Horizon 2020 Program (2014-2020)

Big data PPP

Research addressing main technology challenges of the data economy

Industrial-Driven Big Data as a Self-Service Solution

D5.3: Federated Resource Management for Data Analytics v2[†]

Abstract: The content of this report is mainly focused on the setup of the infrastructure layer which includes the selected underlying storage and processing infrastructure of the I-BiDaaS solution. The document also describes the preliminary work carried out on the distributed large-scale layer which is responsible for the orchestration and management of the underlying physical computational and storage infrastructure.

Contractual Date of Delivery	31/12/2019
Actual Date of Delivery	31/12/2019
Deliverable Security Class	Public
Editor	*Enric Pages (ATOS)*
Contributors	ATOS, BSC, UNSPMF, ITML, FORTH
Quality Assurance	*George Bravos (ITML)* *Giorgos Vasiliadis (FORTH)* *Kostas Lampropoulos (FORTH)*

[†] *The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780787.*

The *I-BiDaaS* Consortium

Foundation for Research and Technology – Hellas (FORTH)	Coordinator	Greece
Barcelona Supercomputing Center (BSC)	Principal Contractor	Spain
IBM Israel – Science and Technology LTD (IBM)	Principal Contractor	Israel
Centro Ricerche FIAT (FCA/CRF)	Principal Contractor	Italy
Software AG (SAG)	Principal Contractor	Germany
Caixabank S.A. (CAIXA)	Principal Contractor	Spain
University of Manchester (UNIMAN)	Principal Contractor	United Kingdom
Ecole Nationale des Ponts et Chaussees (ENPC)	Principal Contractor	France
ATOS Spain S.A. (ATOS)	Principal Contractor	Spain
Aegis IT Research LTD (AEGIS)	Principal Contractor	United Kingdom
Information Technology for Market Leadership (ITML)	Principal Contractor	Greece
University of Novi Sad Faculty of Sciences (UNSPMF)	Principal Contractor	Serbia
Telefonica Investigation y Desarrollo S.A. (TID)	Principal Contractor	Spain

Document Revisions & Quality Assurance

Internal Reviewers

1. *George Bravos, (ITML)*
2. *Giorgos Vasiliadis, (FORTH)*
3. *Kostas Lampropoulos, (FORTH)*

Revisions

Version	Date	By	Overview
0.1	25/10/2019	ATOS	Initial ToC
0.2	12/11/2019	ALL	Minor changes after brainstorming
0.3	19/10/2019	ATOS	Infrastructure layer inputs
0.4	22/11/2019	ITML	Integration process inputs
0.5	25/11/2019	UNSPMF	Runtime environment inputs
0.6	25/11/2019	BSC	COMPs/Hecuba programming models
0.7	26/11/2019	ATOS	RMO contributions
0.8	15/12/2019	ATOS	Contributions merge
0.9	20/12/2019	Internal Reviewers	Document for review
1.0	30/12/2019	ATOS	Final document

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API: Application Programming Interface
CI: Continuous Integration
CPU: Central Processing Unit
CSP: Cloud Service Provider
GPGPUs: General Purpose Graphics Processing Unit
GPU: Graphics Processing Unit
HPC: High Performance Computing
IT : Information Technologies
ITER: Instituto Tecnológico de Energías Renovables
KVM: Kernel-based Virtual Machine
MVP: Minimum Viable Product
OASIS: Organization for the Advancement of Structured Information Standards
PoC: Proof of Concept
RMO: Resource Management and Orchestration
SQL: Structured Query Language
TOSCA: Topology and Orchestration Specification for Cloud Applications
VM: Virtual Machine
WP: Work Package

Executive Summary

I-BiDaaS project goal is to create new opportunities for self-service analytics aiming to achieve a unified Big Data as-a-service solution that will empower non-expert users to easily take advantages of the Big-Data technologies while at the same time increasing the speed of data analytics. For achieving this vision, one of the key aspects is the capability of the platform to operate handling large data volumes.

This report summarizes the work carried out in the scope of "Task 5.1 Provision and configuration of infrastructure resources" and "Task 5.2 Resource Management and optimized automatic usage of computational and storage resources". The document presents the second version of the Federated Resource Management for Data Analytics Report which covers the period from M13 to M24. The work described covers the re-configuration and maintenance processes followed to setup the infrastructures required to support the 1st version of I-BiDaaS. In addition, the document updates the design and specification of the Resource Management and Orchestration (RMO) software modules presented in D5.1, describing the environment set up for orchestrating resources on top of the I-BiDaaS infrastructure layer.

While the first period reported under D5.1 has been focused on the provisioning and management of the infrastructure, the main steps within this period have been focused on the orchestration of computational resources for Big Data applications and I-BiDaaS use cases.

Finally, the document describes the future milestones for the realisation of the final I-BiDaaS solution to be delivered at M30.

1 Introduction

1.1 Overview and Objectives

One of the project main aims is to create new opportunities for self-service analytics towards a completely paradigm tailored to big data analytics. For achieving this vision, one of the key aspects is the capability of the platform to operate handling large data volumes. As such, WP5 within I-BiDaaS aims to provide the distributed large-scale framework that permits powerful and scalable data processing on top of heterogenous and federated infrastructures.

This report has been focussed on the following WP objectives documented in the I-BiDaaS DoA:

- Management of diverse infrastructure resources
- Orchestration of computational resources across diverse resource providers
- Integration and exploitation of infrastructure elasticity capabilities.

To this end, I-BiDaaS will offer a unified Big Data as-a-service solution that will empower non-expert Big-Data users to easily take advantages of the Big-Data technologies while at the same time increasing the speed of data analytics. This report updates the content described as part of "D5.1. Federated Resource Management for Data Analytics v1" [1] where the initial provisioning and setup of the I-BiDaaS infrastructure has been initially described.

1.2 Relation to other Tasks and Work Packages

The content of this report updates the work carried out in the scope of Tasks 5.1 and 5.2. The systems and software modules presented in this document have a direct relationship with the ones previously documented for "D5.1"[1] and "D5.2"[2] public reports.

Due to the provisioning and management of the computational resources plays an important role within the project solution, this report has links with other technical reports across WP2, WP3, and WP4 as well as with the experiments executed in the scope of WP6.

In order to get better understanding about the content of this report, the suggested reading path is the following:

- "D1.3 Positioning of I-BiDaaS" → which provides an overview on the industrial challenges of the data economy as well as setting the scene for the realisation of the I-BiDaaS platform.
- "D1.2 Architecture definition" → which describes the specification for I-BiDaaS architecture and its software modules.
- "D5.1 Federated Resource Management for Data Analytics v1" → where the operational infrastructure environment supporting the realisation of the I-BiDaaS MVP prototype has been defined.
- "D5.2 Big-Data-as-a-Self-Service Test and Integration Report" → which describes the testing and integration work carried out towards the MVP prototype (M12).
- "D5.4 Big-Data-as-a-Self-Service Test and Integration Report" → which describes the testing and integration work carried out towards the 1st I-BiDaaS prototype (M18).

1.3 Target Audiences

The primary target of the document is internal I-BiDaaS technicians from WP2 to WP5 involved in the prototyping and implementation of the platform. Additionally, this document can be also interesting for external technical personal that are willing to adopt the I-BiDaaS solution or/and Cloud Service Providers (CSPs) willing to be incorporated as infrastructure and data providers within our solution.

1.4 Structure of the Document

The outline of this document is as follows: The first chapter introduces the document and its objectives. The second chapter describes the re-configuration and maintenance processes followed within the project to setup the infrastructures resources required. The third chapter presents the work carried out in the scope of Task 5.2, where the Resource Management and Orchestration software module was implemented. The runtime environment providing the execution environment for data analytics is described in section 4, together with the approach selected for the integration task of the project towards the 1st I-BiDaaS prototype. Finally, the last section provides the conclusions and next steps (Section 5).

2 Provision and Maintenance of Infrastructures Resources

2.1 Cloud Testbed Overview

ATOS' Infrastructure Services offered in I-BiDaaS are delivered on top of the hardware resources from *Instituto Tecnológico de Energías Renovables* (ITER) Data Centre located in the Canary Islands. Among the different high-performance computing facilities and data infrastructures providers considered during the elicitation of requirements phase, the final candidate selected for I-BiDaaS is the "*Teide Supercomputer*", which is a general-purpose High-Performance Computing infrastructure, housed in a D-Alix [3] datacentre which provide high available electrical and cooling infrastructure, and high-speed internet connectivity.

The Teide-HPC supercomputer is composed of 1100 Fujitsu computer servers, the core of the compute nodes is Fujitsu PRIMERGY CX250 S1 servers housed in a PRIMERGY CX400 chassis grouped in 4 nodes per chassis, and featured with the latest Intel Sandy Bridge processors.

For further details, check "D5.1. Federated Resource Management for Data Analytics" [1].

2.2 Commodity Cluster Overview

FORTH's commodity cluster contains several modern off-the-shelf commodity GPGPUs, such as NVIDIA GeForce GTX 1080 Ti and NVIDIA TITAN Xp. Such GPGPUs offer extremely high processing throughput for parallelizable workloads with a low power cost.

In addition to this kind of GPUs, also known as discrete or dedicated, there is another type of such processing unit, called integrated GPU. An integrated GPU is packed inside the CPU's die, such as in the case of the Intel HD Graphics. The main characteristic that differentiates these two kinds of GPUs is the memory space, where discrete GPUs have their own dedicated memory space, while integrated GPUs share the same memory space with the CPU. Apparently, integrated GPUs are more power-efficient than discrete GPUs. FORTH's commodity cluster also contains another type of hardware accelerator, which is the Intel Phi coprocessor, as well as powerful Intel processors (based on the Skylake micro-architecture or newer) enabled with the Intel Software Guard Extensions (Intel SGX), which is a set of CPU instruction codes that allows user-level code to allocate private regions of memory, called enclaves, that are protected from processes running at higher privilege levels.

For further details, check "D5.1. Federated Resource Management for Data Analytics" [1].

2.3 Cloud Testbed Maintenance

During the first period (until M12), the work carried out has been focused on provisioning and setting up the testbed provided by ATOS in order to experiment and validate the project outcomes related to the Minimum Viable Product (MVP) version of the platform. The next six months after this period (from M12 to M18), has been devoted to the re-configuration and maintenance of the cloud environment in order to support the new developments towards the 1st I-BiDaaS prototype.

The most relevant update after MVP has been performed on top of the so-called "ibidaas-db" virtual resource, which has been used during the first period as central storage repository for the different software modules across the platform. After M12, the aforementioned system has been extended in order to act as a cluster manager of one or more computers running Docker, allowing us to orchestrate the computational resources used for the data analysis, being able to apply different set of scheduling policies.

Additionally, other examples of maintenance activities related with Task 5.1 are listed below:

- User access management.
- Resizing computational capacity of virtual resources.
- Resizing storage capacity of virtual resources. (with and without LVM support)
- Re-installation of a malfunctioning physical node.
- Re-configure the previous private Cloud Service Provider environment from all-in-one mode to a multi-node setup.
- Extend the previous computational capacity configuring one extra Nova compute node for our private cloud provider.

The reason behind supporting Docker containers technology for the 1st I-BiDaaS prototype was twofold. On one hand, it allowed us to evolve the previous integrated version of the platform (MVP) while the RMO software module has been implemented and tested, in that sense we were able to run in parallel both implementation/integration processes avoiding that any issue in one of the processes affects the other. On the other hand, supporting docker in our 1st version of the platform allows us to ship a lighter version of our solution for early adoption or demonstration of our findings on 3rd party premises. Thanks to that approach, we avoid that future adopters are forced to build their own private Cloud Service Provider for evaluating if the I-BiDaaS solution fits their needs.

2.4 Commodity Cluster Maintenance

FORTH's commodity cluster has been updated by adding two more NVIDIA TITAN Xp GPUs, that were donated from the NVIDIA's GPU Grant Program. The work carried out in this environment has been described as part of the WP4 technical activities. For further information, check "D4.2 Distributed event-processing engine" [4].

3 Resource Management and Optimized Usage of Resources

3.1 Introduction

I-BiDaaS solution is based on three main layers: the infrastructure layer, the distributed large-scale layer, and the application layer. The software module presented in this section belongs to the distributed large-scale layer. The Resource Management and Orchestration software module (RMO) is the responsible of interfacing with the infrastructure layer, allowing to programmatically interact with the set of virtual and physical resources that the application requires. The component aims to support multiple providers and various computational resource types.

After evaluating various technologies through the realization of Proof of Concepts, this report presents the technology enablers selected for the realization of the I-BiDaaS solution at large-scale.

The following subsections provide an update of the design and specification of the Resource Management and Orchestration (RMO) software modules presented in D5.1 [1]. In addition, they describe the environment setup for orchestrating the virtual/containerized resources on top of the infrastructure layer. This software system is the glue between two layers, allowing COMPs and Hecuba workers to deploy the resources planned to accommodate the needs of the application layer.

The work described in this section, which is part of Task 5.2, aims to achieve a fully operational deployment across diverse Cloud Service Providers. An enhanced version of the system described in this report will be presented at the end of the project as part of "D5.5 Federated Resource Management for Data Analytic v3".

3.2 Resource Management and Orchestrator System Capabilities

3.2.1 Technology enablers and Tool Chain Selected

This section presents the set of technologies and systems that have been selected to suit the I-BiDaaS solution and uses cases.

The following table summarizes the technologies selected for our RMO system. The technologies have been selected after a careful evaluation of the existing tools that are currently available in the market and are capable to accommodate the I-BiDaaS testing environment.

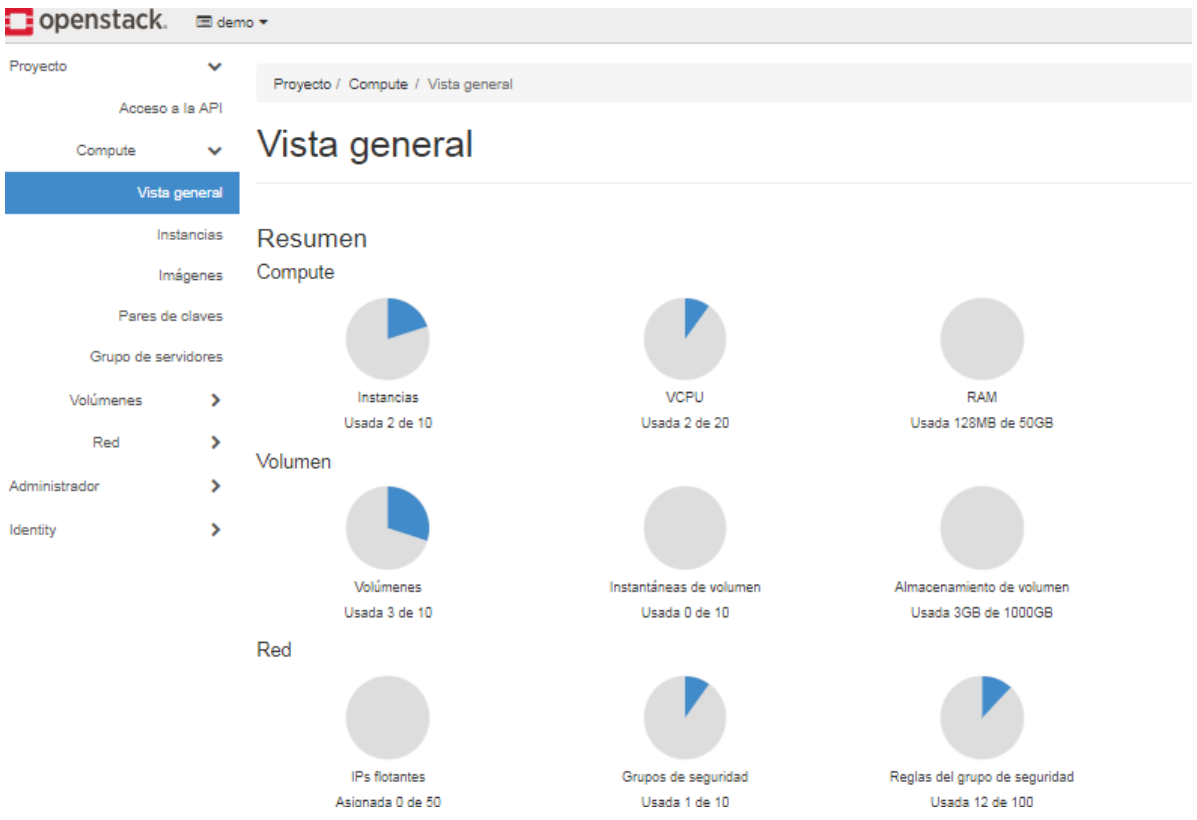
Table 1: Summary of technologies selected for I-BiDaaS RMO system

Cloudify
Cloudify is a Cloud Orchestrator used to manage the interconnection and interaction among cloud-based entities. The orchestration refers to the automation of processes and workflows required to meet the application's performance goals, minimizing the associated deployment and operation costs while maximizing the application performance. This technology offers us a reliable way to abstract various Cloud Service Providers.
TOSCA
TOSCA is an OSAIS standard specifications used to describe cloud web services and their relationships. The language includes specifications to create or modify the web-services associated to a cloud-based topology. The usage of a well-known standard aims to ensure interoperability and sustainability in the future.

OpenStack
OpenStack is the cloud software stack used to control large pools of compute, storage and networking resources in our Private Cloud environment.
AWS EC2, S3 Services
AWS offers cloud computing services on-demand. AWS EC2 and S3, which are the main compute and storage services offered, have been used as a Public Cloud environment for testing.
Docker & Docker Swarms
Docker provides software components and tools for ship and run applications. It becomes very useful during software development as it is a lightweight manner to deploy an application in comparison of using an entire VM. It also offers good support for CI/CD systems while simplifies deployment to different infrastructures. It has been used to support the deployments of the 1st I-BiDaaS prototype.

3.2.1.1 Private Cloud Environment

The OpenStack distribution used at the time of the PoCs was initially configured in all-in-one mode. This environment has been reconfigured to work in multi-node and has also been extended with an additional compute node, allowing larger deployments.



Vista general

Resumen

Compute

Instancias Usada 2 de 10

VCPU Usada 2 de 20

RAM Usada 128MB de 50GB

Volumen

Volúmenes Usada 3 de 10

Instantáneas de volumen Usada 0 de 10

Almacenamiento de volumen Usada 3GB de 1000GB

Red

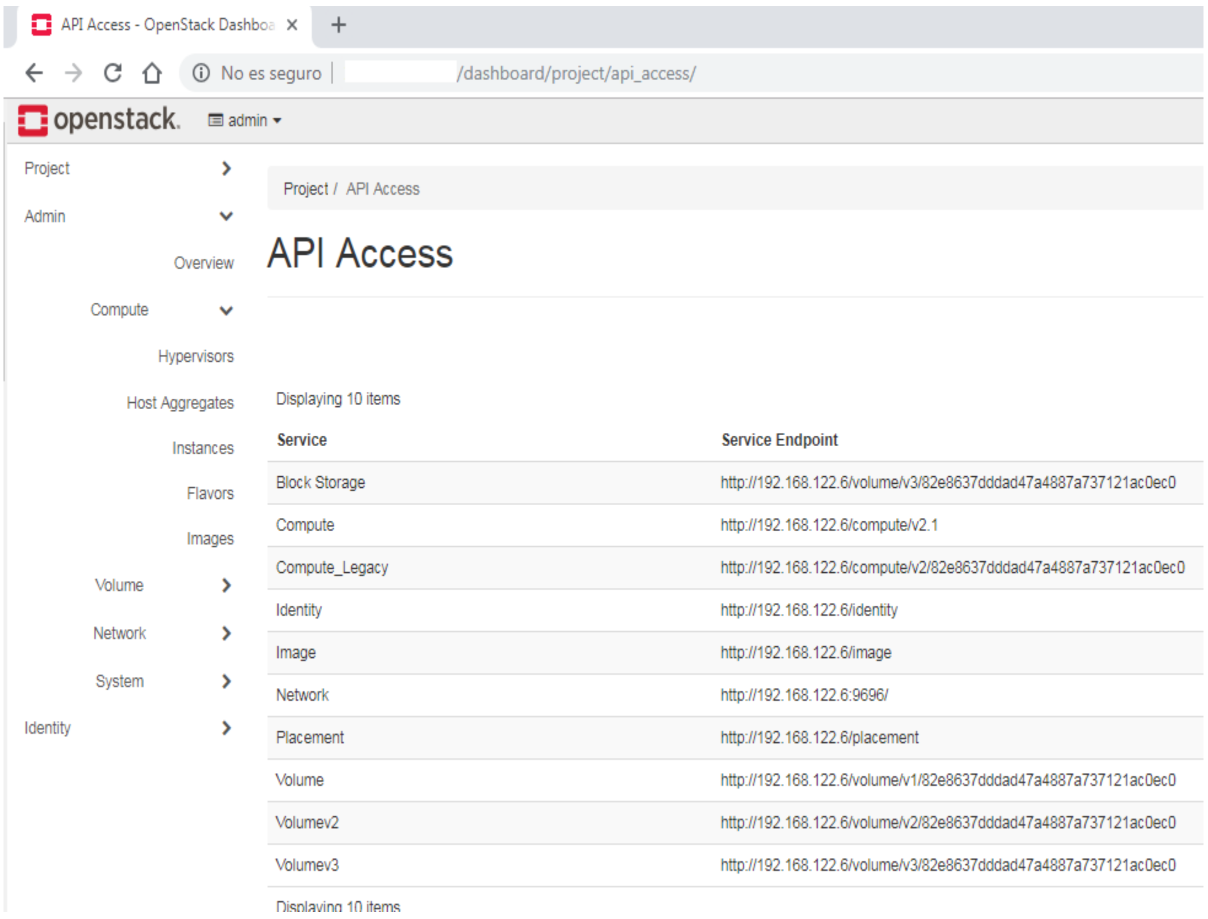
IPs flotantes Asignada 0 de 50

Grupos de seguridad Usada 1 de 10

Reglas del grupo de seguridad Usada 12 de 100

Figure 1. OpenStack Dashboard

Figure 1 shows the central Dashboard used to interact with the OpenStack services. As shown, the main entity in this environment is the Controller node ,where the main cloud services are accessible programmatically through an API (Figure 2). Additionally, further work has been carried out in order to configure private, public networks and routers to support the deployments of pre-packaged virtual images, including the required software for the analytics. The system also includes VM templates that can be instantiated with different flavours (capacity).



API Access

Displaying 10 items

Service	Service Endpoint
Block Storage	http://192.168.122.6/volume/v3/82e8637dddad47a4887a737121ac0ec0
Compute	http://192.168.122.6/compute/v2.1
Compute_Legacy	http://192.168.122.6/compute/v2/82e8637dddad47a4887a737121ac0ec0
Identity	http://192.168.122.6/identity
Image	http://192.168.122.6/image
Network	http://192.168.122.6:9696/
Placement	http://192.168.122.6/placement
Volume	http://192.168.122.6/volume/v1/82e8637dddad47a4887a737121ac0ec0
Volumev2	http://192.168.122.6/volume/v2/82e8637dddad47a4887a737121ac0ec0
Volumev3	http://192.168.122.6/volume/v3/82e8637dddad47a4887a737121ac0ec0

Displaying 10 items

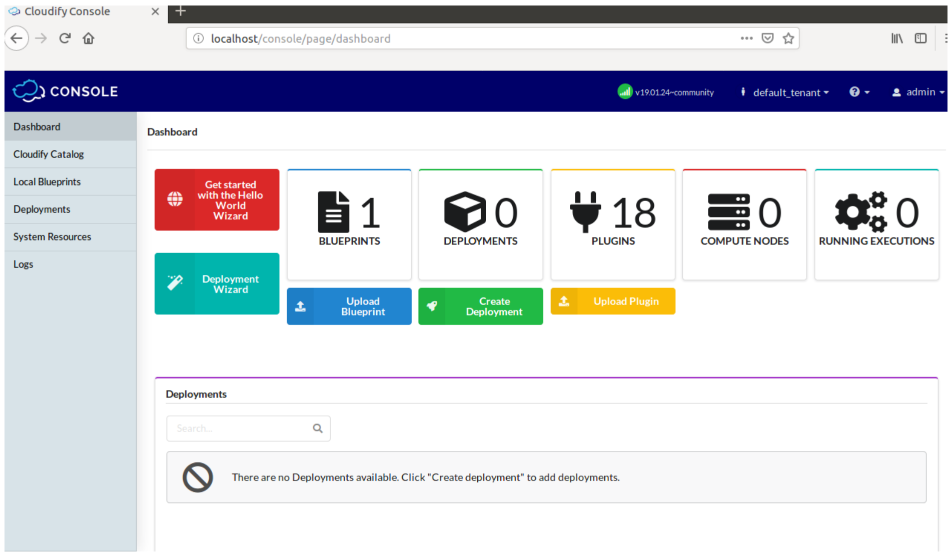
Figure 2. OpenStack API

3.2.1.2 Resource Orchestrator

The brain of the RMO modules is the resource orchestrator, which is based on Cloudify. This technology allows us to describe our deployments using a common specification across different types of providers. In this way, our solution aims to avoid vendor lock-in situation through the abstraction of different providers that could be integrated via plugins to our environment.

In our current solution, OpenStack and Amazon AWS have been the main technologies used for testing our setup. During the testing phase, we performed deployments with various topologies, including different configurations for virtual routers, networks, computational and storage resources.

The entire system can be managed through a dashboard, as shown in Figure 3.



There are no Deployments available. Click "Create deployment" to add deployments.

Figure 3. Cloudify Dashboard.

3.2.2 Deployment Specification

In order to define the logical representation of a cloud-based application, it is needed to specify various elements like the computing nodes to be used, how they relate to one another, or how they need to be deployed and maintained across the lifecycle of the application. The OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) specification offers a well-known standard approach for enhancing interoperability across cloud providers.

The deployment descriptors or blueprints are written in YAML and are packaged together with the configuration files and the installation scripts required for running the application. Once the blueprints have been defined, the Cloudify Dashboard allows us to visualize their topology in a graphical way (Figure 4).

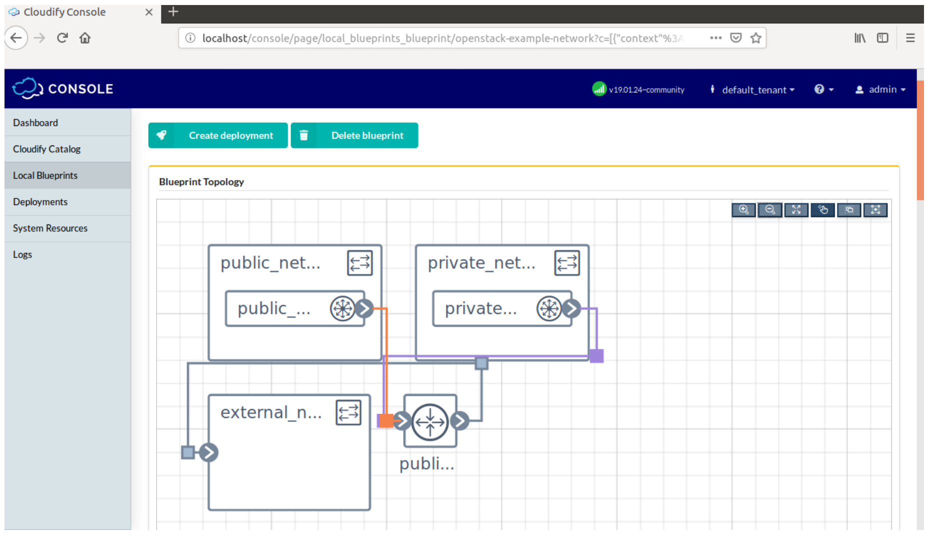
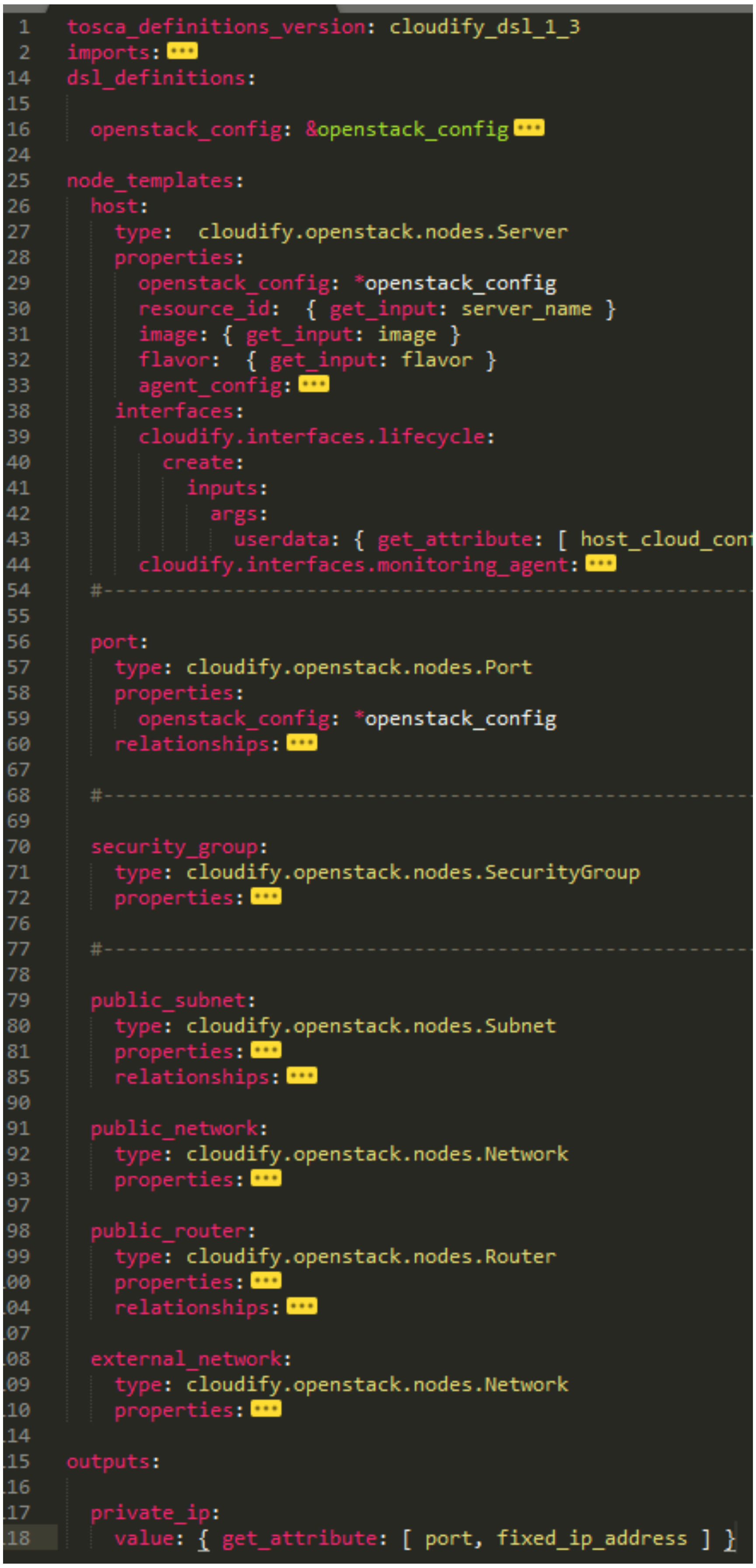


Figure 4. Blueprint Deployment Topology

The steps required to execute a deployment includes the definition of a TOSCA compliant cloud topology description (see Figure 5), and finally the upload of the file to the Resource Orchestrator, using the command that is shown below.

```
cfy blueprints upload <blueprintFileName.yaml> -b <blueprint-id>
```

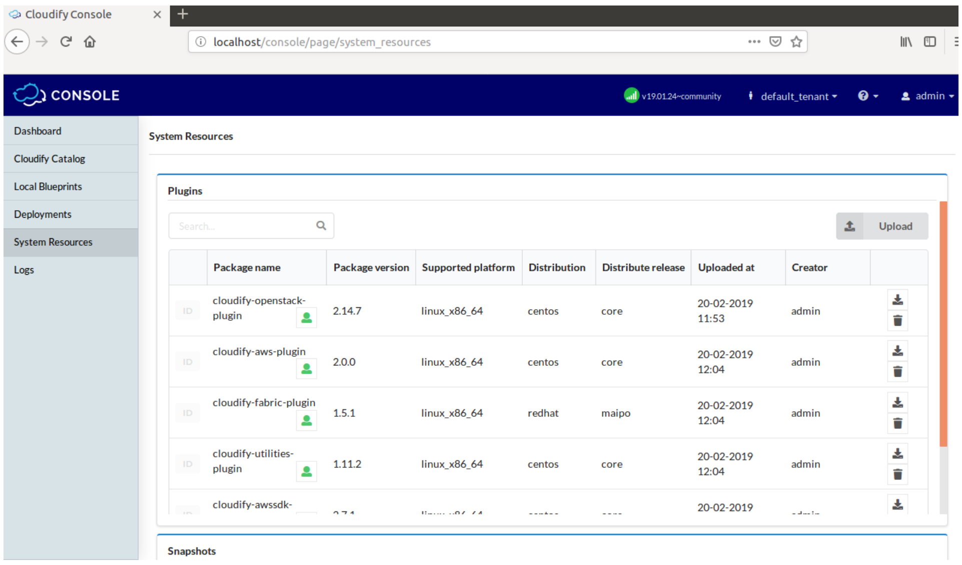


```
tosca_definitions_version: cloudify_dsl_1_3
imports: …
dsl_definitions:

  openstack_config: &openstack_config …

node_templates:
  host:
    type:  cloudify.openstack.nodes.Server
    properties:
      openstack_config: *openstack_config
      resource_id:  { get_input: server_name }
      image: { get_input: image }
      flavor:  { get_input: flavor }
      agent_config: …
    interfaces:
      cloudify.interfaces.lifecycle:
        create:
          inputs:
            args:
              userdata: { get_attribute: [ host_cloud_con
      cloudify.interfaces.monitoring_agent: …
  #-----------------------------------------------------

  port:
    type: cloudify.openstack.nodes.Port
    properties:
      openstack_config: *openstack_config
    relationships: …

  #-----------------------------------------------------

  security_group:
    type: cloudify.openstack.nodes.SecurityGroup
    properties: …

  #-----------------------------------------------------

  public_subnet:
    type: cloudify.openstack.nodes.Subnet
    properties: …
    relationships: …

  public_network:
    type: cloudify.openstack.nodes.Network
    properties: …

  public_router:
    type: cloudify.openstack.nodes.Router
    properties: …
    relationships: …

  external_network:
    type: cloudify.openstack.nodes.Network
    properties: …

outputs:

  private_ip:
    value: { get_attribute: [ port, fixed_ip_address ] }
```

Figure 5. Blueprint description

3.2.3 Adding a new Cloud Service Provider

This action can only be performed by an I-BiDaaS platform operator. When a new Cloud Provider type needs to be on-boarded, a new plugin needs to be configured in the Resource Orchestrator module. The system includes multiple compiled modules (Figure 6) that provide an abstraction for using 3rd party APIs by providing TOSCA types and matching implementation code that can be used via blueprints.



	Package name	Package version	Supported platform	Distribution	Distribute release	Uploaded at	Creator	
ID	cloudify-openstack-plugin	2.14.7	linux_x86_64	centos	core	20-02-2019 11:53	admin	
ID	cloudify-aws-plugin	2.0.0	linux_x86_64	centos	core	20-02-2019 12:04	admin	
ID	cloudify-fabric-plugin	1.5.1	linux_x86_64	redhat	maipo	20-02-2019 12:04	admin	
ID	cloudify-utilities-plugin	1.11.2	linux_x86_64	centos	core	20-02-2019 12:04	admin	
[illegible]	cloudify-awssdk-	[illegible]	[illegible]	[illegible]	[illegible]	20-02-2019	[illegible]	

Figure 6. CSPs Plugins

```
#OpenStack v3.1.0

cfy plugins upload -y openstack_plugin/plugin.yaml http://repository.cloudifysource.org/cloudify/w
agons/cloudify-openstack-plugin/3.1.0/cloudify_openstack_plugin-3.1.0-py27-none-linux_x86_64-
centos-Core.wgn
```

As soon as a new provider type has been uploaded (see sample command above), the concrete stack to be used can be configured. The parameters needed to on-board a new Cloud Service Provider are:

- The *endpoint* where the CSP is accessible.
- The credentials needed to access the provider.

Our current setup includes more than 30 plugins available to be used. Despite this, the main testing environment for I-BiDaaS was our internal Private Cloud based on OpenStack. Additionally, more complex deployments, which include the deployment of Kubernetes clusters, have been tested against Amazon AWS public cloud. The current list of supported providers is OpenStack, Amazon WS, Azure, Google Cloud VMWARE vSpehere and vCloud as well as direct support for Libvirt and Kubernetes.

3.2.4 Monitoring

The availability and performance information in the system is collected thanks to a data collection service included in OpenStack environment called Ceilometer. Ceilometer is a telemetry service that allows to collect meters by polling the infrastructure elements while consumes notifications emitted by other services. The current environment setup for I-BiDaaS includes a Time Series Database service called Gnocchi used to store the collected data.

The summary of metrics collected by the system is the following:

- Memory: Volume of RAM allocated in the instance
- Memory Usage: Volume of RAM used in the instance
- Memory Resident: Volume of RAM used by the instance on the physical machine.
- Memory bandwidth
- CPU Usage
- CPU Time used
- Virtual CPUs allocated
- Disk r/w requests
- Disk r/w bytes
- Disk size
- Disk latency
- Disk capacity
- Disk allocation
- Disk Usage
- Network i/o bytes
- Network i/o packets
- Network i/o packets drop

3.2.5 Achieving Elasticity

Elasticity refers to the capability of the platform to adapt to the computational demand needed for the analytics by provisioning and de-provisioning resources in a dynamic manner.

In the current environment setup, there are two viable options to achieve autoscaling in our deployments. The simplest way to achieve this, is to rely on the application orchestration engine that has been designed for OpenStack, namely OpenStack Heat, as all the subsystems needed are offered under the same software stack. Heat is the orchestration service within OpenStack and it is capable of managing the entire lifecycle of the infrastructure. Heat uses a native template format (HOT-Heat Orchestration Template) while keeps compatibility with AWS CloudFormation format. Through those templates, we can define which resources to scale, which policies need to be applied for the defined resources and when to trigger the scaling process. The scaling process is triggered automatically as soon as an alarm threshold configured in the Ceilometer service is breached.

On the other hand, the previous option relays on the orchestration technology offered by a concrete Cloud provider which will not be applicable for other cloud targets. That said, the most suitable approach for I-BiDaaS to achieve elasticity across a multi-cloud environment is relying on TOSCA templates, that can be triggered from Cloudify. In TOSCA, each logical entity in the application is modelled separately, as well as the relation between the different entities. Through TOSCA specs, we are able to define node entities and associate them with specific actions. The resource orchestrator translates this information in using automated processes, called workflows, that apply the scaling policies that have been defined for each resource type.

3.3 Component Architecture Update

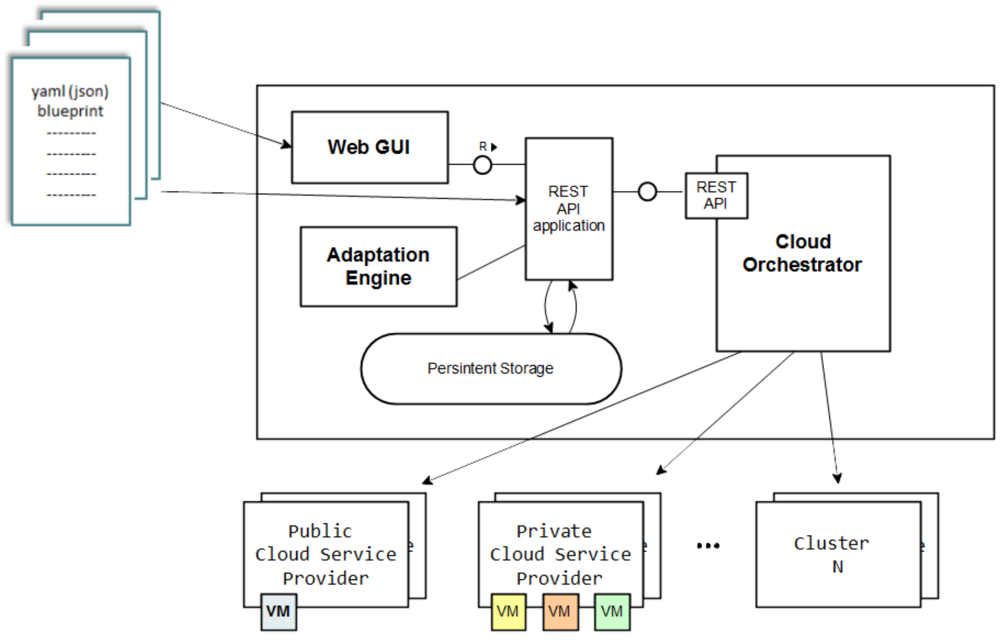


Figure 7. RMO software module architecture

As stated in the previous sections, the Resource Management and Orchestration software systems aim to achieve pre-configured and fully operational deployments across the infrastructure environments, as described in Section 2.

The architecture is depicted in Figure 7, and the software module is composed by the following components:

- A REST API acts as a northbound interface enacting the required calls through the Cloud Orchestrator module to perform the deployment of virtual resources. It consumes deployment templates describing the requirements of the service and the conditions to meet at runtime.
- The Adaptation Engine should guarantee that the service conditions are fulfilled, this is evaluated through the information exposed by the Cloud Orchestrator module, the module will manage the lifecycle of the service applying pre-defined elasticity rules at runtime.
- A SQL-like database stores application states.

The following figure (Figure 8), which updates the one presented in D5.1 [1], maps each of the targeted software modules with one of the technology enablers selected to build our environment.

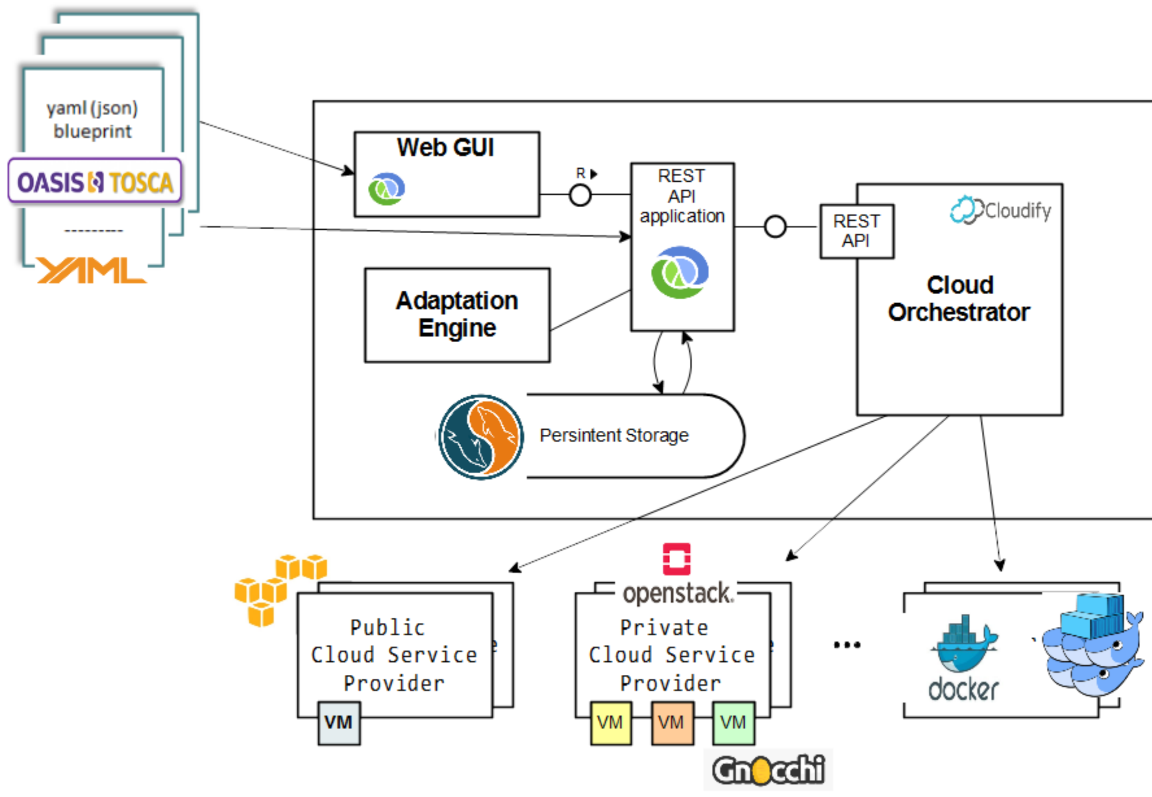


Figure 8. RMO software module technology map

3.4 RoadMap and Features

3.4.1 Tasks Roadmap until M24

Month	Summary of Main Actions
M13-14	- Maintenance of the cloud environment supporting the MVP - 1st I-BiDaaS Prototype Time Plan
M15-16	- Reconfiguration of the environment towards the 1st I-BiDaaS prototype. - Build and configure the selected resource management and orchestration stacks.
M17-M18	- Executing and recording demo scenarios for the review. - Infrastructure support for the 1st Review held in Luxemburg
M19-M20	- Integration between the Resource Management and Orchestration module with our OpenStack Cloud setup. - Collect feedback across technical WPs needs. - Reconfigure our private cloud for working in multi-node mode.
M21-22	- First D5.3 ToC - Extend computational capacity - MVP time plan
M23-24	- 2nd I-BiDaaS Prototype Time Plan - Deliverable Reporting and Quality Review Process

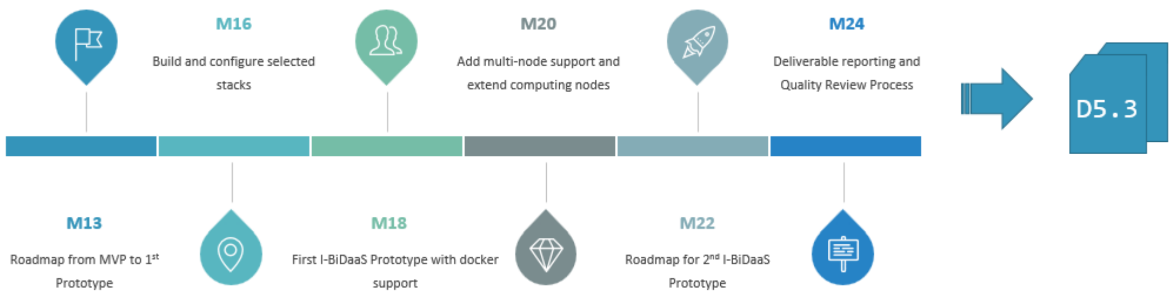

M16
Build and configure selected stacks
M20
Add multi-node support and extend computing nodes
M24
Deliverable reporting and Quality Review Process
D5.3
M13
Roadmap from MVP to 1st Prototype
M18
First I-BiDaaS Prototype with docker support
M22
Roadmap for 2nd I-BiDaaS Prototype

Figure 9. D5.3 Time plan & RoadMap

3.4.2 Code Repository

The source code of the software modules is going to be stored in the source control repository system of the project. Further information can be found in D5.4. Big-Data-as-a-Self-Service Test and Integration Report [5].

4 Runtime Environment

4.1 Introduction

As stated in D5.1 [1], the runtime environment provides the execution environment for data processing applications as well as the interface with the data storage.

Based on the analysis of the user requirements (reported in D1.3 [6]), the I-BiDaaS platform "*should support diversified, analytic processing, machine learning and decision support techniques to support multiple stages of analysis (FR4)*".

This implies the ability to process huge volumes of data and therefore, programming productivity[1] is crucial. To this end, distributed processing capability is required so that different processing algorithms can be implemented and executed in a distributed way.

This, however, poses significant programming challenges inherent to concurrent and distributed programming, e.g., dealing with threading, messaging, data partitioning and transfer. Additional challenges relate to the complexity of the underlying distributed computing infrastructure, which enables the execution of multiple tasks in distributed resources.

Thus, the runtime environment should provide IT developers with a simple means for writing both parallel and distributed applications that can process big amounts of data, in order to improve programming productivity without sacrificing performance. At the same time, it should provide transparent interoperability with the resource management and orchestration component, thus freeing IT developers from having to deal with the details of the specific resource infrastructure.

Furthermore, as non-relational databases are the most common solution when dealing with the huge data sets and massive query workloads relevant to big data, the runtime environment should provide a set of tools and interfaces, which facilitate programmers with an efficient and easy interaction with non-relational data base technologies.

4.2 Enhanced Capabilities

The advanced machine learning module comprises of a set of efficient and scalable machine learning algorithms. Given a particular problem of relatively large dimension (meaning that a sequential algorithm is not suitable for solving it due to its size), an optimal number of workers can be determined for each algorithm that can be used. By increasing the number of workers, the execution time of the algorithm decreases until reaching the optimal number of workers. This can be achieved, as the algorithms are developed relying on the COMPSs framework, within the 1st I-BiDaaS version COMPs and Hecuba workers use docker containers for its execution.

4.3 COMPs and Hecuba Runtime

As described by Lordan et al. [7], COMPSs applications are implemented in a sequential way, and without APIs that deal with the details of the distributed infrastructure or with duties of parallelization and distribution (synchronizations, data transfers, …). This is very important to offer a unique and simple programming interface to create applications. This means, on the one hand, that the application will not be based on a specific API to express the interaction with the

[1] https://en.wikipedia.org/wiki/Programming_productivity

infrastructure, thus avoiding vendor lock-in and lack of portability of applications. On the other hand, we are adopting sequential programming as the main programming paradigm, which is the easiest paradigm to be offered to end users, therefore achieving an easy way for users to program applications. Users do not need to think of how precisely their program is going to be run in the distributed infrastructure, because the COMPSs runtime will take care of the actual execution of the application on the resources available. Instead, users only need to focus on their specific domain of knowledge to create a new program that will be able to run in the cloud or any other distributed infrastructure. Another key aspect of providing a cloud-unaware programming model is that programs can be developed once and run in multiple clouds without changing the implementation. This is very important when portability between clouds must be achieved. In COMPSs, the programmer is freed from having to deal with the specific cloud details, because COMPSs runtime will be in charge of it. The runtime follows a plug-in approach to deal with several cloud frameworks, hiding this burden to the end user and enabling interoperability.

Regarding COMPSs capacity to work at large scale, as a mature programming model, with more than 10 years invested in its development and testing, many different testing results have been obtained during all its development years. The most remarkable are in Amelan et al. [8], where more than 1000 execution cores are used for testing quite complex workflow structures; and in the BioExcel Project [9], where an execution using 800 nodes of MareNostrum IV (using 38,400 cores) was performed for Molecular Dynamics simulations. These second set of tests are still pending to be published.

While COMPSs takes care of the distribution of the computation, Hecuba is in charge to optimize the data management. Hecuba uses the NoSQL open-source database Apache Cassandra to store the data allow to delegate on the database the management of the global view of the data, avoiding explicit synchronisation points and maximising the parallelism degree. Hecuba allows users to access the data as regular python objects stored in memory. The only requirement is to define previously the class that supports each data structure, specifying the data types. Hecuba allows great code simplification and it guarantees performance improvement over alternative storage solutions such as file over high-performance parallel file systems [10].

4.4 I-BiDaaS Prototype Integration

4.4.1 Integration for the 1st I-BiDaaS Prototype

In order to properly demonstrate the services and functionalities of the I-BiDaaS integrated platform, the consortium decided to proceed - on top of the developments of the tailor-made I-BiDaaS use cases described in D2.1 - to a complementary implementation of a generic use case; the main purpose is to demonstrate the functionalities of the different envisioned modes of the I-BiDaaS platform, namely (i) its *Self-service mode*, that allows in-house personnel with domain-specific knowledge and some insights about data analysis (i.e., non-experts) to easily construct Big Data pipelines in a user-friendly way, by selecting a data analytics algorithms from a pre-defined list; (ii) the *Expert mode*, that allows experts (i.e., developers) to upload their custom data analytics source code, based on the available I-BiDaaS highly reusable templates; and (iii) the *Co-Develop mode*, that corresponds to end-to-end solutions to specific industrial domains, developed by the I-BiDaaS team (the I-BiDaaS use cases).

In the generic use case, users can either initiate existing projects (with already implemented algorithms, e.g., K-means) or create custom ones (by uploading and running their one code - expert mode). Then they can create and execute their experiment(s) based on their data and preferences; the process is depicted in Figure 10.

The basic services and functionalities of the 1st version of the I-BiDaaS platform, as delivered by M18 via the Generic Use case implementation, are summarized below:

- The Self-service mode allows industry in-house personnel which have domain knowledge and some insights about data analysis (non-experts) to easily construct Big Data pipelines in a user-friendly way, selecting a pre-defined data analytics algorithm from an available list.
- The Expert mode allows experts (developers) to upload their own data analytics code based on the available I-BiDaaS highly reusable templates.
- The Co-Develop mode corresponds to an end-to-end solution for a given industry project developed by the I-BiDaaS team (the I-BiDaaS use cases).

Projects can also be divided in two different categories: Batch processing projects and stream processing projects. Focusing explicitly on batch processing, the system architecture is shown in Figure 10. The cluster that runs the batch processing jobs is based on docker swarm [11]. The swarm consists of a manager node and a set of worker nodes. The orchestrator assigns dynamically a set of workers from the set of available nodes in docker swarm, based on the user's preferences and the set-up selected through the UI. These workers exploit the Cassandra DB and the shared FS in order to complete the requested job.

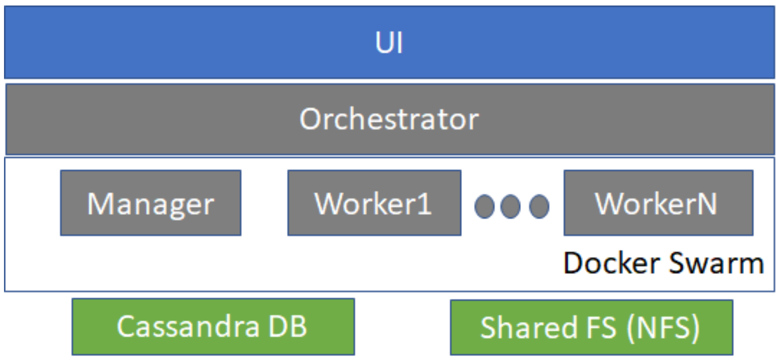


Figure 10. Batch processing architecture: The orchestrator and the docker swarm

On the other hand, Figure 11 demonstrates the case of stream processing; here, the analysis is carried out through the APAMA Stream Processing Engine in collaboration with the UM tool, rather than via docker swarm.

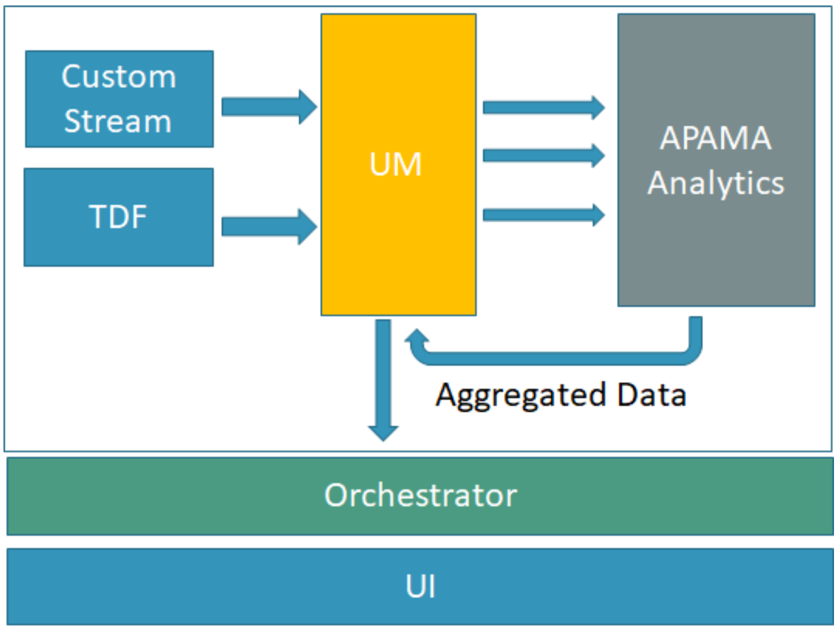


Figure 11. Streaming processing architecture: The orchestrator and APAMA analytics

More information about the architecture and functionalities of the 1st I-BiDaaS Prototype can be found in Deliverable D5.4 [5].

4.4.2 Time plan for the 2nd I-BiDaaS Prototype

The 1st I-BiDaaS prototype was delivered in M18. Then a revised version (mainly with bug fixes) has been developed and presented in the 1st I-BiDaaS review in M20.

The period between M21 and M24, when this deliverable is to be submitted, the consortium has focused on the design of the remaining Use Cases that will be integrated in the 2nd I-BiDaaS Prototype.

Future work, starting from M25, will expand the integration of the I-BiDaaS platform with respect to the 1st integrated version towards the final implementation including all the tools and technologies by FORTH, BSC, ATOS and UNSPMF, and it will include also the remaining CAIXA, CRF and TID's use cases experiments according to a revised experimental protocol as lead by UNIMAN.

Some important upcoming milestones include the integration of the RMO module (M26), and the finalization of the design of the remaining Use Cases (M26).

Moreover, as long as the various use cases that will be implemented within the I-BiDaaS platform become well-defined, a complete list of E2E tests will be prepared and utilized for the validation of I-BiDaaS final platform integration.

The time plan is depicted in the following figure:

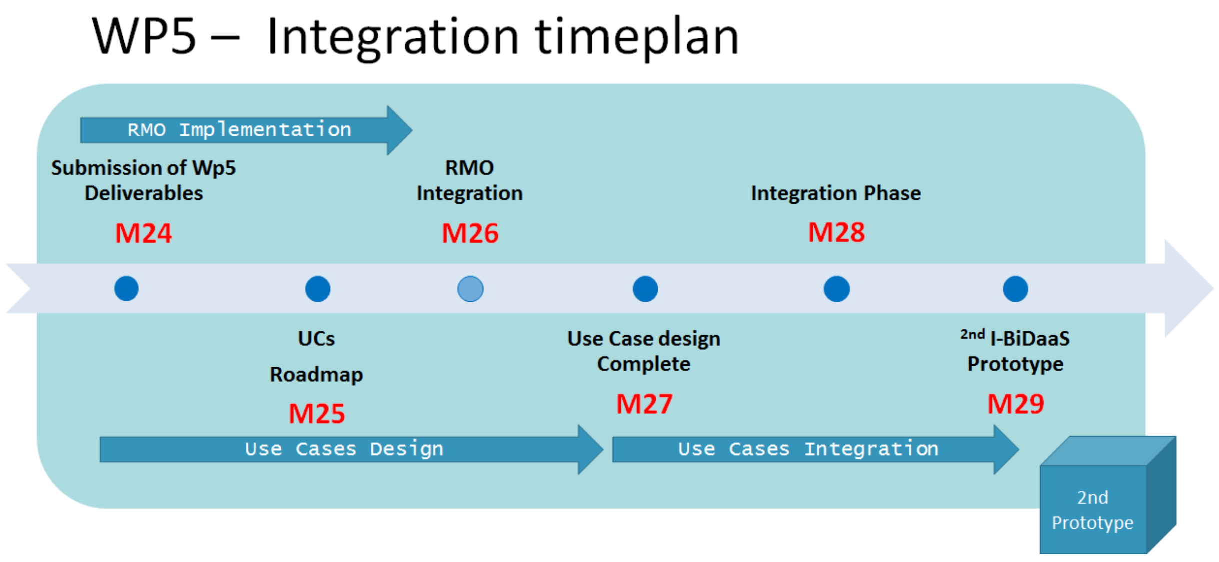


Figure 12. Integration Time Plan Overview

5 Conclusions

According to the DoA, for WP5 the "*the primary objective is to provide the distributed large-scale framework that permits powerful and scalable Data processing on top of heterogeneous and federated infrastructures*".

The list of actions performed to overcome the WP challenges are summarized in the table below:

Table 2: Challenges to Overcome

Challenge	Summary of actions to M24
Management of diverse infrastructure resources for Data Analytics (including cloud and GPU resources)	· Maintenance and extension of the private cloud infrastructure. · Maintenance of GPU resource cluster. (see Section 2)
Orchestration of infrastructure resource management across diverse resource providers	· Implementation and testing of the Resource Management and orchestration software modules and their interactions. (see Section 3)
Seamless integration and exploitation of infrastructure elasticity capabilities by the runtime environments	· The integration processes and tool chain selected toward the 1st I-BiDaaS prototype have been fulfilled. · Specification of the runtime environment which provides the execution environment for data processing applications as well as interface with the data storage pools. (see Section 4)

The work carried out within this period allows us to have an operational infrastructure environment supporting the realisation of the I-BiDaaS 1st prototype as well as the deployment of the software modules which are part of the I-BiDaaS solution.

While the first period has been focused on the provisioning and management of the infrastructure, the next steps have been focused on the orchestration of computational resources for Big Data applications and I-BiDaaS use cases. The 1st I-BiDaaS prototype offers docker support for the deployments; this process will be enhanced after the first release of our solution in order to provide supports across different Cloud Service Providers thanks to the integration with the RMO software module.

6 References

[1] I-BiDaaS Consortium. D5.1: Federated Resource Management for Data Analytics v1, 2018, Project Report.

[2] I-BiDaaS Consortium. D5.2: Big-Data-as-a-Self-Service Test and Integration Report v1, 2018, Project Report.

[3] D-Alix data center: http://www.d-alix.com/

[4] I-BiDaaS Consortium. D4.2: Distributed event-processing engine, 2019, Project Report.

[5] I-BiDaaS Consortium. D5.4: Big-Data-as-a-Self-Service Test and Integration Report v2, 2019, Project Report.

[6] I-BiDaaS Consortium. D1.3: Positioning of I-BiDaaS, 2018, Project Report.

[7] Lordan, F., Tejedor, E., Ejarque, J., Rafanell, R., Álvarez, J., Marozzo, F., ... & Badia, R. M. (2014). Servicess: An interoperable programming framework for the cloud. *Journal of grid computing*, *12*(1), 67-91.

[8] Amela, R., Ramon-Cortes, C., Ejarque, J., Conejero, J., & Badia, R. M. (2018). Executing linear algebra kernels in heterogeneous distributed infrastructures with PyCOMPSs. *Oil & Gas Science and Technology–Revue d'IFP Energies nouvelles*, *73*, 47.

[9] BioExcel Center of E (Santamaria, 2019)xcellence (Horizon 2020 Framework program) under contracts 823830, and 675728.

[10] Santamaria, P., Oden, L., Gil, E., Becerra, Y., Sirvent, R., Glock, P., & Torres, J. (2019, June). Evaluating the Benefits of Key-Value Databases for Scientific Applications. In International Conference on Computational Science (pp. 412-426). Springer, Cham.

[11] Docker Swarm: https://docs.docker.com/engine/swarm/