scenarios. It allows a machine to reach the outside world and to be reached over Internet.
- API network provides all Open Stack’s APIs to tenants. This may be the same network as the external network. It should be a dedicated network used for API communications only, accessible by OpenStack physical machines.

The network configuration for the different networks is defined as follow:

- eth0: Management Network
- eth1: External/API Network
- eth2: Data Network

To install an OpenStack environment the following order should be followed:

1. Install the controller node;
2. Install the compute nodes (including cinder services);
3. Register the compute nodes in the controller;
4. Install the object storage nodes(optional);
5. Register the object storage nodes in the controller (optional).

If an ObjectStorage node is required additional services can be installed, like Swift, on the controller or dedicated node.

4.2.2.1 Controller node Installation and Configuration Steps

The configuration of controller node was realized from the perspective of the XIFI infrastructure node owner and based on the official guide "Basic installation of a controller node" [42], summarized in the following steps:

1. Operating System
2. MySQL Service
3. RabbitMQ Messaging Service
4. Keystone
5. Glance
6. Nova Scheduler
7. Quantum
8. Horizon Dashboard
9. Cinder Scheduler
10. Open vSwitch
11. Swift-proxy service installation
12. Compute node installation
13. Object storage installation (optional)

Note: each command will be launched by the root user (privileged).

4.2.2.1.1 Operating System

First of all, set the controller's operating systems:

Time Zone*: UTC*

Hostname: *controller*

Packages: *OpenSSH-Server*, *NTP (Network Time Protocol)*

And update node's operating system.

Note: each command will be launched by the root user (privileged).

Install the following package:

```
#apt-get install ubuntu-cloud-keyring
```

Edit the Cloud Archive configuring file */etc/apt/sources.list.d/cloud-archive.list*:

```
deb          http://ubuntu-cloud.archive.canonical.com/ubuntuprecise-
updates/grizzly main
```

Upgrade the system (if necessary, reboot the system):

```
#apt-get update; apt-get upgrade
```

Configure the network interface, e.g.:

ip address : *10.20.0.2*

netmask *: 255.255.255.0*

default gateway *: 10.20.0.1*

Edit */etc/sysctl.conf:*

```
net.ipv4.ip_forward = 1
net.ipv4.conf.all.forwarding = 1
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
```

Restart the network service:

```
#service networking restart
```

Apply the sysctl settings:

```
#sysctl -e -p /etc/sysctl.conf
```

Install dnsmasq[refer to table of description]:

```
#apt-get install dnsmasq
```

Configure NTP (Network Time Protocol):

```
#apt-get install ntp
```

Open the ntp configuration file:

```
#nano /etc/ntp.conf
```

Add the following lines:

```
server ntp.ubuntu.com iburst
server 127.127.1.0
fudge 127.127.1.0 stratum 10
```

Restart the service:

```
#service ntp restart
```

4.2.2.1.2 Install and Configure MySQL

At this point the db-server can be installed, the choice is for MySQL.

Note: each command will be launched by the root user (privileged).

Install MySQL packages with the root privileges:

```
#apt-get install mysql-server python-mysqldb
```

Set the network configuration in your file configuration:

```
#sed -i 's/127.0.0.1/0.0.0.0/g' /etc/mysql/my.cnf
```

Restart the service:

```
#service mysql restart
```

Create Databases, Users and Grants. The name of dbs used within this example are: nova, cinder, keystone, quantum. Make sure to have the root privileges for these operations

```
mysql –u root –p <<EOF
CREATE DATABASE nova;
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY
changeit;
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'192.168.0.1' IDENTIFIED BY
changeit;
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'192.168.0.2' IDENTIFIED BY
changeit;
CREATE DATABASE cinder;
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' IDENTIFIED
BY changeit;
CREATE DATABASE glance;
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED
BY changeit;
CREATE DATABASE keystone;
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost'
IDENTIFIED BY changeit;
CREATE DATABASE quantum;
GRANT ALL PRIVILEGES ON quantum.* TO 'quantum'@'localhost'
IDENTIFIED BY changeit;
GRANT ALL PRIVILEGES ON quantum.* TO 'quantum'@'192.168.0.2'
IDENTIFIED BY changeit;
FLUSH PRIVILEGES;
EOF
```

4.2.2.1.3 Install RabbitMQ Messaging Service

Now that dbs are created, install the RabbitMQ Messaging Service.

Note: each command will be launched by the root user (privileged).

Download and install the package:

```
#apt-get install rabbitmq-server
```

Change the password from default to admin:

```
#rabbitmqctl change_password admin
```

Restart the Rabbit service:

```
#service rabbitmq-server restart
```

4.2.2.1.4 Install and Configure Keystone

Note: each command will be launched by the root user (privileged)

```
#apt-get install keystone python-keystone python-keystoneclient
```

Configure its database with users, tenants and service data in the keystone configuration file.

Open the configuration file:

```
#nano  /etc/keystone/keystone.conf
```

Insert the following information:

```
[DEFAULT]
admin_token = changeit
bind_host = 0.0.0.0
public_port = 5000
admin_port = 35357
compute_port = 8774
verbose = True
debug = True
log_file = keystone.log
log_dir = /var/log/keystone
log_config = /etc/keystone/logging.conf
[sql]
connection = mysql://keystone:changeit@localhost:3306/keystone
idle_timeout = 200
[identity]
driver = keystone.identity.back.ends.sql.Identity
[catalog]
driver= keystone.catalog.backends.sql.Catalog
```

Restart Keystone:

```
#service keystone restart
```

Create the tables in its database:

```
#keystone-manage db_sync
```

Set the environments variables: from the home directory create a novarc file which includes the environment variables about the tenants, users and services.

```
#nano /home/trilly/novarc
```

Write in novarc file:

```
export OS_TENANT_NAME = demo
export OS_USERNAME = admin
export OS_PASSWORD =changeit
export OS_AUTH_URL = http://localhost:5000/v2.0/
export SERVICE_ENDPOINT=http://localhost:35357/v2.0
export SERVICE_TOKEN = changeit
```

Since here is reported a sample installation script, must rely on information contained in this script. By using the sample-data.sh (refer to Appendix G) file a default demo and admin tenant will be created.

Export these variables in the configuration file:

```
source novarc
echo "source novarc" >>.bashrc
```

Run the sample-data script (copy the script from Appendix G) that populates the keystone database:

```
#./sample-data.sh
```

Change MySQL and KeyStone as:

```
#MySQL definitions
MYSQL_USER = ketstone
MYSQL_DATABASE = keystone
MYSQL_HOST = localhost
MYSQL_PASSWORD = changeit
#KEYSTONE definitions
KEYSTONE_REGION=RegionOne
SERVICE_TOKEN=changeit
SERVICE_ENDPOINT=http://localhost:35357/v2.0
```

Note: During the execution of bash script "sample-data.sh", get an error, if so, reset the Keystone db by doing as following (must have root privileges):

```
#mysql -u root -p -e "drop database keystone"

#mysql -u root -p -e "create database keystone"

#mysql -u root -p -e "grant all privileges on keystone.* TO 'keystone'@'localhost' identified by 'changeit'

#keystone-manage db_sync
```

Or run the script reset-keystone.sh (copy from Appendix G) and then run again the bash script sample-data.sh.

The identity configuration file is in /etc/keystone/keystone.conf. For details about the content of this file please see the table in the Appendix E of this document. To start the Identity Service use the –config-file parameter to specify a configuration file. If no configuration file is specified, the Identity Service looks for the keystone.conf configuration file in the following directories (in this order):

/.keystone

/

/etc/keystone

/etc

The file ini of keys, /etc/keystone/keystone-paste.ini, provides the Identity Service WSGI middleware pipeline.

NOTE: It's more secure to use a user-name and password to authenticate rather than the service token. When the token is used, the '''keystone''' client could show the following warning on the monitor:

*"**WARNING**: Bypassing authentication using a token & endpoint (authentication credentials are being ignored)".*

4.2.2.1.5 Install and Configure Glance

For the dependencies of the CirrOS 0.3.0 image see the dedicated section of this document[69].

Note: each command will be launched by the root user (privileged).

Install the package:

```
#apt-get  install  glance-api  glance-registry  python-glanceclient
glance-common
```

Open the configuration file of the registry:

```
#nano /etc/glanc e-registry.conf
```

Modify the following lines:

```
sql_connection=mysql://glance:changeit@localhost/glance
[keystone_authtoken]
auth_host=localhost
auth_port=35357
auth_protocol=http
admin_tenant_name=service
admin_user=glance
admin_password=changeit
flavor=keystone
```

After this, change the API configuration file. Open the file:

```
#nano /etc/glance/glance-api.conf
```

Modify the following lines:

```
sql_connection=mysql://glance:changeit@localhost/glance
admin_tenant_name=service
admin_user=glance
admin_password=changeit

notifier_strategy=rabbit
rabbit_host=localhost
rabbit_port=5672
rabbit_use_ssl=false
rabbit_userid=guest
rabbit_password=admin
rabbit_virtual_host=/
rabbit_notification_exchange=glance
rabbit_notification_topic=notifications
rabbit_durable_queues=False

[keystone_authtoken]
auth_host=localhost
auth_port=35357
auth_protocol=http
admin_tenant_name=service
```

```
admin_user=glance
admin_password=changeit

flavor=keystone
```

Now save the file and restart the Glance services:

```
#service glance-api restart && sudo service glance-registry restart
```

Create Glance tables into its database (previous created):

```
#glance-manage db_sync
```

Note: should there be an error during the installation process, reset the glance database using the following script:

```
#./reset-glance.sh
```

Download and import CirrOs 0.3.0 image:

```
#glance image-create –name=cirros-0.3.0-i386 –is-public=true
 --container-format=bare  –disk-format=qc0w2  <  cirros-0.3.0-i386-
disk.img
```

Verify that the image has been introduced in the index:

```
#glance image-list
```

For the complete list of the glance functions, type:

```
#glance   --help
```

4.2.2.1.6 Install and Configure Nova Scheduler

At this point, the Nova package can be installed by following the official guide available at [25]).

```
# apt-get install nova nova-scheduler
```

Configure and modify the file /etc/nova/api-paste.ini:

```
admin_tenant_name=service
admin_user=nova
admin_password=password
signing_dir = /tmp/keystone-signing-nova
```

Append in the /etc/nova/api-paste.ini file the following lines:

```
#============================================================
[composite:osapi volume]
use     =     call:nova.api.openstack.urlmap:urlmap_factory     /:
osvolumeversions
/v1: openstack volume api v1
```

```
#============================================================
#============================================================
[composite: openstack volume api vl]
use = call:nova.api.auth:pipeline_factory
noauth = faultwrap sizelimit noauth ratelimit osapi volume app vl
keystone = faultwrap sizelimit authtoken keystonecontext ratelimit
osapi volume app vl
keystone nolimit = faultwrap sizelimit authtoken keystonecontext
osapi volume app vl
#============================================================
[app:osapi volume app vl]
paste.app factory = nova.api.openstack.volume:APlRouter.factory
#============================================================
#============================================================
[pipeline:osvolumeversions]
pipeline = faultwrap osvolumeversionapp
[app:osvolumeversionapp]
paste.app                                               factory
=nova.api.openstack.volume.versions:Versions.factory
#============================================================
```

Modify the Nova configuration file and append the following lines:

```
[DEFAULT]
# MySQL Connection #
sql connection=mysql://nova:changeit@192.168.0.1/nova
# nova-scheduler #
rabbit host=127.0.0.1
rabbit password=admin
scheduler driver=nova.scheduler.filter scheduler. FilterScheduler
scheduler default filters=AllHostsFilter
# nova-api #
cc host=192.168.0.1
auth strategy=keystone
s3 host=192.168.0.1
ec2 host=192.168.0.1
nova url=http://192.168.0.1:8774/v1.1/
ec2 url=http://192.168.0.1:8773/services/Cloud
keystone ec2 url=http://192.168.0.1:5000/v2.0/ec2tokens
api paste config=/etc/nova/api-paste.ini
allow admin api=True
ec2 private dns show ip=True
dmz cidr=169.254.169.254/32
ec2 dmz host=192.168.0.1
metadata host=192.168.0.1
metadata listen=0.0.0.0
enabled apis=ec2,osapi compute,metadata
# Networking #
network api class=nova.network.quantumv2.api.API
quantum url=http://192.168.0.1:9696
quantum auth strategy=keystone
quantum admin tenant name=service
quantum admin username=quantum
quantum admin password=changeit
quantum admin auth url=http://192.168.0.1:35357/v2.0
libvirt   vif   driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDr
iver
linuxnet  interface  driver=nova.network.linux  net.LinuxOVSInterfac
eDriver
firewall driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
# Cinder #
volume api class=nova.volume.cinder.API
# Glance #
glance api servers=192.168.0.1:9292
image service=nova.image.glance.GlanceImageService
# novnc #
novncproxy base url=http://192.168.56.1:6080/vnc auto.html
novncproxy port=6080
novncproxy host=0.0.0.0
# Misc #
logdir=/var/log/nova
state path=/var/lib/nova
lock path=/var/lock/nova
rootwrap config=/etc/nova/rootwrap.conf
verbose=true
my ip=192.168.0.1
```

Change the permission of the nova-manage log file:

```
#chmod 644 /var/log/nova-manage.log
```

Create Nova tables into the database:

```
#nova-manage db sync
```

Restart Nova Services as root:

```
#service nova-api restart
#service nova-cert restart
#service nova-consoleauth restart service nova-scheduler restart
#service nova-novncproxy restart
```

Configure the nova scheduler; more information about the Nova Scheduler is available [25]. For the XIFI project:

```
scheduler  driver=nova.scheduler.filter  scheduler.  FilterScheduler
scheduler default filters=AllHostsFilter
```

4.2.2.1.7 Install and configure Quantum

The steps describe in this section are based on the OpenStack Operations Guide [23][43].

Note: each command will be launched by the root user (privileged).

Install quantum-server and CLI for accessing the API:

```
#apt-get install quantum-server python-quantumclient
```

Open and edit the configuration file of Quantum:

```
 #nano /etc/quantum/quantum.conf file
```

And modify the following lines:

```
core plugin =
quantum.plugins.openvswitch.ovs quantum plugin.OVSQuant
umPluginV2
auth strategy = keystone
fake rabbit = False
rabbit host=127.0.0.1
rabbit port=5672
rabbit password=admin
rabbit userid=guest
alloca overlapping ips = True
control exchange = quantum
[keystone authtoken]
auth host = 127.0.0.1
auth port = 35357
auth protocol = http
admin tenant name = service
admin user = quantum
admin password = changeit
signing dir = /var/lib/quantum/keystone-signing
```

Configure OpenVSwitch plugin editing the following file:

```
#nano /etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini
```

```
[DATABASE]
sql connection =
mysql://quantum:changeit@localhost:3306/quantum
[OVS]
tenant network type = gre
tunnel id ranges = 1:1000
enable tunneling = True
```

Open the /etc/quantum/api-paste.ini file:

```
#nano /etc/quantum/api-paste.ini
```

And append the following lines:

```
admin tenant name = service
admin user = quantum
admin password = changeit
```

Start the services:

```
#service quantum-server restart
```

4.2.2.1.8 Install Horizon Dashboard

The steps described in this section are based on the official guide of installation Openstack dashboard[20].

Note: each command will be launched by the root user (privileged),

Install the following packages:

```
#apt-get  install  apache2  libapache2-mod-wsgi  openstack-dashboard
memcached python-memcache
#apt-get install lessc
#python /usr/share/openstack dashboard/manage.py compress
```

4.2.2.1.9 Install and configure Cinder Scheduler

The steps describe in this section are re-adapted on the basis of the official guide “Installing and configuring Cinder”[30].

Note: each command will be launched by the root user (privileged).

```
#apt-get install -y cinder-api cinder-scheduler cinder-volume
iscsitarget  open-iscsi  iscsitarget-dkms  python-cinderclient  linux-
headers-'uname -r'
```

Configure & start the iSCSI services :

```
#sed  's/false/true/g' /etc/default/iscsitarget
#service iscsitarget start
#service open-iscsi start
```

Edit the configuration file of Cinder:

```
#nano /etc/cinder/cinder.conf
```

And modify the following lines:

```
api paste confg = /etc/cinder/api-paste.ini iscsi helper = tgtadm
volume name template = volume-%s
volume group = cinder-volumes
verbose = True
auth strategy = keystone
state path = /var/lib/cinder
lock path = /var/lock/cinder
volumes dir = /var/lib/cinder/volumes
sql connection =
mysql://cinder:changeit@localhost:3306/cinder
rabbit password = admin
```

Open the following file:

```
#nano /etc/cinder/api-paste.ini
```

And modify the following lines:

```
service protocol = http
service host = 127.0.0.1
service port = 5000
auth host = 127.0.0.1
auth port = 35357
auth protocol = http admin tenant
name = service admin user = cinder
admin password = changeit signing
dir = /var/lib/cinder
```

With this command create a cinder volume files and assign a preferred size. In this example prepare a 2GB file volume for cinder.

Note: the of parameter MUST contain a valid file path:

```
#dd if=/dev/zero of=cinder-volumes bs=1 count=0 seek=2G
```

Mount it:

```
#losetup /dev/loop2 cinder-volumes
```

Initialize it as an lvm 'physical volume', then create the lvm 'volume group':

```
#pvcreate /dev/loop2
#vgcreate cinder-volumes /dev/loop2
```

Restart the services.

```
#service cinder-volume restart
#service cinder-api restart
```

Create, for example a 1 GB test volume:

```
cinder create --display name test 1
```

4.2.2.1.10 Open vSwitch

The steps described in this section are based on the official guide of the network-node[44].

Note: each command will be launched by the root user (privileged).

Install the following packages:

```
#apt-get install quantum-plugin-openvswitch-agent quantum-dhcp-agent
quantum-13-agent
```

Start Open vSwitch:

```
#service openvswitch-switch start
```

Edit etc/sysctl.conf:

```
net.ipv4.ip forward=1
net.ipv4.conf.all.forwarding=1
net.ipv4.conf.all.rp filter=0
net.ipv4.conf.default.rp filter=0
```

Restart the network service (as root):

```
#/etc/init.d/networking restart
```

Apply the sysctl settings:

```
sysctl -e -p /etc/sysctl.conf
```

Add in the file /etc/hosts, the controller and compute hostnames with correct IP:

```
127.0.0.1  localhost
192.168.0.1      controller
192.168.0.2      compute
```

Now, create an internal network bridge on ethl:

```
#ovs-vsctl add-br br-ex
#ovs-vsctl add-port br-ex eth0 ovs-vsctl add-br br-int
```

Restart networking:

```
#/etc/init.d/networking restart
```

Configure the OpenStack Networking services.

Edit /etc/quantum/13_agent.ini file and modify:

```
auth url = http://192.168.0.1:35357/v2.0
admin tenant name = service
admin user = quantum
admin password = changeit
metadata ip = 192.168.0.1
use namespaces = False
```

Edit /etc/quantum/api-paste.ini file and modify:

```
auth host = 192.168.0.1
admin tenant name = service admin user = quantum
admin password = changeit
```

Edit /etc/quantum/quantum.conf, as follows:

```
core plugin =
quantum.plugins.openvswitch.ovs quantum plugin.OVSQuant
umPluginV2
[DEFAULT]
auth strategy = keystone
fake rabbit = False
rabbit password = admin rabbit host = 127.0.0.1 [keystone authtoken]
auth host = 127.0.0.1 admin tenant name = service
admin user = quantum
admin password = changeit
```

Edit /etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini, as follows:

```
[database]
sql connection =
mysql://quantum:changeit@127.0.0.1/quantum
[ovs]
tenant network type = gre
enable tunneling = True
tunnel id ranges = 1:1000
integration bridge = br-int
tunnel bridge = br-tun
local ip = 192.168.0.1
[securitygroup]
firewall driver =
quantum.agent.linux.iptables firewall. OVSHybridlptables
FirewallDriver
```

Note: If OVS plugin fails to create OVS patch port (refer to the Appendix D: the error is showed in /var/log/quantum/openvswitch-agent.log) and install the following packages:

```
#aptitude install -y openvswitch-datapath-source
#aptitude install module-assistant
#module-assistant auto-install openvswitch-datapath
```

Edit /etc/quantum/dhcp_agent.ini, as follows:

```
[DEFAULT]
enable isolated_metadata = True
enable metadata network = True
```

Edit /etc/quantum/metadata_agent.ini, as follows:

```
[DEFAULT]
auth url = http://127.0.0.1:35357/v2.0
auth region = RegionOne
admin tenant name = service
admin user = quantum
admin password = changeit
nova metadata ip = 127.0.0.1
metadata proxy shared secret = changeit
```

Start the services, as root, as follows:

```
#service quantum-plugin-openvswitch-agent start
#service quantum-dhcp-agent restart
#service quantum-metadata-agent restart
#service quantum-13-agent restart
```

4.2.2.1.11 Swift-proxy service installation

Install the swift-proxy service [31]:

```
#apt-get install swift-proxy memcached python-keystoneclient python-
swiftclient python-webob
```

Create self-signed cert for SSL:

```
# cd /etc/swift
# openssl req -new -x509 -nodes -out cert.crt -keyout cert.key
```

Modify memcached to listen on the default interfaces on a local, non-public network. Edit this line in the /etc/memcached.conf file:

```
-l 127.0.0.1
```

Change it to:

```
-l <PROXY_LOCAL_NET_IP>
```

Restart the memcached server:

```
# service memcached restart

# git clone https://github.com/openstack/swift.git
# cd swift
# python setup.py install
# swift-init proxy start
```

Create /etc/swift/proxy-server.conf:

```
[DEFAULT]
bind_port = 8080
user = swift
[pipeline:main]
pipeline = healthcheck cache authtoken keystoneauth proxy-server
[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true
[filter:keystoneauth]
use = egg:swift#keystoneauth
operator_roles = Member,admin,swiftoperator
```

```
[filter:authtoken]
paste.filter_factory                                                    =
keystoneclient.middleware.auth_token:filter_factory
# Delaying the auth decision is required to support token-less
# usage for anonymous referrers ('.r:*').
delay_auth_decision = true
# cache directory for signing certificate
signing_dir = /home/swift/keystone-signing
# auth_* settings refer to the Keystone server
auth_protocol = http
auth_host = controller
auth_port = 35357
# the same admin_token as provided in keystone.conf
admin_token = ADMIN_TOKEN
#  the  service  tenant  and  swift  userid  and  password  created  in
Keystone
admin_tenant_name = service
admin_user = swift
admin_password = SWIFT_PASS
[filter:cache]
use = egg:swift#memcache
[filter:catch_errors]
use = egg:swift#catch_errors
[filter:healthcheck]
use = egg:swift#healthcheck
```

Create the signing_dir and set its permissions accordingly.

```
# mkdir -p /home/swift/keystone-signing
# chown -R swift:swift /home/swift/keystone-signing
```

Create the account, container, and object rings. The builder command creates a builder file with a few parameters. The parameter with the value of 18 represents 2 ^ 18th, the value that the partition is sized to. Set this “partition power” value based on the total amount of storage you expect your entire ring to use. The value 3 represents the number of replicas of each object, with the last value being the number of hours to restrict moving a partition more than once.

```
# cd /etc/swift
# swift-ring-builder account.builder create 18 3 1
# swift-ring-builder container.builder create 18 3 1
# swift-ring-builder object.builder create 18 3 1
```

For each storage device on each node add entries to each ring:

```
#swift-ring-builder account.builder add z<ZONE>-
<STORAGE_LOCAL_NET_IP>:6002[R<STORAGE_REPLICATION_NET_IP>:6005]/<DEV
ICE> 100
```

```
#swift-ring-builder container.builder add z<ZONE>-
<STORAGE_LOCAL_NET_IP_1>:6001[R<STORAGE_REPLICATION_NET_IP>:6004]/<D
EVICE> 100

#swift-ring-builder object.builder add z<ZONE>-
<STORAGE_LOCAL_NET_IP_1>:6000[R<STORAGE_REPLICATION_NET_IP>:6003]/<D
EVICE> 100
```

Verify the ring contents for each ring:

```
# swift-ring-builder account.builder
# swift-ring-builder container.builder
# swift-ring-builder object.builder
```

Rebalance the rings:

```
# swift-ring-builder account.builder rebalance
# swift-ring-builder container.builder rebalance
# swift-ring-builder object.builder rebalance
```

Copy the account.ring.gz, container.ring.gz, and object.ring.gz files to each of the Proxy and Storage nodes in /etc/swift.

Make sure the swift user owns all configuration files:

```
# chown -R swift:swift /etc/swift
```

Start Proxy services:

```
# service proxy-server start
```

For the swift proxy installation refer to the official guide of Openstack [92].

4.2.2.2 Compute Node

The steps described in this section are based on the official guide of the compute node [45].

1. Network interfaces configuration
2. KVM installation
3. Nova configuration
4. Open vSwitch

The authors take for granted that the operating systems are installed on the servers. If not: install the operating system before to continue with the installation.

4.2.2.2.1 Configure network interface service installation

Install the compute node following the steps below.

Edit network interfaces, /etc/network/interfaces:

```
# The loopback network interface auto lo
iface lo inet loopback
# Only for install packets auto eth0
iface eth0 inet dhcp
# Data/Management interface network
auto ethl
iface ethl inet static
address 192.168.0.2
netmask 255.255.255.0
gateway 192.168.0.1
# Only for external access auto eth2
iface eth2 inet static address 192.168.56.2 netmask 255.255.255.0
```

Edit /etc/sysctl.conf:

```
net.ipv4.conf.all.rp filter = 0 net.ipv4.conf.default.rp filter = 0
```

Restart the network service:

```
#/etc/init.d/networking restart
```

Apply the sysctl settings:

```
#sysctl -e -p /etc/sysctl.conf
```

4.2.2.2.2 KVM installation and configuration

Install Hypervisor packages:

```
#apt-get install -y kvm libvirt-bin pm-utils
```

Libvirt (Virtualization API)[5].

Edit /etc/libvirt/qemu.conf file and add:

```
cgroup device acl = [
"/dev/null",      "/dev/full",      "/dev/zero",      "/dev/random",
"/dev/urandom",
"/dev/ptmx",   "/dev/kvm",   "/dev/kqemu",   "/dev/rtc",   "/dev/hpet",
"/dev/net/tun"]
```

Disable KVM default virtual bridge to avoid any confusion:

```
#virsh net-destroy default sudo virsh net-undefine default
```

[5] The libvirt library is a Linux API over the virtualization capabilities of Linux that supports a variety of hypervisors, including Xen, KVM and Qemu.

Where virsh stands for the virtualization interactive terminal and it communicates with libvrt API. Allow Live Migration.

Edit /etc/libvirt/libvirtd.conf file:

```
listen tls = listen tcp = 1 auth tcp = "none"
```

Modify libvirtd opts variable in /etc/init/libvirt-bin.conf file:

```
 env libvirtd opts="-d -l"
```

Edit /etc/default/libvirt-bin file:

```
libvirtd opts="-d -l"
```

Restart libvirt:

```
#service libvirt-bin restart
```

4.2.2.2.3 Nova Compute installation and configuration

At this point the hyper-visor, which will be used for virtualization, can be installed. For the XIFI project, use the KVM hyper-visor.

```
#apt-get instali nova-compute-kvm
```

Configure Nova editing /etc/nova/api-paste.ini file:

```
auth host = 192.168.0.1
admin tenant name = service
admin user = nova
admin password = changeit
signing dir = /tmp/keystone-signing-nova
```

Edit /etc/nova/nova-compute.conf file and add:

```
[DEFAULT]
libvirt type=kvm
compute driver=libvirt.LibvirtDriver
```

Edit /etc/nova/nova.conf file in compute node and modify:

```
[DEFAULT]
dhcpbridge flagfile=/etc/nova/nova.conf
dhcpbridge=/usr/bin/nova-dhcpbridge
logdir=/var/log/nova
state path=/var/lib/nova
lock path=/var/lock/nova
force dhcp release=True
iscsi helper=tgtadm
libvirt use virtio for bridges=True
connection type=libvirt
root helper=sudo nova-rootwrap /etc/nova/rootwrap.conf
verbose=True
```

```
ec2 private dns show ip=True
api        paste        config=/etc/nova/api-paste.ini        volumes
path=/var/lib/nova/volumes
# MySQL Connection #
sql connection=mysql://nova:changeit@192.168.0.1/nova
# nova-scheduler #
rabbit host=192.168.0.1
rabbit password=admin
scheduler driver=nova.scheduler.filter scheduler.Filter
Scheduler
scheduler default filters=AllHostsFilter
# nova-api #
cc host=192.168.0.1
auth strategy=keystone
s3 host=192.168.0.1
ec2 host=192.168.0.1
nova url=http://192.168.0.1:8774/v1.1/
ec2 url=http://192.168.0.1:8773/services/Cloud
keystone ec2 url=http://192.168.0.1:5000/v2.0/ec2tokens
api paste config=/etc/nova/api-paste.ini
allow admin api=True
ec2 private dns show ip=True
dmz cidr=169.254.169.254/32
ec2 dmz host=192.168.0.1
metadata host=192.168.0.1
metadata listen=0.0.0.0
enabled apis=metadata
service quantum metadata proxy=True
quantum metadata proxy shared secret=changeit
# Networking #
network api class=nova.network.quantumv2.api.API
quantum url=http://192.168.0.1:9696
quantum auth strategy=keystone
quantum admin tenant name=service
quantum admin username=quantum
quantum admin password=changeit
quantum admin auth url=http://192.168.0.1:35357/v2.0
libvirt vif driver=nova.virt.libvirt.vif.LibvirtHybrid0
VSBridgeDriver
linuxnet     interface     driver=nova.network.linux     net.Linux0
VSlnterfaceDriver
firewall driver=nova.virt.libvirt.firewall.IptablesFire wallDriver
# Compute #
compute driver=libvirt.LibvirtDriver connection type=libvirt
libvirt type=kvm
# Cinder #
volume api class=nova.volume.cinder.API
# Glance #
glance api servers=192.168.0.1:9292
image service=nova.image.glance.GlanceImageService
# novnc #
novnc enable=True
novncproxy base url=http://192.168.56.1:6080/vnc auto.h
tml
vncserver proxyclient address=192.168.0.2
vncserver listen=0.0.0.0
# Misc #
logdir=/var/log/nova
state path=/var/lib/nova
lock path=/var/lock/nova
rootwrap config=/etc/nova/rootwrap.conf
verbose=True
```

Edit /etc/nova/nova.conf file in controller node and modify:

```
compute driver=libvirt.
```

Restart Nova services:

```
#service nova-compute restart
```

4.2.2.2.4 Open vSwitch

This section describes the installation of the Open vSwitch package [27].

Install the packages:

```
#apt-get install openvswitch-switch
```

Start Open vSwitch service:

```
#service openvswitch-switch start
```

Note: if OVS plugin failed to create OVS [47] patch port (showed in the file log /var/log/quantum/openvswitch-agent.log), the following packages should be installed:

```
#aptitude install -y openvswitch-datapath-source
#aptitude install module-assistant
#module-assistant auto-install openvswitch-datapath
```

Configure Virtual Bridging [61] :

```
#ovs-vsctl add-br br-int
```

Install the packages:

```
#apt-get install -y quantum-plugin-openvswitch-agent
```

Edit /etc/quantum/quantum.conf file and modify the following lines:

```
core plugin =
quantum.plugins.openvswitch.ovs quantum plugin.OVSQuantu
mPluginV2
auth strategy = keystone
allow overlapping ips = True
control exchange = quantum
fake rabbit = False
rabbit host = 192.168.0.1
rabbit password = admin
```

```
#[keystone authtoken]
auth host = 192.168.0.1
auth port = 35357
auth protocol = http
admin tenant name = service
admin user = quantum
admin password = changeit
signing dir = /var/lib/quantum/keystone-signing
```

Edit /etc/quantum/plugins/openvswitch/ovs_quantumplugin.ini file and modify:

```
[DATABASE]
sql connection =
mysql://quantum:changeit@192.168.0.1:3306/quantum
[OVS]
tenant network type = gre
tunnel id ranges = 1:1000
integration bridge = br-int
tunnel bridge = br-tun
local ip = 192.168.0.2
enable tunneling = True
```

Start the Agent:

```
#service quantum-plugin-openvswitch-agent restart
```

Enable the NFS Driver in the /etc/cinder/cinder.conf:

```
volume_driver=cinder.volume.drivers.nfs.NfsDriver
```

Create at least one NFS share.

For example having the following shares:

- 192.168.1.200:/storage
- 192.168.1.201:/storage
- 192.168.1.202:/storage

Insert these shares in the /etc/cinder/shares.txt file:

```
# cat /etc/cinder/shares.txt
192.168.1.200:/storage
192.168.1.201:/storage
192.168.1.202:/storage
```

As last step configure the directory that cinder will use to store the configured NFS shares. Change the value of the nfs_mount_point_base option in the /etc/cinder/cinder.conf file accordingly.
For example having:

```
nfs_mount_point_base = /var/lib/cinder/nfs
```

The images will be stored in that directory:

```
# ls /var/lib/cinder/nfs
...
46c5db75dc3a3a50a10bfd1a456a9f3f
```

Having this directory set, volumes can be easily created as usual:

```
# nova volume-create --display-name=myvol 5
```

4.2.2.3 Install Object Storage

In this section is explained a two-node Object Storage replica.

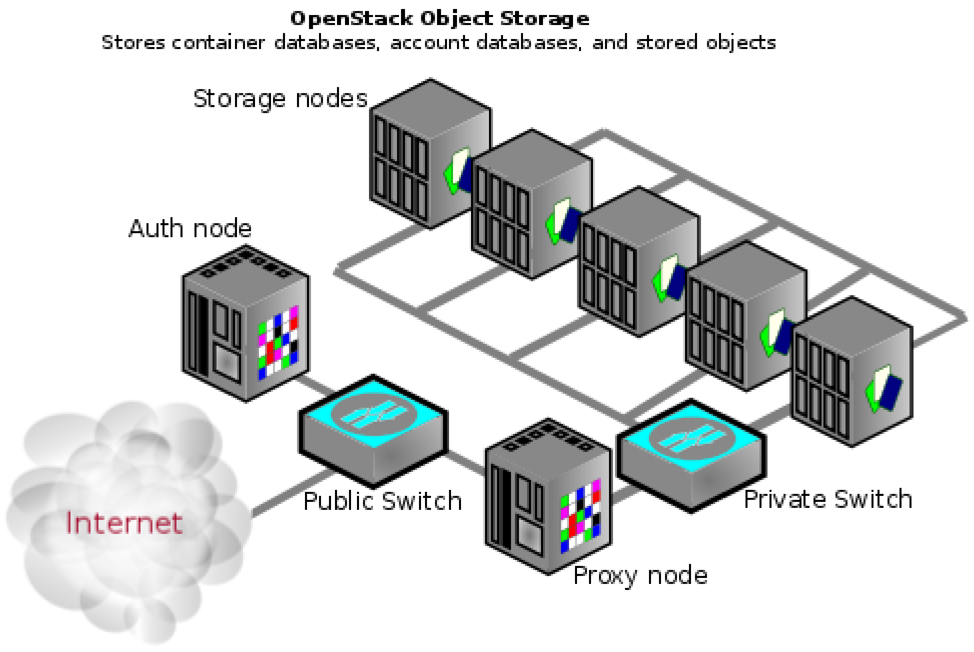


Figure 18: OpenStack Object Storage

In this configuration refer to the figure above[62]), the infrastructure will be composed by two Object Storage servers and one Proxy node, which will be used as API front-end for the Object Storage requests.

The components will be:

- One Proxy node which runs swift-proxy-server service
- Two Object Storage nodes that run the swift-account-server, swift-container-server and swift-object-server services.

The steps identified are:

1. General installation to be performed on both proxy and storage nodes
2. Storage nodes configuration
3. Proxy nodes configuration.

4.2.2.3.1 General installation steps (to be performed on both proxy and storage nodes)

Note: each command will be launched by the root user.

Create the configuration directories and assign them the right permissions:

```
# mkdir -p /etc/swift
# chown -R swift:swift /etc/swift/
```

Create the /etc/swift/swift.conf file and populate it with the following configuration:

```
[swift-hash]
# random unique string that can never change (DO NOT LOSE)
swift_hash_path_suffix = fLIbertYgibbitZ
```

Note: The suffix value in /etc/swift/swift.conf should be set to some random string of text to be used as a salt when hashing to determine mappings in the ring. This file should be the same on every node in the cluster!

4.2.2.3.2 Storage Nodes configuration steps

Install the node core packets:

```
# apt-get install swift-account swift-container swift-object
xfsprogs
```

Since each disk uses anobject storage disk, a single partition should be created on it and formatted as XFS filesystem (for better performances).

For each disk should be issued the following commands, in order to correctly initialize them.

Create the partition:

```
# fdisk /dev/sdb
```

Format the partition as XFS:

```
# mkfs.xfs -i size=1024 /dev/sdb1
```

Add an entry in the /etc/fstab file to automount the partition at boot time:

```
# echo "/dev/sdb1 /srv/node/sdb1 xfs
noatime,nodiratime,nobarrier,logbufs=8 0 0" >> /etc/fstab
```

Create the directory, mount and assign permissions:

```
# mkdir -p /srv/node/sdb1
# mount /srv/node/sdb1
# chown -R swift:swift /srv/node
```

Create /etc/rsyncd.conf file with the following content:

```
uid = swift
gid = swift
```

```
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address = <STORAGE_LOCAL_NET_IP>

[account]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/account.lock

[container]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/container.lock

[object]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/object.lock
```

Edit the following line in /etc/default/rsync:

```
RSYNC_ENABLE = true
```

Restart the rsync daemon:

```
# service rsync restart
```

Create the swift recon cache directory and set its permissions:

```
# mkdir -p /var/swift/recon
# chown -R swift:swift /var/swift/recon
```

4.2.2.3.3 Proxy Node configuration steps

Install the swift-proxy packets:

```
# apt-get install swift-proxy memcached python-keystoneclient
python-swiftclient python-webob
```

Create self-signed cert for SSL:

```
# cd /etc/swift
# openssl req -new -x509 -nodes -out cert.crt -keyout cert.key
```

Edit the following line in /etc/memcached.onf

```
-l <PROXY_LOCAL_NET_IP>
```

Restart the memcached service:

```
# service memcached restart
```

Create /etc/swift/proxy-server.conf with the following content

```
[DEFAULT]
bind_port = 8888
user = swift

[pipeline:main]
pipeline = healthcheck cache authtoken keystoneauth proxy-server

[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true

[filter:keystoneauth]
use = egg:swift#keystoneauth
operator_roles = Member,admin,swiftoperator

[filter:authtoken]
paste.filter_factory                                                     =
keystoneclient.middleware.auth_token:filter_factory

# Delaying the auth decision is required to support token-less
# usage for anonymous referrers ('.r:*').
delay_auth_decision = true

# cache directory for signing certificate
signing_dir = /home/swift/keystone-signing

# auth_* settings refer to the Keystone server
auth_protocol = http
auth_host = 192.168.56.3
auth_port = 35357

# the same admin_token as provided in keystone.conf
admin_token = 012345SECRET99TOKEN012345

#  the  service  tenant  and  swift  userid  and  password  created  in
Keystone
```

```
admin_tenant_name = service
admin_user = swift
admin_password = swift

[filter:cache]
use = egg:swift#memcache

[filter:catch_errors]
use = egg:swift#catch_errors

[filter:healthcheck]
use = egg:swift#healthcheck
```

Create the signing_dir and set its permissions accordingly:

```
# mkdir -p /home/swift/keystone-signing
# chown -R swift:swift /home/swift/keystone-signing
```

Create the account, container and object rings. The builder command is basically creating a builder file with a few parameters. The parameter with the value of 18 represents 2 ^ 18th, the value that the partition will be sized to. Set this “partition power” value based on the total amount of storage you expect your entire ring to use. The value of 3 represents the number of replicas of each object, with the last value being the number of hours to restrict moving a partition more than once.

```
# cd /etc/swift
# swift-ring-builder account.builder create 18 3 1
# swift-ring-builder container.builder create 18 3 1
# swift-ring-builder object.builder create 18 3 1
```

For each storage device on each node add entries to each ring:

```
# swift-ring-builder account.builder add z<ZONE>-
<STORAGE_LOCAL_NET_IP>:6002/<DEVICE> 100
# swift-ring-builder container.builder add z<ZONE>-
<STORAGE_LOCAL_NET_IP_1>:6001/<DEVICE> 100
#swift-ring-builder object.builder add z<ZONE>-
<STORAGE_LOCAL_NET_IP_1>:6000/<DEVICE> 100
```

For example, to set up a storage node with a partition in Zone 1 on IP 10.0.0.1, the mount point of this partition is /srv/node/sdb1, and the path in rsyncd.conf is /srv/node/, the DEVICE would be sdb1 and the commands would look like:

```
# swift-ring-builder account.builder add z1-10.0.0.1:6002/sdb1 100
# swift-ring-builder container.builder add z1-10.0.0.1:6001/sdb1 100
# swift-ring-builder object.builder add z1-10.0.0.1:6000/sdb1 100
```

Verify the ring contents for each ring:

```
# swift-ring-builder account.builder
# swift-ring-builder container.builder
# swift-ring-builder object.builder
```

Rebalance the rings:

```
# swift-ring-builder account.builder rebalance
# swift-ring-builder container.builder rebalance
# swift-ring-builder object.builder rebalance
```

Copy the account.ring.gz, container.ring.gz, and object.ring.gz files to each of the Proxy and Storage nodes in /etc/swift.

Make sure all the config files are owned by the swift user:

```
# chown -R swift:swift /etc/swift
```

Start Proxy services:

```
# service proxy-server start
```

Start the Storage Nodes Services.

Now that the ring files are on each storage node the services can be started. On each storage node run the following:

```
# service swift-object start
# service swift-object-replicator start
# service swift-object-updater start
# service swift-object-auditor start
# service swift-container start
# service swift-container-replicator start
# service swift-container-updater start
# service swift-container-auditor start
# service swift-account start
# service swift-account-replicator start
# service swift-account-updater start
# service swift-account-auditor start
```

4.2.3 Automated Installation for Infrastructure ToolBox

The ITBox is distributed as ISO image (download the installation image from the ftp.fi-XIFI.eu/itbox, which contains an installer for ITBox Master Server).

The ISO can be installed indifferently, 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 UI (the login prompt on the console of the master node will show you the URL you need to use. The default address is **http://10.20.0.2:8000/**), or he can start using the command line interface. The cluster's servers will be booted in bootstrap mode (CentOS based Linux in memory) via PXE and thus these servers will be seen by the system as "discovered" and the user will see notifications in the user interface. At this point the user can create an environment, starting with their configuration. In more details, the steps to perform are:

- On each server, the user selects PXE boot and enables the CPU Virtualization.
- The user boots the servers in boostrap mode. When, the boostrap is installed, the servers are in "discovered" state, and they are usable by ITBox.
- Creation of a new environment (named also project). The user can select installation options: operating system (Ubuntu 12.04), hypervisor (KVM), network plugin (Neutron), Cinder, Glance and DCRM GE.
- The user assigns a role to every server: Controller, Compute, Block storage.
- Verification of cluster network settings. Thus, the user can test if the cluster is well configured.
- After all the previous steps are accomplished, the Infrastructure Owner can finally trigger deployment on servers.
- When the deployment process is finished, the user runs a test in order to verify the correct installation of all cloud environment modules and if the node is ready to join the federation.

The user can install his/her own cloud infrastructure by using several deployment models: multi-node, multi-node with High Availability using two controllers, multi-node with High Availability using at least three controllers.

The High Availability can be obtained using different technologies and strategies. For example, Fuel Mirantis suggests a solution Active/Active implementation with Galera, Corosync and Pacemaker, which requires at least three controller nodes. The user could choose a simplest Active/Passive solution using only Pacemaker and Corosync without Galera.

The following deployment model is a hybrid solution: High Availability Active/Active with Galera, Corosync and Pacemaker, using only two controllers.

Since Galera uses a "quorum based" algorithm, so as to guarantee data consistency, at least three nodes are needed. So, in order to use only two controller nodes, a Galera Arbitrator will be used (garbed daemon), in order to resolve the *"Split-brain problem",* which is present with an even number of nodes.

The suggested solution is yet scalable and it allows saving a controller node.

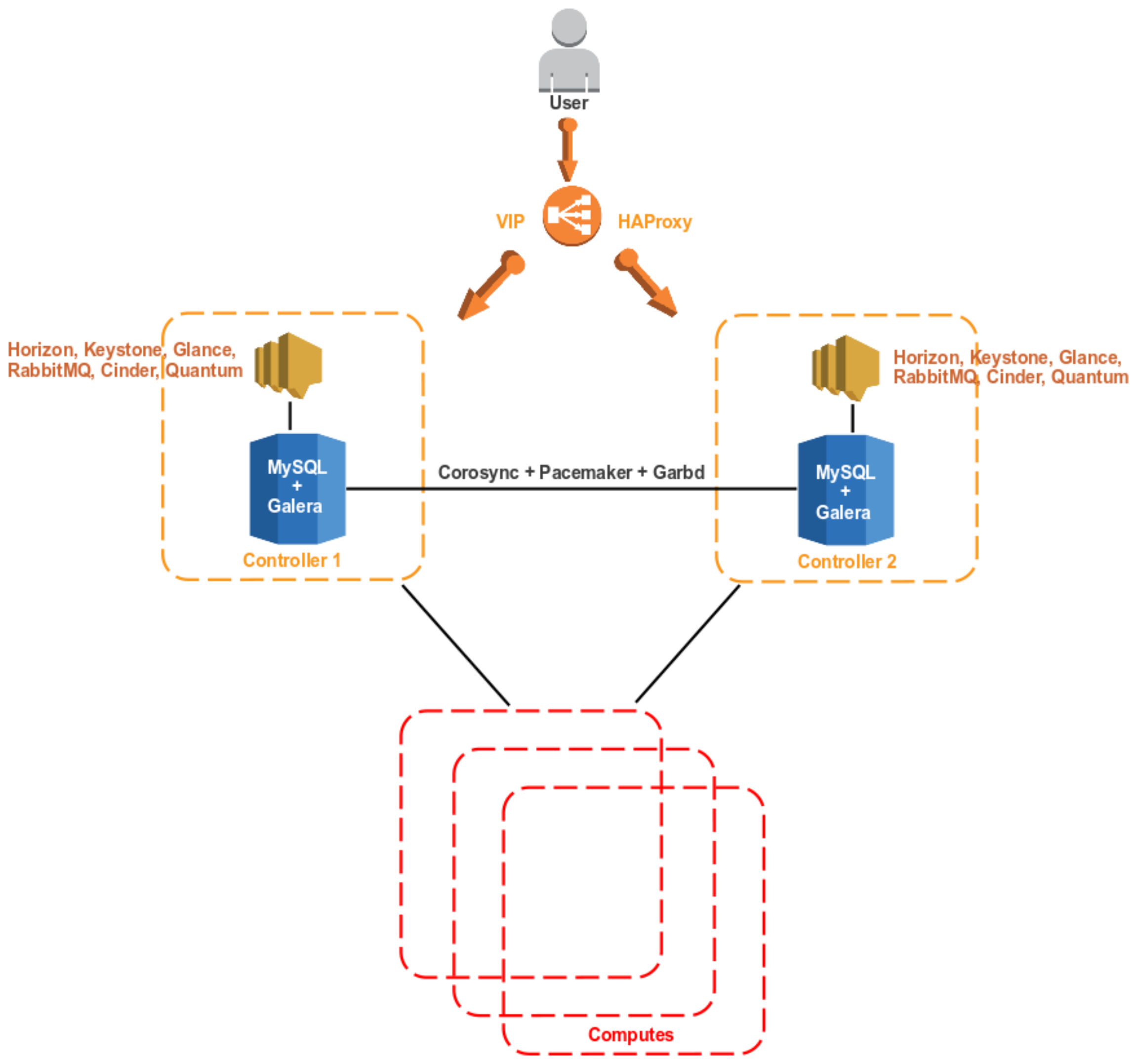

User
VIP
HAProxy
Horizon, Keystone, Glance, RabbitMQ, Cinder, Quantum
MySQL + Galera
Controller 1
Corosync + Pacemaker + Garbd
Horizon, Keystone, Glance, RabbitMQ, Cinder, Quantum
MySQL + Galera
Controller 2
Computes

Figure 19: Infrastructure Toolbox

4.2.3.1 ITBox Network Settings

The ITBox requires one network (called "ITBox network") for PXE booting, and the configuration of three networks for OpenStack: Public, Storage and Management.

The ITBox network requires being untagged, so a specific physical interface is allocated just for ITBox management purpose. The OpenStack networks instead can use VLAN segregation, so in the basic configuration, they can be grouped in one physical interface.

The Public network is used to expose API and offer Internet access to VMs using NAT. Basically it is a range of public IP, reachable by Internet. By default each server obtains a Public IP from the selected IP range: for this reason, the IP range dimension should be allocated as many as the server number (look also at the section "*Manually change of a public IPs*"). In the VLAN tagging field the user can choose a specific ID to associate. If the user's network devices have pre-configured VLAN ID subnet allocated for Internet access, the user should use this pre-configured VLAN.

The section Neutron L3 Configuration is related to the Public network, because typically the Floating IP range and the Public network are in the same subnet.

Essentially, the user has to choose how to allocate two ranges from the subnet used for Internet access: one for the Public Network and one for the Floating IP.

The Management and Storage network are OpenStack internal networks, have IP reachable only within the Datacenter and connect all servers. The Management network is used for internal OpenStack components communication, while the Storage network is dedicated for the storage data traffic.

The ITBox supports 3 types of OpenStack Network environments: *nova-network*, *Neutron with GRE segmentation* and *Neutron with VLAN segmentation*. Nova-network is the legacy mode with basic network configuration. Neutron instead offers more freedom and advanced L2 and L3 features. The ITBox supports Neutron using Open vSwitch plugin. The suggested configuration is to use Neutron with GRE encapsulation: this is the easiest way to install and manage the OpenStack Networking part. Using GRE option, a mesh or tunnel are automatically created between each server upon the OpenStack Management Network, so the user has not to set any additional VLAN or allocate a dedicated network adapter (required using *Neutron with VLAN segmentation* option)

4.2.3.2 Installation and validation check

In order to verify the correct installation of the ITBox, a series of command are executed.

The result for the validation and correctness of the procedures for the installation and deployment of the XIFI nodes are described in the D2.3- Chapter4 "Tools and procedures for the installation and deployement of XIFI node- Paragraph5 Infrastructure Toolbox" [6].

For any issue or question please refer to jira.fi-ware.org.

4.2.3.3 DNS configuration

The ITBox gives the possibility to set the DNS for the Internal Network in the section L3 Configuration. By default, Google Public DNS are used, with the IP addresses 8.8.8.8 and 8.8.4.4.

4.2.4 Quota Configuration

A System Administrator might change the default quotas. For example the disk space associated to user (or project) or a number of volumes per tenant.

The python-novaclient package provides the nova quota-* commands, to manage tenant quotas [63][63] . The default values are:

- quota_instances: 3 (number of instances allowed per tenant)
- quota_floating_ips: 3 (number of floating ips allowed per tenant)
- quota_cores: 6 (number of instance cores allowed per tenant)
- quota_volumes: 10 (number of volumes allowed per tenant)
- quota_gigabytes: 1000 (number of volume gigabytes allowed per tenant)
- quota_ram: 2034 (megabytes of instance ram allowed per tenant)

4.2.5 Setting Internal Networking using Neutron GRE segmentation

The internal Network is used for VM data communication within the Datacenter. Using the GRE option this network is deployed by means of tunnels between each server using the OpenStack Management network addresses. In the section *Neutron L2 Configuration*, the user can select the range of Tunnel Ids: by default the range starts from 2 to 65535. In this section *Neutron L3 Configuration* the user can select the IP subnet, Gateway and Name servers (DNS) of the Internal Network: this is a default virtual network that the ITBox pre-configures as a sample network. After the ITBox installation, each OpenStack user can define its own internal network(s). The suggested configuration is to use default values of Neutron L2 and L3 Configuration sections.

4.2.5.1 Manually change of a public IPs

The ITBox basic installation sets to every server a public IP and it is associated to br-ex bridge. In order to change it, the user should edit the file /etc/network/interfaces.d/ifcfg-br-ex and replace the IP in address, netmask and gateway fields with 0.0.0.0 value.

For example:

```
auto br-ex
iface br-ex inet static
address 0.0.0.0
netmask 0.0.0.0
gateway 0.0.0.0
```

Finally, it is necessary to restart network interfaces.

4.2.6 DCRM GE

On top of nova services provided by OpenStack, FIWARE integrates the DCRM GE. The Infrastructure Toolbox v1.3.4.0 includes the DCRM GE installation as described in the deliverable D3.3 – Infrastructures Management Toolkit API- Chapter Components – paragraph 2.1 Infrastructure Toolbox [10].

The DCRM GE can be installed manually following the FIWARE installation guide [49] .

4.2.6.1 Installation and validation check

In order to verify the correctness of the installation, the following procedures should be executed:

http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/IaaS_Data_Center_Resource_Management_-_Installation_and_Administration_Guide#Diagnosis_Procedures

4.2.7 Object Storage GE (Optional)

The Object Storage GE adds functionalities on top of Swift services provided by OpenStack (as a consequence, Swift is mandatory for the Object Storage installation). For the installation and configuration of this component please check the documentation in the "FIWARE Installation and Administration GUIDE” [51]. For the release plan refer to the FIWARE documentation[17].

4.2.7.1 Installation and validation check

In order to verify the correctness of the installation, the following procedures should be executed:

http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Object_Storage_-_Unit_Testing_Plan

4.2.8 XIFI Infrastructure Monitoring Middleware (XIMM)

The following figure depicts the general architecture of the XIMM abstraction layer already introduced in the section 3.5 “XIFI Infrastructure Monitoring Middleware (XIMM)” of this document. In the figure, the reader may see how this layer is connected to the Federation Monitoring by means of the Context Broker.

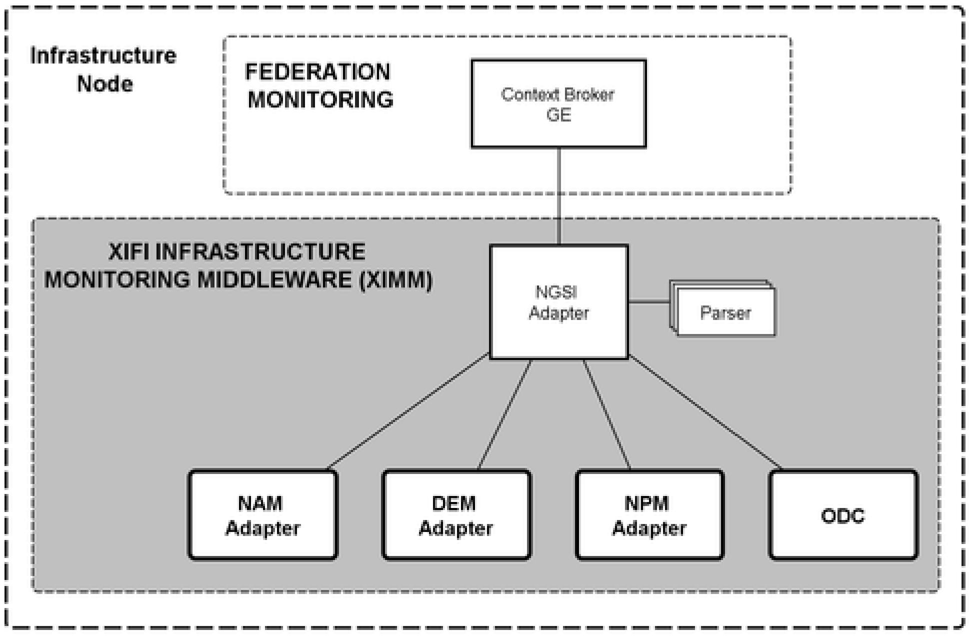


Figure 20: XIMM General Architecture

Independently from the specific role of each infrastructure node within the federation, all of them require to deploy this abstraction layer to feed the Federation Monitoring with standardized monitoring data. Otherwise, the information about the performance of the federated resources would not be able reachable to be consumed because it does not fit a common data model.

4.2.8.1 Installation and validation check

The general procedure to deploy the XIMM in each node is described below. The reader will find further details about installation and configuration in the proper deliverable in charge of specifying these particular components, D3.5 [82].

- **Setup NGSI Adapter**. The joint component is recommended to be deployed as the first step of the process. NGSI Adapter asynchronously processes all incoming adaptation requests, so that XIMM adapters are not blocked.

  A single NGSI Adapter instance may result enough to handle all the data from the XIMM adapters beneath. However, this deployment depends on the specific configuration of each node. It may occur that more than one instance improves the overall performance, or it becomes necessary because monitoring resources are distributed geographically within the same node.

  The configuration of this adapter requires setting the communication with the Context Broker. This will be the basis of the joint between the adaptation and federation layers.

- **Setup XIMM Adapter**s. It must be highlighted that each adapter is deployed in a different manner depending on the goal to achieve. A normal scenario is the establishment of a monitoring node within the cloud environment of the infrastructure by means of a dedicated physical server. Nonetheless, these software packages also allow deployments in virtual resources. Hence a general procedure cannot be provided.

  Although specific documentation is provided in deliverable D3.5, here are listed some relevant points to bear in mind:

  - NAM Adapter. It is possible to deploy this software component in a VM. Nevertheless, this configuration may carry accuracy and stability problems with the obtained values. Hence, it is strongly recommended to install the component in a physical resource. Each node in the federation shall assure the presence of at least one instance of this component to be reachable by other nodes. A node owner could deploy more instances to provide a more fine-grained status and avoid single points of failures.

- DEM Adapter. Depending on the version of this adapter, the deployment and configuration will be different. Whilst the active version requires a proper NMS, such as Nagios, to operate, DEM Passive Adapter does not rely on any system but requires to be embedded in a proper XIFI-ready image to be capable of monitoring virtual resources.
- Openstack Data Collector should be installed on each node of the federation where OpenStack is deployed.
- Each adapter will have to be configured to establish the connection with the NGSI Adapter previously deployed. This connectivity will be based on HTTP POST requests.

- **Start the components**. Once that each component is installed and configured, they are ready to start running.
- **Validation check**. In order to verify if the installation of OpenStack Data Collector component has been completed with success, it is sufficient to execute the test cases defined for this component in the deliverable D3.5 [82].

4.2.9 Federation Monitoring

The installation procedure for this component is detailed in the deliverable D2.5[7]. It must be taken into account that:

- This component must be installed on each node of the federation but a part of it (part of the Big Data Environment and the API Server - see D2.5[7]) must be installed only on a master node.
- API Server must be reachable on a public IP.
- This components depends from XIMM components and gather data through the Context Broker [59].
- Sizing of HDFS (Hadoop Distributed File System) can be tuned based on the number of resources monitored, the measures collected, the frequency of collection and the retention of data.

4.2.9.1 Installation and validation check

In order to verify if the installation of Federation Monitoring component has been successfully completed, it is sufficient to execute one (or more) of the test cases defined for this component in the deliverable D2.5 [7] where each API provided is tested.

4.2.10 Federation Access Control GE

The installation procedure of this component can be found on the FI-WARE public wiki[17].

Some specific requirements have to be satisfied for the installation on a XIFI node:

- The component communicates over SSL and therefore uses a SSL server certificate for authenticating to clients. When configuring the server certificate, the subject name in the certificate must match the public hostname of the server where the component is deployed, and this hostname must be registered in the DNS . For more information about DNS setup in XIFI, please referto deliverable D5.2 [15] to define the hostname. What matters is that any client accessing the component should be able to resolve the hostname with DNS. User can register multiple servers, i.e. IP addresses for the same Access Control GE hostname to provide DNS failover, since the GE will be deployed on multiple nodes.

- There should be a Security Probe installed on the same host in order to retrieve Access Control logs, so that the Probe can report access control events to the Security Monitoring, and de facto, the Security Dashboard. You can refer to deliverable D4.2 [12] for more details on the integration of the Security Probe.
- The component should have HTTP(s) access to the endpoints of the Identity Management GE component, in order to retrieve information about users or applications registered in the IdM GE.
- The component should be reachable over HTTPS by the Security Proxy component, so that the Proxy can request the Access Control GE for authorization decisions to specific APIs.

After going through the aforementioned installation steps, disable the use of the *Domain_Admin* role on the slave node, to prevent external modification of the GE configuration via the GE API. This is done by removing the <Policy> element *RPS: Domain_Admin_Role* from file *${GLASSFISH_DOMAIN_DIR}/config/authzforce/core-policy.xml*. Then restart the Glassfish server.

Finally, make sure the file synchronization system set up on the master node is synchronizing the local configuration directory *${GLASSFISH_DOMAIN_DIR}/config/authzforce* on the slave node. This must be a one-way synchronization process, insofar as changes on slave nodes must be ignored and overridden by changes in master nodes.

4.2.10.1 Installation / validation check

In order to verify if the installation of the component has been successfully completed, it is sufficient to execute one (or more) of the test cases defined for this component in the deliverable D2.5 [7] where each API provided is tested.

4.2.11 Security Probe(SIEM Agent)

In order to include the new slave node to the security monitoring performed in the XIFI Federation, each infrastructure owner shall proceed with the installation and configuration of the Security Probes in its node. As a general rule, one Security Probe per node is enough but several could be installed if necessary, each of them collecting from different data sources or subnets. They can be installed on a physical or virtual host (with Ubuntu/Debian operating system installed) providing that the logs/events required for the security directives applied in the XIFI Federation arrive to that host so they can be read by the Security Probe. More details about these security directives can be found in D2.5 [7].

The steps to be followed are outlined below, more detailed information will be found within the deliverable D4.4 - Baseline Tools v2 [81].

1. Installation of a Security Probe.

The Security Probe is distributed as a python script so once installed all the dependencies required (refer to D4.4 Baseline Tools v2[81]) and downloaded the software, only need to execute:

```
# python setup.py install --prefix=/usr
```

2. Configuration of a Security Probe.

The security probes require to be configured to establish the communication with the SIEM server installed in the master node (ip address and port) and the plugins used by the collection of the security events.

This is done in the file **/etc/ossim/agent/config.cfg** (see the description of the parameters in D4.4 - Baseline Tools v2[81]). The security events, once normalized in the slave nodes, will be sent to the master node using SSL (Secure Socket Layer). For this reason, it is necessary that the administrator of the master node provides the SSL certificate that needs to be installed in the host where the Security Probe is running.

3. Run the Security Probe.

Once it is configured, start Security Probe as a service in the host:

```
# sudo /etc/init.d/ossim-agent start
```

The commands that need to be executed to ensure the service is automatically started and running are available in the deliverable D4.4 Baseline Tools v2 [81].

If and only if using the Access Control GE on the XIFI node, set-up a specific parser for the Access Control logs, in order to process these logs and send security events in the proper format to the Security Probe and Monitoring. Please refer to the Security Monitoring installation manual in the deliverable D4.4 - Baseline Tools v2 [81] for the installation and configuration steps.

4.2.11.1 Installation and validation check

To verify the installation, check if a script python called *ossim-agent* is running as a daemon:

```
XIFI@node03:~$ ps -ef|grep ossim-agent

root        55045        1  12 12:26 ?         00:00:00 /usr/bin/python -OOt
/usr/bin ossim-agent -d
```

4.3 Installing a XIFI Node (Master Node)

This new section describes the steps to install and configure a new XIFI master node. As described in D1.5 - Federated Platform Architecture Version 2 Jump to: navigation, search – Chapter 3 Updates on Federation Architecture - paragraph Architecture of a XIFI Master Node [3], three groups of components are identified on the basis of their functionalities:

- User oriented services and tools
    - GUI portal (Marketplace)
    - Resource Catalogue
    - Reccomendation tool
    - Cloud Portal
    - Interoperability tools
    - Federation Manager (Registration)
    - SLA Manager
- Service and tools supporting the setup, deployment and operation of the Federation
    - Infrastructure toolbox

  - DCA (Deployment and Configuration Adapter),
  - FI-WARE PaaS GE,
  - SDC GE
- Federation Security tools
  - Security DashBoard
  - Security Monitor GE

Steps to be followed for the installation of a master node

As mentioned in the Introduction [1.1] of this document a master node has all components of a slave node (cloud computing, monitoring and security components) with addition components (User oriented services and tools , Service and tools supporting the setup, deployment and operation of the Federation and Federation Security tools components).

For these reason to know the steps to be followed in order to install and configure a new XIFI master node refer to the section Installing XIFI node (Slave Node) dedicated to the installation and configuration of a new XIFI slave node.

In addition to this, in order to become a master node, a number of XIFI components must be installed:

- IdM Management GE
- Security Proxy
- GUI portal (Marketplace)
- Resource Catalogue
- Reccomendation tool
- Cloud Portal
- Interoperability tools
- Federation Manager (Registration)
- SLA Manager
- Security DashBoard
- Security Monitor GE
- Security Proxy
- Federation Access Control GE
- FI-WARE PaaS GE
- SDC GE
- DCA (Deployment and Configuration Adapter)

4.3.1 Identity Management GE

This component is centralized and doesn’t need to be installed on a XIFI slave node to join the Federation, it must be deployed only on the XIFI master node.

To install the IdM GE, refer to the manual installation of the IdM GE [52] in particular:

- Interface to User Management GUI as reported in the D4.1- Services and Tools specification -- Chapter 3 Component Specification – paragraph 3.1 Identity Federation and Single Sign On [11];

- API spec including federation and delegation features in the D2.5 – APIs and Tools for Infrastructure v2 - Chapter 4 Sub-systems APIs [7] .

In addition and in order to enable the SAML 2.0 support, the following components have also to be installed and properly configured:

To enable SAML SP support:

- install and configure Shibboleth as SP;
- update Apache configuration.

To enable SAML IDP support

- install and configure Shibboleth as IDP;
- update Tomcat configuration;
- install and configure LDAP.

All details to perform the above mentioned activities, can be found in the deliverable D2.5 APIs and Tools for Infrastructure v2 – Appendix A Update on the Identity Management [7].

4.3.1.1 Installation and validation check

A set of tests to be executed in order to validate the installation of the IdM is published in the "Federated Identity Management GE: Test cases" wiki page [79].

4.3.2 Security Proxy

This component is centralized, thus a user (eg. Infrastructure owner) doesn't need to install it in order to setup a new slave node and to join the Federation. This component should be deployed only in the XIFI master node. The compoent owner (in particular the administrators[6] of the component) have to register the user node Openstack endpoints as a new node in this component.

4.3.2.1 Installation and validation check

The set of tests needed to validate the installation of the Security Proxy is published in the XIFI Wiki public page "Security Proxy: Test Cases" [79].

4.3.3 GUI Portal (Marketplace)

The installation procedure includes both Marketplace and the Resource Catalogue & Recommendation tool. They cannot be installed separately and they must to be installed in the same server of the master node, which contains the Django installation. Hence, the installation of this component is detailed in the section Resource Catalogue and Recommendation Tool (cif. section below 4.3.4) of this document.

4.3.4 Resource Catalogue and Recommendation Tool

The installation procedure for this component is detailed in the deliverable D4.4 - Baseline Tools v2 [81] and in the XIFI Wiki public page [84][79]. It must be taken into account that:

- It will be installed only in one master node.

[6] UPM-DIT and TID are the administrators of this component.

- It is necessary to have already installed and configured the Federation Identity Manager, in order to allow configuring the security at the beginning.
- The server where this component is installed must be reachable on a public IP.
- It can be installed in a dedicated VM or in a common server with more components (it is recommended in a dedicated). If it is installed in a server having other components, must be considered that:
    - The Python version is 2.7 (it is not supported the version 3.x) and the Django version is 1.4. Thus, if there are different versions of Python in this server, which are installed previously, it must be configured the Python virtualenv to have more than one version working together in the same server.
    - It is necessary to install the version 3.6.2 of the PyLucene. If there is another version, there will be problems with the indexation.

4.3.4.1 Installation and validation check

The set of tests needed to validate the installation of the Resource Catalogue and Recommendation tool is reported in the deliverable D4.4 - Baseline Tools v2 [81] and in the description of the component published in the public XIFI Wiki page [84].

4.3.5 Cloud Portal

This component is centralized, the IO of the nodes does not need to install it in order to setup a new slave node and to join the Federation. It should be installed only on one master node and should not be installed on a slave node.

4.3.5.1 Installation and validation check

The installation guide and the set of tests needed to validate the installation of the Cloud Portal are in the public wiki area of Cloud Portal94.

4.3.6 Interoperability tools

Before installing the Interoperability Tool, the Resource Catalogue and FIWARE Object Storage GE have to be installed. Take into account that to install this component, it is required:

- Java 7 Platform [98].
- Maven tool for building applications [99].
- Subverion [100].

The installation of each of them is described on Ubuntu v12.04. For other operating systems (e.g. Windows, Mac) please follow the individual software installationseach of them. No specific configuration is required,.

The Interoperability tool is available at https://XIFIsvn.res.eng.it/wp4/InteroperabilityTool.

For the manual installation, download the latest version of the Interoperability Tool package and extract it into the chosen location.

To install the tool, use Maven from the chosen directory location:

```
mvn install
```

This will create locally a set of JAR files that can then be executed; there are no further dependencies or installations to carry out. If it is deployed in a unix environment, then execute the run script:

```
chmod a+x run > ./run
```

In other environments, Maven can be used directly:

```
mvn exec:java -
Dexec.mainClass="uk.soton.itinnovation.XIFIinteroperability.Interoperabilit
yService
```

The service will now be hosted at port 8192 of the machine, and can be simply used by a web browser pointed to the URL: http://xxx.xxxx.xxxx.xxxx/interoperability/patterns

More details can be found in D4.2 –Baseline tools v1 – Chapter 4 Interoperability Tool **Errore. L'origine riferimento non è stata trovata.**.

4.3.6.1 Installation and validation check

The set of tests needed to validate the installation of the Interoperability service is published in the public XIFI Wiki page “Interoperability Tool: Test Cases” [93].

4.3.7 Federation Manager

It is already installed only on one master node[7] and should not be installed on a slave node, but only on a Master node. The installation procedure for this component is detailed in the deliverable D2.5 [7]. Installation / validation check

The validation process to check the installation of this component is described in the deliverable D2.5 [7].

4.3.8 SLA Manager

The installation procedure for this component is detailed in the deliverable D4.4 - Base line v2 [81], the description of the component can be found in the public XIFI Wiki page [85][91]. It must be taken into account that:

- It is necessary to have already installed and configured the Federation Identity Manager, in order to allow configuring the security at the beginning.
- Server must be reachable on a public IP.
- The component must have access to reach the Federation Monitoring API.
- It can be installed in a dedicated VM or in a common server with more components (it is recommended in a dedicated). If it is installed in a server having other components, must be considered that::
    - The Python version is 2.7 (it is not supported the version 3.x) and the Django version is 1.4. So, if there are different versions in this server, the Python virtualenv must be configured to have more than one version.
    - It is tested with the java virtual machine, Oracle JDK 1.6.x, and if there is another java virtual machine, it must be i) configured the server to coexist with different JDK or ii)

[7] Currently it has been installed in the Spain node (master node)

validated that this component is working appropriately with other version of the JDK.

- The database is the MySql and it can be installed in the same server or using an existing one and creating the required schema and user (detailed instructions are described in the deliverable D4.4 - Baseline Tools v2 [81]).

4.3.8.1 Installation and validation check

The set of tests needed to validate the installation of the SLA Manager is reported in the deliverable D4.4 - Baseline Tools v2 [81]and in the description of the component that is published in the public XIFI Wiki page [85].

4.3.9 PaaS GE

The PaaS Manager GE component has multiregion (XIFI nodes) support, so no adaptation is needed.

For the installation of this component refer to the FIWARE documentation[88].

4.3.9.1 Installation and validation check

The details on how to validate the installation of this component can be found in the FIWARE documentation [88].

4.3.10 SDC GE

The SDC GE component is not affected by the federation. No adaptation is needed.

For the installation of this component refer to the FIWARE[17].

4.3.10.1 Installation and validation check

The details on how to validate the installation of this component can be found in the FIWARE documentation[17].

4.3.11 Federation Access Control GE

The installation is the same as described for a slave node(cif. Federation Access Control GE section, page 78), except for two things:

- The PAP is kept enabled (i.e. skip the configuration step removing the *Domain_Admin* role).
- Set up a two-way file synchronization system, e.g. *csync2*, that is able to synchronize the policy repository, i.e. directory *${GLASSFISH_DOMAIN_DIR}/config/authzforce/domains* with the one on the other master node; and a one-way synchronization process with slave nodes, i.e. changes on slave nodes are ignored, only changes on master nodes are committed. Please refer to the manual installation mentioned in the slave node section (cif. Federation Access Control GE section, page 78) for the meaning of *GLASSFISH_DOMAIN_DIR*.

4.3.11.1 Installation and validation check

The installation and validation check procedure is described in section 5 (Sanity check procedures) of the Access Control GE Installation and Administration Guide [87].

4.3.12 Security Monitoring GE

For the installation of this component refer to the dedicated documentation D4.4 - Baseline Tools v2 [81], the description of the component can be found in the public XIFI Wiki [104].

It must be taken into account that:

- It can be installed in a dedicated VM or in a common server with more components but it is required Ubuntu or Debian linux distribution. It has been tested on Ubuntu with: JDK 1.6.0_37, OSSIM v4.1.0, Storm 0.9.0-wip16, Apache ZooKeeper Server Package v3.4.5, ZeroMQ v2.1.7 and Python 2.7. The database used by the Security Monitoring GE is MySql, adding the IPv6 and IDNA support functions.
- It is necessary to have access to the Security Probes installed in the slave nodes and the set of ports included in the documentation need to be opened in the firewalls. The SIEM Server must be reachable through the MD-VPN so that the SIEM agents installed in the slave nodes can send the security events collected. SSL protocol can be used in this communication and in this case a certificate needs to be generated and shared to the Security Probes installed in the slave nodes.
- The component must have access to reach the Federation Identity Manager API and Federation Monitoring API.

4.3.12.1 Installation and validation check

The set of tests needed to validate the installation of the Security Monitoring GE is reported in the deliverable D4.4 - Baseline Tools v2 [81] and in the description of the component that can be found in the public XIFI Wiki [104].

4.3.13 Security Dashboard

The Security Dashboard is centralized and needs to be installed only in one master node. The server where this component is installed must be reachable on a public IP.

The Security Dashboard depends on the Security Monitoring component. Therefore, make sure that the Security Monitoring component has been installed and configured before installing the Security Dashboard. Please refer to the previous section for the Security Monitoring installation.

The installation procedure for this component is detailed in the deliverable D4.4 - Base line v2 [81] and in the description of the component that can be found in the public XIFI Wiki page [103][91].

It is important to remark that SSL access needs to be enabled in the Apache2 configuration. Besides, the Security Dashboard is configured by default to be accessible with the hostname *securitydashboard.fi-XIFI.eu*. In case other hostname is used, the references included in the file *www/config.ini* needs to be updated to the new alias.

Finally, it is also necessary to have installed and configured the Federation Identity Manager before, in order to allow configuring this information in the *www/config.ini* at the beginning.

4.3.13.1 Installation and validation check

To check whether the Security Dashboard has been successfully installed, the following dashboard should be shown (once logged into the Identity Management) when the endpoint *https://securitydashboard.fi-XIFI.eu* is invoked:

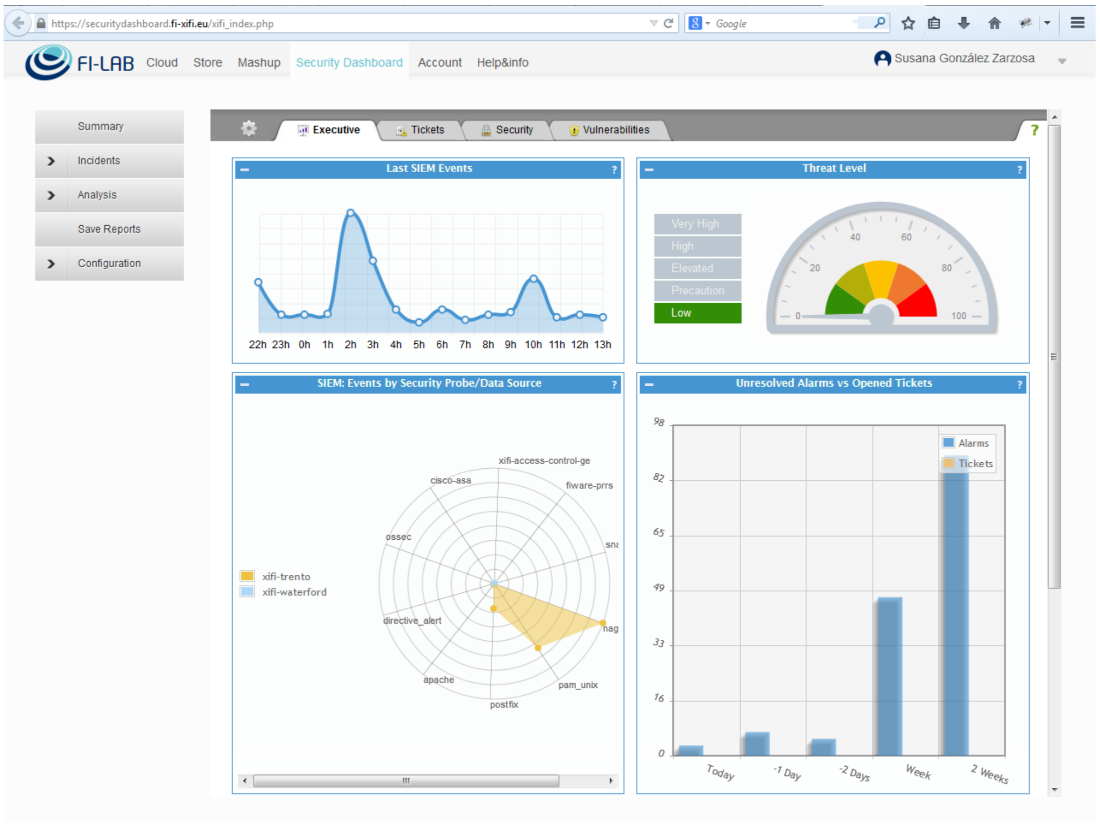


Figure 21: Security Dashboard validation check

4.3.14 Deployment and Configuration Adapter

The Deployment and Configuration Adapter is installed on one Master node [Trento node[101]].

This component depends on:

- The Cloud management platform and in particular OpenStack Nova and Glance components.
- The repository of the Generic Enablers. The Repository is holding 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.

Before installing the component verify that:

- 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) is already installed
- The cloud-hosting Generic Enablers (GEs) that support the deployment and configuration of the GEs are installed:
    - PaaS Manager;
    - SDC;
    - DCRM:
        - SM
- A persistency server is already installed: DCA currently tested with MySQL DB Server
- The availability of the recipes needed to deploy and configure a GE in an automated way (as stated in the conditions and instructions of FIWARE)

Refer to D3.3 – Infrastructures Management Toolkit API – Chapter 2 Components – paragraph 2.2 DCA [10]for more details for the installation and configuration of this component.

4.3.14.1 Installation and validation check

A set of tests to check the correct operation of a deployed instance of the Deployement and Configuration Adapter, after it has been installed and configured, is reported in the public XIFI Wiki page “DCA:Test cases” [94].

4.4 Installation and validation check for sub-systems

The details on how to validate the installation and the setup of a new XIFI node are reported in the deliverable D2.3 – Test and Validation Report v1 – Chapter 4 Tools and procedures for the installation and deployment of a XIFI node [6] for the following sub-systems:

- Security Sub-systems:
    - Identity Management GE (federated)
    - Security Proxy
- User oriented and GUI sub-system
    - Infographics Pages

If the test results obtained by the user performing those tests are not the same as reported in the deliverable D2.3 – Test and Validation Report v1 [6] refer to jira.fiware.org for any issue or question.

The tests related the other sub-system components will be provided in the future release of the deliverable D2.6 - XIFI Test and Validation Report v2.

5 CONCLUSIONS

This deliverable is intended as a reference manual that defines the procedures for installation and configuration of the new XIFI nodes (both slave and master) in the context of the federation.

This manual:

- deals with all the XIFI *components* needed to install a new XIFI node (master and/or slave) and to set up its connection to the federation;
- provides an update of the installation procedures for a slave node and describes the installation procedures for a master node;
- explains how to install and configure a slave or master XIFI node on which is installed the management of local resources and of non-conventional services.

Rather than duplicating information on how to validate the correctness of the installation and configuration of the components in a new XIFI node (which was already published in related documentation), the XIFI Handbook provides the references to that documentation[8].

Users can also get indications on how to solve issues related to the installation processes or obtain additional information to verify the correctness of the installations.

[8] It is not in the purpose of this document to give the information related the correctness of the installation and configuration components in a new XIFI node

REFERENCES

[1] The XIFI Consortium Deliverable D1.1b -XIFI Core Concepts Requirements and Architecture Draft:https://bscw.fi-XIFI.eu/pub/bscw.cgi/d44695/XIFI-D1.1b-XIFI_core_concepts_requirements_and_architecture_draft.pdf

[2] The XIFI Consortium Deliverable D1.3 Federated Platform Architecture Version 1 : https://bscw.fi-XIFI.eu/pub/bscw.cgi/d58595/XIFI-D1.3-Federated_Platform_Architecture_v1.pdf

[3] The XIFI Consortium Deliverable D1.5- Federated Platform Architecture Version 2 will be published at here: http://wiki.fi-XIFI.eu/Main_Page#Technical_Deliverables

[4] The XIFI Consortium Deliverable D2.1- XIFI Handbook v1:https://bscw.fi-XIFI.eu/pub/bscw.cgi/d58595/XIFI-D2.1-XIFI_Handbook_v1.pdf

[5] The XIFI Consortium Deliverable D2.2 - APIs and Tools for Infrastructure Federation":https://bscw.fi-XIFI.eu/pub/bscw.cgi/d59218/XIFI-D2.2-APIs_and_Tools_for_Infrastructure_Federation_v1.pdf

[6] The XIFI Consortium Deliverable D2.3 - Test and Validation Report v1 will be published at here: http://wiki.fi-XIFI.eu/Main_Page#Technical_Deliverables

[7] The XIFI Consortium Deliverable D2.5 - APIs and Tools for Infrastructure v2will be published at here: http://wiki.fi-XIFI.eu/Main_Page#Technical_Deliverables

[8] The XIFI Consortium Deliverable D3.1 - XIFI infrastructure adaptation components and API specification:https://bscw.fi-XIFI.eu/pub/bscw.cgi/d44705/XIFI-D3.1-XIFI_infrastructure_adaptation_components_API_open_specification.pdf

[9] The XIFI Consortium Deliverable D3.2 - Infrastructures monitoring and interoperability adaptation components toolkit and API :https://bscw.fi-XIFI.eu/pub/bscw.cgi/d58608/XIFI-D3.2-Infrastructures_monitoring_and_interoperability_adaptation_components_toolkit_and_API.pdf

[10] The XIFI Consortium Deliverable D3.3 - Infrastructures Management Toolkit API:https://bscw.fi-XIFI.eu/pub/bscw.cgi/d58587/XIFI-D3.3-Infrastructures_Management_Toolkit_API.pdf

[11] The XIFI Consortium Deliverable D4.1 - Services and Tools specification:https://bscw.fi-XIFI.eu/pub/bscw.cgi/d44635/XIFI-D4.1-Services_and_tools_specification.pdf

[12] The XIFI Consortium Deliverable D4.2 - Baseline Tools v1:https://bscw.fi-XIFI.eu/pub/bscw.cgi/d58659/XIFI-D4.2-Baseline_Tools_v1.pdf

[13] The XIFI Consortium Deliverable D4.3- XIFI Marketplace Implementation: https://bscw.fi-XIFI.eu/pub/bscw.cgi/d64391/XIFI-D4.3-XIFI_Marketplace_implementation.pdf

[14] The XIFI Consortium Deliverable D5.1 - Procedures and Protocols for XIFI Federation:https://bscw.fi-XIFI.eu/pub/bscw.cgi/d44719/XIFI-D5.1-Procedures_and_protocols_for_XIFI_federation.pdf

[15] The XIFI Consortium Deliverable D5.2- XIFI Core Backbone-:https://bscw.fi-XIFI.eu/pub/bscw.cgi/d64414/XIFI-D5.2-XIFI_Core_Backbone.pdf

[16] The XIFI Consortium Deliverable D5.4- XIFI federation extension and support will be published at here: http://wiki.fi-XIFI.eu/Main_Page#Technical_Deliverables

[17] For the FI-WARE documentation: https://forge.fiware.eu/plugins/mediawiki/wiki/fiware/index.php

[18] For the documentation of the OS Ubuntu LTS 12.04 see you the official documentation available at http://www.ubuntu.com/

[19] For the documentation of KVM see you the official documentation available at

http://www.linux-kvm.org/page/Main_Page

[20] For the installation of Openstack dashboard: **http://docs.openstack.org/admin-guide-cloud/content//installing-openstack-dashboard.html**

[21] For the documentation of OpenStack the official guide available at: **http://docs.openstack.org/trunk/basic-install/content/basic-install_intro.html** .

[22] Open source software for building private and public clouds: **http://www.openstack.org**

[23] OpenStack Operations Guide: **http://docs.openstack.org/trunk/openstack-ops/content/**

[24] OpenStack Conceptual Architecture: **http://docs.openstack.org/grizzly/openstack-compute/admin/content/conceptual-architecture.html**

[25] OpenStack Nova: **https://wiki.openstack.org/wiki/Nova**

[26] OpenStack Neutron: **https://wiki.openstack.org/wiki/Neutron**

[27] Open Virtual Switch: **http://openvswitch.org**

[28] OpenStack Glance: **https://wiki.openstack.org/wiki/Glance**

[29] OpenStack Keystone: **https://wiki.openstack.org/wiki/Keystone**

[30] OpenStack Cinder: **https://wiki.openstack.org/wiki/Cinder**

[31] OpenStack Swift: **https://wiki.openstack.org/wiki/Swift**

[32] OpenStack Horizon: **https://wiki.openstack.org/wiki/Horizon**

[33] Open Network Management Application Platform: **http://www.opennms.org**

[34] OpenStack Compute and Image System Requirements: **http://docs.openstack.org/grizzly/openstack-compute/admin/content/compute-system-requirements.html**

[35] OpenStack Cloud Controller Design: **http://docs.openstack.org/grizzly/openstack-ops/content/cloud_controller_design.html**

[36] OpenStack Instance Storage Solution: **http://docs.openstack.org/grizzly/openstack-ops/content/compute_nodes.html**

[37] OpenStack System Requirements: **http://docs.openstack.org/grizzly/openstack-object-storage/admin/content/object-storage-system-requirements.html**

[38] OpenStack High Availability Guide: **http://docs.openstack.org/high-availability-guide/content/index.html**

[39] OpenStack Block Storage Service Administration Guide: **http://docs.openstack.org/grizzly/openstack-block-storage/admin/content/index.html**

[40] OpenStack Networking Administration Guide: **http://docs.openstack.org/grizzly/openstack-network/admin/content/connectivity.html**

[41] OpenStack Network Operations Guide:**http://docs.openstack.org/trunk/openstack-ops/content/network_design.html**

[42] Openstack Networking Administration Guide: **http://docs.openstack.org/grizzly/openstack-network/admin/content/index.html**

[43] For the documentation of the controller node: **http://docs.openstack.org/grizzly/basic-install/apt/content/basic-install_controller.html**

[44] For the documentation of the network node: **http://docs.openstack.org/grizzly/basic-install/apt/content/basic-install_network.html**

[45] For the documentation of the compute node: **http://docs.openstack.org/grizzly/basic-**

install/apt/content/basic-install_compute.html

[46] Installing and configuring Cinder: http://docs.openstack.org/grizzly/openstack-compute/install/apt/content/cinder-install.html

[47] OVS plugin Error Failed to create OVS patch port: https://ask.openstack.org/en/question/1427/ovs-plugin-error-falled-to-create-ovs-patch-port/

[48] Identity configuration files: http://docs.openstack.org/trunk/openstack-compute/admin/content/keystone-configuration-file.html

[49] For the DCRM GE documentation:

[50] https://forge.fiware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecificaton.Cloud.DCRM

[51] For the Object Storage documentation: https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.CloudStorage .

[52] For the Identity Management GE:

- Documentation: https://github.com/ging/fi-ware-idm/wiki
- Installtion Guide: https://github.com/ging/fi-ware-idm/wiki/Installation-guide
- For the user and programmer guide: http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Identity_Management_-_KeyRock_-_User_and_Programmers_Guide
- For the description of the architecture of IdM : https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.ArchitectureDescription.Identity_Management_Generic_Enabler

[53] For the installation and configuration of Keystone Proxy you can refer to the documentation at https://github.com/ging/fi-ware-keystone-proxy/wiki/Installation-Guide .

[54] The official site of Nagios : is available at: http://www.nagios.org/

[55] The official site of OpenNMS is available at: http://www.opennms.org/

[56] The official site of Zabbix is available at: http://www.zabbix.com/

[57] For the cloud hosting architecture documentation : http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Cloud_Hosting_Architecture.

[58] For the description on how to store generic objects as images or videos as metadata:http://cordis.europa.eu/fp7/ict/netinnovation/deliverables/fi-ware/fi-ware-d432.pdf - section 1.4, chapter “Context”.

[59] For the Context Broker Orion :

- documentation: http://catalogue.fi-ware.eu/enablers/publishsubscribe-context-broker-orion-context-broker,
- a detailed description on how to federate two or more Context Brokers is available at: https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Publish/Subscribe_Broker_-_Orion_Context_Broker_-_User_and_Programmers_Guide#Context_Broker_Federation

[60] For the Big Data GEs (COSMOS) documentation: http://catalogue.fi-ware.eu/enablers/bigdata-analysis-cosmos.

[61] For the Virtual Brinding documentation:http://docs.openstack.org/grizzly/openstack-network/admin

[62] The original source for the figure18 Chapter 4 How to deploy a new XIF node – paragraph 4.2.2.3 Install Object Storage: **http://docs.openstack.org/grizzly/openstack-compute/install/apt/content/figures/6/figures/swift_install_arch.png**

[63] For the Nova client command : **http://docs.openstack.org/user-guide/content/novaclient_commands.html**

[64] For the DRCM installation guide: **https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/IaaS_Data_Center_Resource_Management_-_Installation_and_Administration_Guide** .

[65] For the configuration file of the OpenStackServices in the Keystone Proxy : **https://github.com/ging/fi-ware-keystone-proxy/blob/master/config.js.template#L24**.

[66] For the installation and administration guide of Object Storage GE: **https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Object_Storage_-_Installation_and_Administration_Guide**

[67] For the release plan refer to the FIWARE documentation: **http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Object_Storage_-_Unit_Testing_Plan**

[68] Redmine official page: **http://www.redmine.org**.

[69] Cirros documentation: **http://docs.openstack.org/image-guide/content/ch_obtaining_images.html** and **https://launchpad.net/cirros/+download**.

[70] For the DCRM GE documentation:https://forge.fiware.eu/plugins/mediawiki/wiki/fiware/index.php; https://forge.fiware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecificaton.Cloud.DCRM

[71] For the Object Storage documentation:https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.ph

[72] For the cloud hosting architecture documentation : http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php

[73] For the description on how to store generic objects as images or videos as metadata section1.4:http://cordis.europa.eu/fp7/ict/netinnovation/deliverables/fi-ware/fi-ware-d432.pdf and chapter "Context" section 1.4: http://cordis.europa.eu/fp7/ict/netinnovation/deliverables/fi-ware/fi-ware-d432.pdf.

[74] For the Context Broker Orion : documentation: http://catalogue.fi-ware.eu/enablers/publishsubscribe-context-broker-orion-context-broker;a detailed description on how to federate two or more Context Brokers is available at: https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Publish/Subscribe_Broker_-_Orion_Context_Broker_-_User_and_Programmers_Guide#Context_Broker_Federation

[75] For the Big Data GEs (COSMOS) documentation:http://catalogue.fi-ware.eu/enablers/bigdata-analysis-cosmos];[http://docs.openstack.org/grizzly/openstack-network/admin

[76] For the Virtual Brinding documentation:http://docs.openstack.org/grizzly/openstack-network/admin

[77] For the DRCM installation guide, IaaS_Data_Center_Resource_Management_-_Installation_and_Administration_Guide: https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php

[78] List of sub-systems for XIFI components can be found in the following public link:http://wiki.fi-XIFI.eu/XIFI:Sub-systems

[79] List of XIFI components can be found in the following public link: http://wiki.fi-XIFI.eu/Public:Software_Components

[80] RedMine link: https://redmine.fi-XIFI.eu

[81] The XIFI Consortium Deliverable, D4.4 - Baseline Tools v2 will be published at here: http://wiki.fi-XIFI.eu/Main_Page#Technical_Deliverables

[82] The XIFI Consortium Deliverable D3.5 - Infrastructures monitoring and interoperabilityadaptation components API v2 will be published at here: http://wiki.fi-XIFI.eu/Main_Page#Technical_Deliverables

[83] Public link for Infographics and Status Pages: http://wiki.fi-XIFI.eu/Public:Infographics_and_Status_Pages

[84] Public link for Resource Catalogue and Recomendation tool: http://wiki.fi-XIFI.eu/Public:Resource_Catalogue%26Recommender

[85] Public link for SLA Manager: http://wiki.fi-XIFI.eu/Public:SLA_Manager

[86] Public link for Future Internet Developers :http://wiki.fi-XIFI.eu/Public:D1.1#XIFI_and_stakeholders

[87] For the description of the installation procedure for Access Control GE: https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Access_Control_-_Installation_and_Administration_Guide

[88] For the description of installation procedure for PaaS GE: http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/PaaS_Manager_-_Installation_and_Administration_Guide

[89] For the description of installation procedure for Software Deployment and Configuraiton GE : http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Software_Deployment_%26_Configuration_-_User_and_Programmers_Guide

[90] FIWARE Catalogue: http://catalogue.fi-ware.org/

[91] Public process for FIWARE Catologue: http://catalogue.fi-ware.org/process

[92] Public link for Federation Monitoring : http://wiki.fi-XIFI.eu/Public:Federation_Monitoring

[93] Public link for Interoperability Tool: http://wiki.fi-XIFI.eu/Public:Interoperability_Tool

[94] Public link for DCA: http://wiki.fi-XIFI.eu/Public:Deployment_and_Configuration_Adapter

[95] Public link for Federated IdentityManagement: http://wiki.fi-XIFI.eu/Public:Federated_Identity_Management

[96] Public link for Infrastructure Toolbox: http://wiki.fi-XIFI.eu/Public:InfrastructureToolbox

[97] For ETICS are available the following information:

- The ETICS Portal available at : **http://etics.eng.it/eticsPortal/**
- The ETICS documentation available at: **http://etics.res.eng.it/doc/index.php/Main_Page**
- The ETICS tutorial videos available at: **https://www.youtube.com/user/eticsproject**

[98] For the Java Development Kit (JDK 7 and later): http://www.oracle.com/technetwork/java/javase/overview/index.html

[99] For Maven (Version 2 and later): http://maven.apache.org/download.cgi

[100] For Subversion: http://subversion.apache.org/

[101] For the description of the status of Trento node: http://wiki.fi-XIFI.eu/Public:NodeDescription#Trento_Node

[102] The XIFI Consortium Deliverable D2.6 - XIFI Test and Validation Report v2 will be published at here: http://wiki.fi-XIFI.eu/Main_Page#Technical_Deliverables.

[103] Public link for Security Dashboard: http://wiki.fi-XIFI.eu/Public:Security_Dashboard

[104] Public link for Security Monitoring: http://wiki.fi-XIFI.eu/Public:Security_Monitoring_GE

Appendix A - FAQ

This section collects the questions coming from IO and component owner registered during the XIFI lifecycle through the REDMine tool and regarding how to setup a new XIFI node. Before reading this FAQ, please refer to the dedicated section "HelpDesk" of the D5.4[16] and D2.5[7],which are:

- Official Software Component kits
- Cloud installation and Cloud management GEs
- Connectivity and Network
- Monitoring and Security/FI-LAB Joining

Additional information about the Federation Components is also available in a dedicated FAQ section of the deliverable D2.5 - XIFI federation extension and support/XI-FI Federation Inclusion FAQ .

A.1 General Questions

How can the user know the status of Infrastructure services?

The user should be able to know the status of FIWARE Lab Infrastructure services by accessing a web page with his/her browser. [refer to task#728 https://redmine.fi-XIFI.eu/issues/728#change-1658]

What are the components for the XIFI Generic node? And for the Master Node?

The list of components for XIFI Generic and Master nodes is provided here [3][3].

Which are the main documents to read in order to deploy a new node?

Here follows a list of documents that should be read and why:

- D1.1b [1]: it gives an overall picture of the project. It provides useful information to understand the rest of the deliverables.
- D5.1- Procedures and protocols for XIFI federation [14]:activities for infrastructures.. Here will be found a description of all Openstack modules, the definition of the reference architecture for deployment and needed references to useful documentation.
- D5.2 - XIFI Core Backbone [15]: this document describes the deployment plan for network backbone connectivity, and Openstack on the five legacy nodes. It also includes some guidelines in order to join the federation and the basic services performance testing.
- D2.1 - XIFI Handbook_v1[4]: an How-To manual (handbook) about the deployment of new XIFI node (slave and master) describing manual and automated procedures for (ITBox) Openstack deployment.

Additional documents that should be considered in order to deploy a new node into the XIFI federation are:

- User's request and deployment installation plan *D5.4 - XIFI federation extension and support*
- Official Site of the Ubuntu LTS12.04 [18].
- Official site of KVM [19].

A.2 Technical Question

How to deploy a XIFI node?

For a new node with 8 physical servers, consider the following deployment architecture:

- 1x ITBox node
- 3x Controller (including monitoring, network & storage) nodes
- 4x Compute nodes

Should every new node have the HA or it is up to the node to decide?

The HA deployment is recommended for production nodes. For more details refer to D5.1-Physical Deployment Models [14].

How to setup addition of extra networks (eg: public ip addresses)?

This information can be found in the deliverable D5.2 [15].

How can I set the nodes to PXE boot?

The servers are usually configured to look for PXE if no OS is present on the disk. No further actions are needed than setting all nodes administrative network on eth0, so that Fuel can act as PXE server.

How to setup the deployment of monitoring?

The easiest method is to use a dedicated node and deploy through the ITBox. In case of French node, the monitoring is running on VMs, but, at the moment, there is not a complete configuration guide.

Missing bc package on ITBox Ubuntu distribution

While analyzing Daemon logs on the controller, it was noticed that mysql daemon returns some errors related to the missing bc command. The problem is that the Ubuntu Distrib shipped with Fuel does not provide the bc command package. Fuel server is the only source declared on the sources.list file. Up to now, the solution adopted was to install it manually on each node.

Here is the output found on /var/log/daemon.log

```
 <30>Mar 18 13:47:33 node-12 mysql-wss7010: INFO: Master flag was already set, skipping our (probably legitimate yet) promotion <29>Mar 18 13:47:33 node-12 lrmd2744: notice: operation_finished: p_mysql_start_0:7010 [ /usr/lib/ocf/resource.d/mirantis/mysql-wss: line 776: bc: command not found ] <29>Mar 18 13:47:33 node-12 lrmd2744: notice: operation_finished: p_mysql_start_0:7010 [ /usr/lib/ocf/resource.d/mirantis/mysql-wss: line 776: [: : integer expression expected ] <29>Mar 18 13:47:33 node-12 lrmd2744: notice: operation_finished: p_mysql_start_0:7010 [ /usr/lib/ocf/resource.d/mirantis/mysql-wss: line 776: bc: command not found ] <29>Mar 18 13:47:33 node-12 lrmd2744: notice: operation_finished: p_mysql_start_0:7010 [ /usr/lib/ocf/resource.d/mirantis/mysql-wss: line 776: [: : integer expression expected ] <29>Mar 18 13:47:33 node-12 lrmd2744: notice: operation_finished: p_mysql_start_0:7010 [ /usr/lib/ocf/resource.d/mirantis/mysql-wss: line 776: bc: command not found ] <29>Mar 18 13:47:33 node-12 lrmd2744: notice: operation_finished: p_mysql_start_0:7010 [ /usr/lib/ocf/resource.d/mirantis/mysql-wss: line 776: [: : integer expression expected ] <29>Mar 18 13:47:34 node-12 crmd2747: notice: process_lrm_event: LRM operation p_mysql_start_0 (call=74, rc=0, cib-update=30, confirmed=true) ok
```

[From Support #1007: https://redmine.fi-XIFI.eu/issues/1007]

DRCM and HA environment

In order to have ITBox 1.4 HA deployment the following steps are suggested:

- add DCRM GE to all controller nodes
- add the required configuration to the Compute nodes
- Validate it through the execution on the first controller node the following commands:

```
puppet agent -no-daemonize -onetime -verbose -tags=dcrm

puppet agent -no-daemonize -onetime -verbose -tags=dcrm::controller
```

On the ITBOX master node in /etc/puppet/modules/dcrm/manifest/controller.pp comment the following lines:

```
file { "script_copy_pivot":
  source => "puppet:///modules/dcrm/manage.sh",
  path => "/tmp/manage.sh",
  recurse => true,
  mode => 0755
  }->
  exec { "update-db-pivot":
  path => "/usr/bin:/usr/sbin:/bin:/sbin",
  command => "sh /tmp/manage.sh ${sql_connection}",
  onlyif                    =>                    "test                    f
/usr/lib/python2.7/dist#packages/nova/db/sqlalchemy/migrate_repo/versions/162_Add_instance_stats
_table.pyc"
  }
```

Now, on the second and third node:

```
puppet agent -no-daemonize -onetime -verbose -tags=dcrm

puppet agent -no-daemonize -onetime -verbose -tags=dcrm::controller
```

Finally, on compute nodes:

```
puppet agent -no-daemonize -onetime -verbose -tags=dcrm::compute
```

To see what changes the Puppet script will produce, but without executing the changes (DRY-RUN mode), it can be used the –noop directive.

E.g.: puppet agent –no-daemonize –onetime –verbose –noop –tags=dcrm

More information can be found at http://docs.puppetlabs.com/puppet/2.7/reference/

Issue: tags drcm doesn’t work, missing Puppet log on the fuel server

These issues will have the same solution of the problems (and related solutions) reported below.If there are problems with the code presented on the task description, like the followings:

- tags=dcrm doesn’t work , missing information for the dcrm in the stdout of the command
- not puppet’s log on the fuel server (puppet_fuel_log.txt)

then a possible reason could be that the dcrm tag is searched in the HA manifest, but it does not exist.

A solution could be to modify the init.pp or the /osnailyfacter/manifests/cluster_ha.pp file and include the dcrm..

Appendix B – Continuous integration

In order to assure that the several components of XIFI architecture can be deployed on the nodes of XIFI federation, a system of continuous integration processes was identified.

This process can be divided in two main sub-processes:

1) Continuous Integration in order to validate the Infrastructure Toolbox: this will be referred to as CI1 (Continuous Integration One);

2) Continuous Integration in order to validate the Federation Layer and XIFI tools and APIs: this will be referred to as CI2 (Continuous Integration Two).

The tables below represent the description of the processes, the involved WPs and Tasks in charge to work on or manage them, the release date (always updated), the needed tools for executing and managing these processes.

	CI1	**CI2**
Description	The CI1 process is used for the test deployment and execution of the primary components involved in the nodes of XIFI federation.	The CI2 process is mainly concerned with the deployment of the Federation Layer and XIFI Tools & APIs.
Owner	The WP3 partners manage this process and execute it in the server owned by CREATE-NET partner.	WP4 and WP2/Task 2.2 manage this process.
Release Plan	Every three months can be produced a stable version of the primary components and can be used by the test-bed infrastructure supporting it.	The update of this process is linked to the CI1 process: every time that the CI1 process reaches a stable deployment version this process will be updated.
Tool used by	The software integration tool used is ETICS, which is integrated with the automated scripts as FUEL and Puppet.	ETICS will be extended in order to integrate it with the FI-WARE Cloud Services. It will support automated scripts as Chef and Puppet.

Appendix C – Tools for the development process

This section includes all the tools that help the development process of XIFI components: owner tools and third-party tools as RedMine, Etics, OpenStack, CirrOs.

REDMINE

In order to keep track of the activities done or to do by all XIFI partners, it was created a RedMine [68] web application available at http://redmine.fi-XIFI.eu/projects/XIFI .

SVN

A Version Control System, based on traditional SVN, was set up to store the software and configurations developed in the XIFI project, it is available at https://XIFIsvn.res.eng.it . XIFI users can access it through their XIFI credentials. The structure of this system is composed by a top level identified by the acronym corresponding to each XIFI Work Package (WP2, WP3, and WP4). The organization of the content of these folders does not have particular rules, but it's recommended to use the same structure so that people working cross different WPs can easily browse the different levels following the proposed schema below:

WPx

- software
- trunk[component-name]
- branches
- tags
- design

The software directory follows the classical SVN approach. The design directory is meant to collect and share diagrams and image files usually reported in the on-line documentation or deliverables.

ETICS

The ETICS system [97] is software building tool that allows to manage a number of activities: starting from the source code repositories up to producing the packages that are suitable for the deployment in the target machine. It's fundamentally independent from any languages and technologies used to build the software, and provides facilities for building integration and quality assurance, by executing quality assurance analysers like Findbugs and Checkstyle and many others, for different languages and technologies.

The most important characteristic is that the build-plan and the test-plan are defined only once. It is important to say that in ETICS these processes are sequential. To fully respect the Continuous Integration, at first the user should define all the necessary parameters and criterias to build the code and then ETICS will perform a test-plan.

ETICS Build and Test process consists of specific steps. In this section will be explained the High-level process of ETICS Build and Test, for more details refer to the deliverable D2.3[6].

ETICS Build:

1. The developer creates and configures an ETICS Component to point at a specific Source Code control system (as an SVN Server) where is contained the source code.
2. The developer inserts the specific commands to run the checkout from the repository and other commands to perform the build process.
3. The developer configures parameters to correctly perform the build, including the type of the packet generated from the build result (for example: .tar.gz, .deb, .rpm or .exe).

4. The developer submits the build and waits for the report results.

ETICS Test process:

If the Build process successfully ends:

1. ETICS generates a report;
2. The user can configure a test plan to deploy his/her Application;
3. The user can test his/her Application in accordance with the respective needs.

The developer should perform these activities in order to get an exhaustive test:

1. Creation of the Test Node in the ETICS Web Interface
2. Configuration of the nodes dependencies
3. Configuration of the test commands on the nodes.

THIRD-PARTY PRODUCTS

This section lists the software of third-parties used in the development process of XIFI. It is a list of externals references and no description will be provided. Each involved partner will declare the third-party products used.

- OpenStack [23];
- CirrOS[69]: it is a Tiny OS specialized in running on a cloud;
- Ubuntu LTS [18] ;
- KVM [19].

Appendix D - Terminology

The following table helps the user to understand some terminology used in this manual:

Name	Description
Cinder	It manages storage volumes.
Domain	It is an instance of an operating system.
Glance	The OpenStack image cache.
Hypervisor	Software which runs virtual machines (VMs), e.g. KVM, QEMU, Xen, VMware, HyperV.
Node	One of the servers in the system. A single chassis (sometimes called a server).
Nova	The OpenStack Compute module for VM deployment.
Swift	A reference to OpenStack image cache.

Appendix E – Log-In Files

This table lists the directories of log files for some used services as Apache, Keystone, etc.

Services Directory	File
Apache	/var/log/apache2/ error.log
Keystone	/var/log /keystone/ keystone.log
Glance	/ var/log /glance/ registry.log / var/log /glance/api.log
Nova	/var/log/nova/nova-api.log /var/log/nova/nova-conductor.log /var/log/nova/nova-manage.log /var/log/nova/nova-cert.log /var/log/nova/nova-consoleauth.log /var/log/nova/nova-scheduler.log /var/log/nova/nova-compute.log
Cinder	/var/log/cinder/cinder-api.log /var/log/cinder/cinder-scheduler.log /var/log/cinder/cinder-volume.log
Rabbit	/var/log /rabbitmq/<rabbit_user>@<hostname>.log /var/log /rabbitmq/<rabbit_user>@<hostname>-sasilog /var/log /rabbitmq/shutdown_log /var/log /rabbitmq/startup_log
Quantum	/ var/log /quantum/ server.log

Appendix F – Configuration files

This table lists the directories of the services configuration files:

Services	Directory	File
Keystone	/etc/keystone	keystone.conf
Glance	/etc/glance	glance-api.conf glange-registry.conf
Nova	/etc/nova	api-paste.ini nova.conf
Cinder	/etc/cinder	cinder.conf api-paste.ini
Quantum	/etc/quantum	quantum.conf api-paste.ini
Quantum-ovs-plugin	/etc/quantum/plugins/openvswitch/	ovs_quantum_pluing.ini

Appendix G – Scripts

This section lists some useful scripts for the configuration of Keystone, Glance services:

sample-data.sh, this script populate the database of the keystone

```
#!/usr/bin/env bash
# Copyright 2013 OpenStack LLC
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0
# Unless required by applicable law or agreed to in writing,
software
# distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT
# WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the
# License for the specific language governing permissions and
limitations
# under the License.
# Sample initial data for Keystone using python-keystoneclient
# This script is based on the original DevStack keystone data.sh
script.
# Tenant   User  Roles
# demo     admin admin
# service  glance     admin
# service  nova  admin
# service  cinder     admin
# service  quantum admin
# service  ec2   admin
# service  swift admin
ADMIN PASSWORD=changeit
GLANCE PASSWORD=changeit
NOVA PASSWORD=changeit
CINDER PASSWORD=changeit
QUANTUM PASSWORD=changeit
EC2 PASSWORD=changeit
SWIFT PASSWORD=changeit
CONTROLLER PUBLIC ADDRESS=${CONTROLLER PUBLIC ADDRESS:-localhost}
CONTROLLER ADMIN ADDRESS=${CONTROLLER ADMIN ADDRESS:-localhost}
CONTROLLER INTERNAL ADDRESS=${CONTROLLER INTERNAL ADDRESS:-
localhost}
TOOLS D1R=$(cd $(dirname "$0") && pwd)
KEYSTONE CONF=${KEYSTONE CONF:-/etc/keystone/keystone.conf}
if [[ -r "$KEYSTONE CONF" ]]; then
 EC2RC="$(dirname "$KEYSTONE CONF")/ec2rc"
elif [[ -r "$TOOLS DIR/../etc/keystone.conf" ]]; then
# assume git checkout
 KEYSTONE CONF="$TOOLS DIR/../etc/keystone.conf"
 EC2RC="$TOOLS DIR/../etc/ec2rc"
else
 KEYSTONE CONF=""
 EC2RC="ec2rc"
fi
```

```
# Extract some info from Keystone's configuration file
if [[ -r "$KEYSTONE CONF" ]]; then
 CONFIG SERVICE TOKEN=$(sed 's/[[:space:]]//g' $KEYSTONE CONF
 grep ^admin token= I cut -d'=' -f2)
 CONFIG ADMIN PORT=$(sed 's/[[:space:]]//g' $KEYSTONE CONF i grep
 ^admin port= i cut -d'=' -f2)
fi
export SERVICE TOKEN=${SERVICE TOKEN:-$CONFIG SERVICE TOKEN}
if [[ -z "$SERVICE TOKEN" 1]; then
 echo "No service token found."
 echo "Set SERVICE TOKEN manually from keystone.conf admin token."
 exit 1
fi
export   SERVICE   ENDPOINT=${SERVICE   ENDPOINT:-http://$CONTROLLER
PUBLIC ADDRES S:${CONFIG ADMIN PORT:-35357}/v2.0}
function get id () {
echo '"$@" I grep ' id ' I awk '{print $4}''
# Default tenant
echo "Creating tenants ..."
DEMO TENANT=$(get id keystone tenant-create --name=demo \
--description "Default
Tenant")
ADMIN USER=$(get id keystone user-create --name=admin \
--pass="${ADMIN PASSWORD}")
ADMIN ROLE=$(get id keystone role-create --name=admin)
keystone user-role-add --user-id $ADM1N USER \
--role-id $ADM1N ROLE \
--tenant-id $DEMO TENANT
echo "Creating services ..." # Service tenant
SERVICE TENANT=$(get id keystone tenant-create --name=service \
--description "Service
Tenant")
GLANCE USER=$(get id keystone user-create --name=glance \
--pass="${GLANCE PASSWORD}"
--tenant-id
$SERVICE TENANT)
keystone user-role-add --user-id $GLANCE USER \
--role-id $ADM1N ROLE \
--tenant-id $SERVICE TENANT
NOVA USER=$(get id keystone user-create --name=nova \
--pass="${NOVA PASSWORD}" \
--tenant-id $SERVICE TENANT)
keystone user-role-add --user-id $NOVA USER \
--role-id $ADM1N ROLE \ --tenant-id $SERVICE TENANT
CINDER USER=$(get id keystone user-create --name=cinder \
--pass="${CINDER PASSWORD}"
--tenant-id $SERVICE TENANT)
keystone user-role-add --user-id $C1NDER USER \
--role-id $ADM1N ROLE \
--tenant-id $SERVICE TENANT
QUANTUM USER=$(get id keystone user-create --name=quantum \
--pass="${QUANTUM PASSWORD}" \
--tenant-id
$SERVICE TENANT)
keystone user-role-add --user-id $QUANTUM USER \
--role-id $ADM1N ROLE \ --tenant-id $SERVICE TENANT
EC2 USER=$(get id keystone user-create --name=ec2 \
--pass="${EC2 PASSWORD}" \
--tenant-id $SERVICE TENANT)
keystone user-role-add --user-id $EC2 USER \
--role-id $ADM1N ROLE \
--tenant-id $SERVICE TENANT
SWIFT USER=$(get id keystone user-create --name=swift \
--pass="${SWIFT PASSWORD}" \
--tenant-id $SERVICE TENANT)
keystone user-role-add --user-id $SW1FT USER \
--role-id $ADM1N ROLE \
--tenant-id $SERV10E TENANT echo "Creating endpoints ..."
# Keystone service
KEYSTONE SERVICE=$(get id \
keystone service-create --name=keystone \
--type=identity \
--description="Keystone  Identity  Service")  if  [[  -z  "$DISABLE
ENDPOINTS" ]]; then
keystone  endpoint-create  --region  RegionOne  --service-id  $KEYSTONE
SERVICE \
--publicurl
"http://$CONTROLLER PUBLIC ADDRESS:\$(public port)s/v2.0" \
--adminurl
"http://$CONTROLLER ADMIN ADDRESS:\$(admin port)s/v2.0" \
--internalurl
"http://$CONTROLLER INTERNAL ADDRESS:\$(public port)s/v2.0" fi
# Nova service
NOVA SERVICE=$(get id \
keystone service-create --name=nova \
--type=compute \
--description="Nova Compute Service") if [[ -z "$DISABLE ENDPOINTS"
]]; then
keystone   endpoint-create   --region   RegionOne   --service-id   $NOVA
SERVICE \
--publicurl
"http://$CONTROLLER  PUBLIC  ADDRESS:\$(compute  port)s/v1.1/\$(tenant
i d)s" \
--adminurl
"http://$CONTROLLER  ADMIN  ADDRESS:\$(compute  port)s/v1.1/\$(tenant
id )s" \
--internalurl
"http://$CONTROLLER           INTERNAL           ADDRESS:\$(compute
port)s/v1.1/\$(tenant id)s"
fi
# Quantum service
QUANTUM SERVICE=$(get id \
keystone service-create --name=quantum \
--type=network \
--description="Networking  Service")  if  [[  -z  "$DISABLE  ENDPOINTS"
]]; then
keystone  endpoint-create  --region  RegionOne  --service-id  $QUANTUM
SERVICE \
--publicurl  "http://$CONTROLLER PUBLIC ADDRESS:9696/" \
--adminurl "http://$CONTROLLER PUBLIC ADDRESS:9696/" \
--internalurl "http://$CONTROLLER PUBLIC ADDRESS:9696/" fi
# Cinder service
CINDER SERVICE=$(get id \
keystone service-create --name=cinder \
--type=volume \
--description="Cinder  Service")  if  [[  -z  "$DISABLE  ENDPOINTS"  ]];
then
keystone  endpoint-create  --region  RegionOne  --service-id  $CINDER
SERVICE \
--publicurl="http://$CONTROLLER    PUBLIC    ADDRESS:8776/v1/%(tenant
id)s"
--internalurl="http://$CONTROLLER   PUBLIC   ADDRESS:8776/v1/%(tenant
id) s" \
--adminurl="http://$CONTROLLER  PUBLIC  ADDRESS:8776/v1a(tenant  id)s"
fi
# Image service
GLANCE SERVICE=$(get id \
keystone service-create --name=glance \
--type=image \
--description="Glance Image Service")
if [[ -z "$DISABLE ENDPOINTS"    then
keystone  endpoint-create  --region  RegionOne  --service-id  $GLANCE
SERVICE \
--publicurl "http://$CONTROLLER PUBLIC ADDRESS:9292" \
--adminurl "http://$CONTROLLER ADMIN ADDRESS:9292" \
--internalurl "http://$CONTROLLER INTERNAL ADDRESS:9292" fi
# EC2 service
EC2 SERV10E=$(get id \
keystone service-create --name=ec2 \
--type=ec2 \
--description="EC2 Compatibility Layer")
if [[ -z "$DISABLE ENDPOINTS" ]]; then
keystone   endpoint-create   --region   RegionOne   --service-id   $EC2
SERVICE \
--publicurl
"http://$CONTROLLER PUBLIC ADDRESS:8773/services/Cloud" \
--adminurl
"http://$CONTROLLER ADMIN ADDRESS:8773/services/Admin" \
--internalurl
"http://$CONTROLLER INTERNAL ADDRESS:8773/services/Cloud" fi
# Swift service
SWIFT SERVICE=$(get id \
keystone service-create --name=swift \
--type="object-store" \
--description="Swift  Service")  if  [[  -z  "$DISABLE  ENDPOINTS"  ]];
then
keystone  endpoint-create  --region  RegionOne  --service-id  $SW1FT
SERVICE \
--publicurl
"http://$CONTROLLER PUBLIC ADDRESS:8888/v1/AUTH \$(tenant id)s" \
--adminurl "http://$CONTROLLER ADMIN ADDRESS:8888/v1" \
--internalurl
"http://$CONTROLLER INTERNAL ADDRESS:8888/v1/AUTH \$(tenant id)s" fi
#  create  ec2  creds  and  parse  the  secret  and  access  key  returned
RESULT=$(keystone ec2-credentials-create --tenant-id=$SERV10E TENANT
--user-id=$ADM1N USER)
ADMIN ACCESS='echo "$RESULT" I grep access I awk '{print $4)'' ADMIN
SECRET='echo "$RESULT" i grep secret I awk '{print $4)''
# write the secret and access to ec2rc
cat > $EC2RC «EOF
ADMIN ACCESS=$ADM1N ACCESS
ADMIN SECRET=$ADM1N SECRET
EOF
echo "Status: DONE!!!"
```

To reset keystone database, use the reset-keystone.sh:

```
#!/bin/sh # Reset Keystone database
MYSQL USERNAME='root'
MYSQL PWD='root'
KEYSTONE PWD='changeit'
echo "Dropping keystone database ..."
mysql -u $MYSQL USERNAME -p$MYSQL PWD -e "drop database keystone"
mysql -u $MYSQL USERNAME -p$MYSQL PWD -e "create database keystone"
mysql -u $MYSQL USERNAME -p$MYSQL PWD -e "grant all privileges on
keystone.* TO 'keystone'@'localhost' identified by '$KEYSTONE PWD'"
echo "Launching keystone db sync ..."
keystone-manage db sync
echo "Status: DONE!!!"
```

To reset glance database, use the script reset-glance.sh:

```
#!/bin/sh
# Reset glance database
MYSQL USERNAME='root'
MYSQL PWD='root'
GLANCE PWD='changeit'
mysql -u $MYSQL USERNAME -p$MYSQL PWD -e "drop database glance"
mysql -u $MYSQL USERNAME -p$MYSQL PWD -e "create database glance"
mysql  -u  $MYSQL  USERNAME  -p$MYSQL  PWD  -e  "grant  all  privileges  on
glance.*   TO   'glance'@'localhost'   identified   by   '$GLANCE   PWD'"
glance-manage db sync
```

Appendix H - Identity Configuration Files Tables

The following table gives an overview of the sections for the Identity Configuration Files:

Section	Description
[DEFAULT]	General configuration
[sql]	Optional storage backend configuration
[ec2]	Amazon EC2 authentication driver configuration
[s3]	Amazon S3 authentication driver configuration
[identity]	Identity Service system driver configuration
[catalog]	Service catalog driver configuration
[identity]	Identity Service system driver configuration
[token]	Token driver configuration
[policy]	Policy system driver configuration for RBAC
[signing]	Cryptographic signatures for PKI based tokens
[ssl]	SSL configuration