Infrastructure Operation Report

June 19, 2017

Deliverable Code: D8.2

Version: 1.0 – Final

Dissemination level: Public

D8.2 is in update of the Infrastructure Operation Report document that was first delivered in M18 of the project. D8.2 in particular, focuses in months 19-24 providing an update of the activities that took place during this period and the progress towards the goals of WP8 and the cloud infrastructure provisioning activities.

H2020-EINFRA-2014-2015 / H2020-EINFRA-2014-2
Topic: EINFRA-1-2014
Managing, preserving and computing with big research data
Research & Innovation action
Grant Agreement 654021

Document Description

D8.2 – Infrastructure Operation Report

WP8 – Operation and Maintenance	
WP participating organizations: ARC, GRNET	
Contractual Delivery Date: 5/2017	**Actual Delivery Date:** 6/2017
Nature: Report	**Version:** 1.0
Public Deliverable	

Preparation slip

	Name	Organization	Date
From	Evangelos Floros, Stavros Sachtouris, Thodoris Sotiropoulos Byron Georgantopoulos	GRNET	16/6/2017
Edited by	Evangelos Floros	GRNET	16/6/2017
Reviewed by	Mark Greenwood, Angus Roberts	USFD	16/6/2017
Approved by	Androniki Pavlidou	ARC	19/6/2017
For delivery	Mike Chatzopoulos	ARC	19/6/2017

Document change record

Issue	Item	Reason for Change	Author	Organization
V0.1	Draft version	ToC definition. Contributions in Section 2 and 3	Vangelis Floros	GRNET
V0.2	Draft	Contribution to Annexes	Stavros Sachtouris, Thodoris Sotiropoulos	GRNET

V0.3	Draft	Service downtime information. Overall document completion.	Vangelis Floros	GRNET
V0.4	Draft for review	Comments from internal WP8 review	Byron Georgantopoulos, Vangelis Floros	GRNET
V0.5	Pre-release	Edits after external review	Vangelis Floros	GRNET
V1.0	Final	Release edits	Vangelis Floros	GRNET

Table of Contents

Table of Figures

Disclaimer

This document contains description of the OpenMinTeD project findings, work and products. Certain parts of it might be under partner Intellectual Property Right (IPR) rules so, prior to using its content please contact the consortium head for approval.

In case you believe that this document harms in any way IPR held by you as a person or as a representative of an entity, please do notify us immediately.

The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any sort of responsibility that might occur as a result of using its content.

This publication has been produced with the assistance of the European Union. The content of this publication is the sole responsibility of the OpenMinTeD consortium and can in no way be taken to reflect the views of the European Union.

The European Union is established in accordance with the Treaty on European Union (Maastricht). There are currently 28 Member States of the Union. It is based on the European Communities and the member states cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice and the Court of Auditors. (http://europa.eu.int/)

OpenMinTeD is a project funded by the European Union (Grant Agreement No 654021).

Acronyms

IAAS	Infrastructure as a Service
GRNET	Greek Research and Technology Network
VM	Virtual Machine
API	Application Programming Interface
CLI	Command Line Interface
UI	User Interface
REST	REpresentational State Transfer
CPU	Central Processing Unit
SSH	Secure Shell
GUI	Graphical User Interface
EC	European Commission
CPU	Central Processing Unit
RAM	Random Access Memory
OS	Operating System
IP	Internet Protocol
AAI	Authentication and Authorization Infrastructure
IdP	Identity Provider
DRMAA	Distributed Resource Management Application API
GUI	Graphical User Interface
NFS	Network File System
OIDC	OpenID Connect
SAML	Security Assertion Markup Language

Publishable Summary

This document reports on the activities carried by WP8 in project months 19-24. During this period, the activity focused on two aspects of work: the continuing support and provisioning of a stable cloud infrastructure and expanding the capabilities of this infrastructure responding to project-specific requirements. For what concerns resource provisioning and operations, the task continued with no significant issues or problems. Cloud resource consumption has remained at the same levels since no service has yet been put into production from the rest of the project activities. A slight increase in computing and storage resources have been noted mostly due to the increasing test and development activities. During this period, the ~okeanos middleware was upgraded an event which introduced a short downtime to the service. This downtime though had only a minor affect on the technical work of OpenMinTeD that resides on the cloud.

Development-wise, WP8 continued with the integration activities of Galaxy with ~okeanos cloud. The work on the Pithos+ plugin concluded and currently we are able to use Pithos+ as the object storage of Galaxy for storing data and workflow execution datasets. A major challenge was the development of the workflow execution backend that would allow Galaxy to seamlessly offload the workload execution activities on a pool of computing resources. For this purpose, WP8 investigated possible workload management alternatives concluding with the selection of Apache Mesos/Chronos framework as the best option for managing the execution of Docker containers as part of Galaxy workflows. The necessary plugin logic was developed which allows Galaxy to submit jobs to a Mesos/Chronos which in turn exploits a dynamically managed pool of ~okeanos VMs to execute applications. Both the Pithos+ plugin as well as the Mesos/Chronos plugin have been pushed to the upstream Galaxy platform development and will be officially part of the distribution in the coming months. This way WP8 and OpenMinTeD have actively participated in the development and expansion of an open source initiative like Galaxy, contributing results that are of benefit to other applications and scientific context.

Finally, a set of Ansible playbooks have been developed that allow fast deployment of this multistage workload execution infrastructure facilitating the setup of multiple similar environments in OpenMinTeD for the various required development and testing purposes.

1. Introduction

WP8 “Maintenance and Operations” is the activity responsible for the establishment and provision of cloud computing infrastructure and the relevant supported services, in order to satisfy the computing demands of OpenMinTeD. WP8 is a key activity of the project since it provides the computing platform where OpenMinTeD’s architecture is materialized and as such this is where the various services live, applications run and data resides.

During the first period of work (M1-M18) the activity mainly focused on establishing the necessary base infrastructure and setting up the respective user support procedures. A number of investigative activities were carried out on a technical level in an effort to remain proactive and be ready to cover potential requirements of the project. The finalization of the project architecture and the selection of Galaxy platform as the core workflow execution engine was a major milestone of the project, and in particular for WP8, since it helped put the whole role of the cloud infrastructure in context and demystified the technical challenges that this infrastructure would be called upon to fulfill.

This document reports on the activities carried out by WP8 in project months 19-24. During this period, the activity focused on two aspects of work: the continuing support and provisioning of a stable cloud infrastructure and expanding the capabilities of this infrastructure responding to project-specific requirements. By its nature, the work of WP8 falls within a DevOps modus operandi: Development activities focus on how to exploit existing APIs in order to enable a functionality that was not available before. Operation activities exploit this development effort, putting the results into action and making sure that they operate in a level of service quality acceptable for hosting production services. This twofold nature is reflected in the structure of this document. In particular Section 2 recaps the operational aspect of WP8 whereas Section 3 details the development activities that took place during the previous period. Section 4 concludes with a short summary of the achievements and the planned work for coming months. The document is complimented by three Annexes (A-C) that offer a more technical insight of the development work that was carried out during this period.

2. *Project Resource Provisioning*

2.1 Overall Resource Utilization

Two ~okeanos Projects have been created by GRNET. The first project registered with the name **openminted.grnet.gr** is dedicated to hosting the OpenMinTeD platform and all the end-user services that are being developed in the context of the project. A second project named **aai.openminted.grnet.gr** allocates resources dedicated for the AAI services of OpenMinTeD.

2.1.1 OpenMinTeD generic ~okeanos project

The **openminted.grnet.gr** ~okeanos project provides cloud resources for generic usage in the project. So far, the main resource consumption was triggered by development and integration tests related to Galaxy and its ecosystem of services (job scheduler, multiple test Galaxy installations, test Registry installations etc). At the end of M24 a snapshot of resource consumption was as follows:

RESOURCES

	Max per member	*Total*	*Usage*
File Storage Space	3.00 TB	10.0 TB	0% (1.35 GB)
Hard Disk Storage	2.00 TB	16.0 TB	14% (2.28 TB)
CPUs	200	1600	8% (134)
RAM	800.0 GB	6.00 TB	2% (181.0 GB)
VMs	25	500	5% (27)
Private Networks	5	20	15% (3)
Public IPs	16	64	51% (33)

Figure 1 - OpenMinTeD project resource utilization

As can be seen the resource utilization is still relatively low since no production deployment of any of the major services (Registry, Workflow) has taken place yet. Also, none of the OpenMinTeD endorsed corpora have been transferred into ~okeanos and obviously, the platform has not yet opened to end users who are expected to start moving and processing large amounts of data on the cloud. This situation is expected to change considerably in the coming months in which period the first production versions of the project services will be deployed and offered publicly. In addition, the Open Call for applications is expected to place significant demands, especially processing capabilities (and as a result requirement for large number of “worker” VMs).

2.1.2 OpenMinTeD AAI project

The **aai.openminted.grnet.gr** ~okeanos project hosts the resources required for the OpenMinTeD AAI federation IdP Proxy service. The project currently operates at full capacity having allocated all resources made available for this purpose. So far, these resources are sufficient. In case additional capabilities are required to be deployed there is a flexibility to expand the available capacity in order to satisfy the increased demand. In particular, as of the writing of this report the resource consumption was as follows:

RESOURCES

	Max per member	Total	Usage
File Storage Space	0 bytes	0 bytes	--
Hard Disk Storage	160.0 GB	160.0 GB	100% (160.0 GB)
CPUs	32	32	100% (32)
RAM	32.0 GB	32.0 GB	87% (28.0 GB)
VMs	16	16	68% (11)
Private Networks	2	2	50% (1)
Public IPs	16	16	68% (11)

Figure 2 - Resource allocation for the aai.openminted.grnet.gr project

2.2 Virtual Machine profiles

The following table summarizes the resources utilized by OpenMinTeD projects as of the writing of this report. We've taken into account stable services running either on production basis or permanent pre-production services currently used for experimentation and development.

VM #	Role	CPU cores	Main Memory (GB)	Volume Storage (GB)
1	Nginx proxy server	4	8	40
2	Nexus Repository	4	8	40
3	Build server (Jenkins)	4	8	40
4	WebAnno server	4	4	20
5	WP9 Demonstrator 1	8	8	60
6	WP9 Demonstrator 2	8	8	60
7	TDM linguistic pipeline	8	8	60
8	Virtuoso triple store	8	8	60
9	Galaxy development (INRA)	2	6	40
10	Galaxy development (GRNET)	2	4	20
11	OMTD Admin	2	1	20
12	OMTD NFS	8	8	60
13	OMTD Galaxy	8	8	60
14	OMTD Mesos	8	8	40
15	OMTD Chronos	4	4	40
16	HTTP Load balancer (nginx)	4	2	5

17	HTTP Load balancer (nginx/standby)	4	2	5
18	SAML (SimpleSAMLphp) Account Registry (COmanage)	4	4	10
19	SAML (SimpleSAMLphp) Account Registry (COmanage)	4	4	10
20	Cache (memcached)	1	1	5
21	Cache (memcached)	1	1	5
22	DB (postgresql/master)	2	2	20
23	DB (postgresql/hot standby)	2	2	20
24	OIDC (MITREid Connect)	4	4	10
25	OIDC (MITREid Connect)	4	4	10
26	Backup (barman)	2	2	60
27	Development	4	8	60
28	Development	4	8	40
29	Registry Production	4	8	60
30	Ontology server	2	4	40
	Total Resources consumed on by OpenMinTeD	**128 (CPU cores)**	**158 GB**	**1020 GB**

As it can be seen a total of 30 VMs are operating constantly consuming a total of 128~CPU cores, 158~GB of main memory and 1~TB of persistent storage.

2.3 Service downtimes

The ~okeanos service experienced a short downtime for a few hours on May 23 between 08:00 and 12:00 CEST due to scheduled service maintenance. The maintenance included an upgrade of core middleware as well as upgrade of the OS (Debian Linux) on hosting servers. During the downtime, all running VMs had to be shutdown. All VMs were brought back online without user intervention once the maintenance process completed and the ~okeanos core services became available again. All technical members of OpenMinTeD were notified with an email on 11/5 and again on 22/5 in order to make aware aware of the incident and prepare any corrective actions in case there were services running on VMs that would require graceful shutdown or for services that weren't configured to start unattended after a VM reboot. The WP8 support team remained on standby during the whole process to take care of any potential issues that would arise during the downtime. The process completed without any major problem and all project VMs came back online unaffected by the downtime.

3. ~okeanos as workload execution and storage backend

The Galaxy platform has been selected by the project to form the core of the workflow description and execution service. Out of the box Galaxy can support a large range of execution backends including: local machines, computing clusters managed by DRMAA [6] compliant resource managers, local docker engine and docker swarm enabled pool of VMs. Datasets can be stored on local storage, shared storage volumes or cloud storage services such as Amazon S3 or OpenStack Swift [15] compatible services. The goal set in the context of WP8 is to seamlessly integrate Galaxy with ~okeanos VM management (Cyclades [3]) and object storage service (Pithos+ [5]) in order to allow Galaxy to fully exploit the capabilities of the latter regarding workload execution and storage management.

3.1 Pithos+ integration with Galaxy

Effort on Pithos+ integration with Galaxy started during the previous months right after the selection of Galaxy as the a workflow engine. The goal was to allow Galaxy users to be able to upload data to Pithos+ storage, access them during workflow execution and store their intermediate and final results. In order to enable the integration, a Pithos+ specific plugin had to be developed expanding in parallel the core Galaxy code in order to be able to use it. More detailed technical information about the implementation of the plugin is provided in Annex A: Pithos+ as Galaxy storage backend. The usage of Pithos+ as a storage backend is completely transparent to the end user since the integration is done on the Galaxy hosting layer by the OpenMinTeD service provider. The plugin code has been pushed upstream to the main Galaxy development team and is expected to be included in one of the future software releases.

3.2 Using Chronos/Mesos as Galaxy workload manager

A major requirement coming from the project is the ability to use ~okeanos computing capabilities in order to execute TDM workflows. After the selection of Galaxy as the workflow engine, focus was placed on how to use clusters of VMs running on ~okeanos in order to offload the necessary activities. For this purpose, multiple options were evaluated and have been reported in D8.1 [13]. Core requirement of these activities was that OpenMinTeD application will be packaged and distributed as Docker [12] containers, therefore the backend should provide docker engine capabilities. One additional requirement is the ability to dynamically alter the number of available computing resources (Virtual Machines) in order to be able to handle fluctuating workflow execution demands, either in an autonomous or semi-autonomous manner (in which case human intervention would be required).

The final architecture we have designed and implemented relies on the Mesos/Chronos framework for resource scheduling of VMs. Mesos [9] is a cloud resource manager, that integrates with IaaS clouds and is responsible for negotiating VM allocation for specific tasks. Chronos [10] is a resource scheduler

that accepts workload definitions (jobs), negotiates resources with Mesos and keeps track of job execution. This combination of tools was selected among others because they integrate seamlessly with ~okeanos, they support docker containers out of the box and are backed by Apache Software Foundation ensuring strong community commitment and long term sustainability.

The final workload execution architecture is depicted in Figure 3.

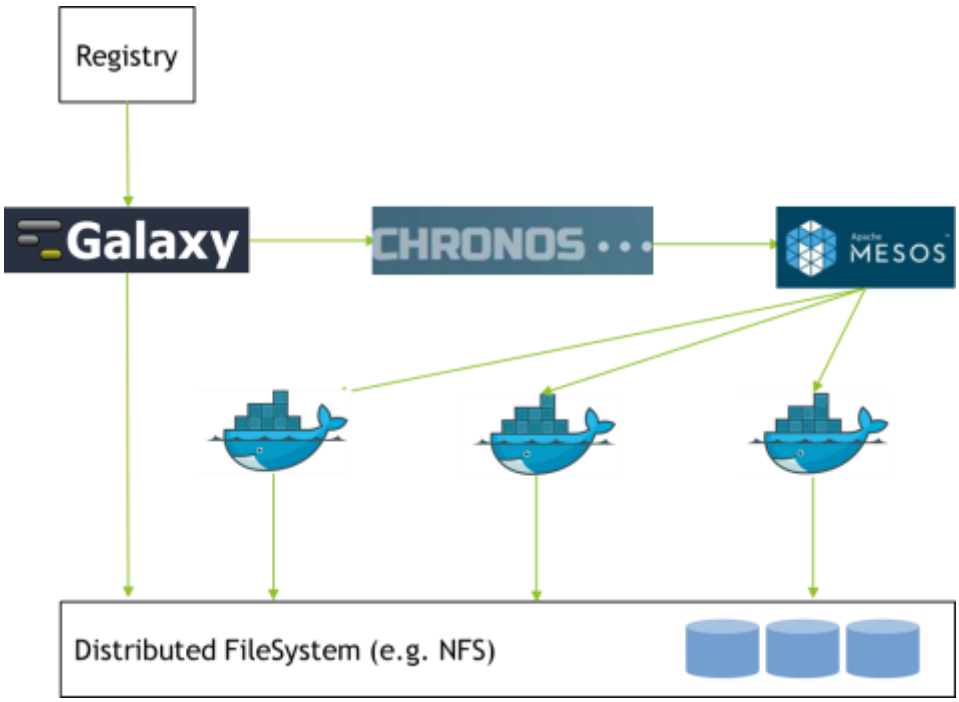

Figure 3 - Workflow execution architecture

According to this architecture Galaxy communicates with Chronos passing information and requirements about workflow execution. The applications (named "tools" in the Galaxy nomenclature) to be executed as part of the workflows are provided in the form of Docker containers. Chronos in turn negotiates with Mesos requesting the necessary resources (CPU cores, memory, storage) required for running the workflow. Mesos manages a pool of ~okeanos VMs which are configured with the Docker engine and thus have the ability to run containers. The pool of VMs share a common file system with the VM hosting the Galaxy service. This is required in order to pass input and output data as well as the applications themselves. For the time being this shared file system is based on NFS. For production environments, with higher demands concerning performance and availability, other more advanced solutions could be used such as OrangeFS, BeeFS, ClusterFS etc.

3.3 Service deployment automation

The deployment of the architecture depicted in Figure 3 is a tedious process requiring multiple steps. Automation of the process is crucial in order to be able to replicate it many times either for setting up development environments or for the final production service deployments. For this purpose, we have

developed a set of Ansible [16] playbooks that take care of the necessary steps in order to deploy the full OpenMinTeD workload execution backend including a Galaxy instance, a Chronos instance, a Mesos instance and multiple Docker engine-enabled VMs all configured under a shared file system. The playbook is available from Github: https://github.com/openminted/omtd-stack-setup. It requires that the user holds an account in ~okeanos service and enough quota to deploy the necessary number of VMs and allocate storage and other computing resources necessary for operation (public IP address, main memory etc). More information about the playbook and the setup procedure can be found in Annex C: Automated execution backend setup.

4. Conclusions and planned work

During months M19-M24 the WP8 team focused mostly on middleware expansion and integration activities. The team was presented with a clear challenge for the cloud infrastructure to efficiently support workload execution submitted by Galaxy and also to satisfy the storage requirements of TDM applications. The culmination of this work was the integration of Pithos+ with Galaxy on one side, for what concerns data management requirements, and the design, development and implementation of a workload management architecture based on the Mesos/Chronos framework. Both solutions have been put into pre-production as integral part of the services currently tested by OpenMinTeD. This work also gave a good incentive for the WP8 team to closely collaborate with Galaxy open source development community and contribute to the developments of the software which is gaining increased popularity in the context of Life Sciences and TDM scientific communities.

In the coming months, we will continue working on the expansion and improvement of the OpenMinTeD workload backend functionality. As the OpenMinTeD platform is expected to open to the broader TDM community more end-users are expected to come increasing the load and utilization of the services. WP8 will also work closely with WP6 in the context of the Monitoring and Accounting task. The requirement on the cloud infrastructure is to facilitate monitoring of resource consumption by Docker containers running as part of Galaxy workflows. As workflows are submitted from the OpenMinTeD portal the infrastructure should allow third party software to probe for and retrieve detailed accounting information regarding resource usage both by each individual job as well as aggregated for each end-user.

For what concerns the cloud operations aspect of WP8, no major issues have been noted and the provisioning of cloud service continues to run efficiently satisfying the requirements of the project technical activities. As with the integration activities, this task is also expected to intensify as OpenMinTeD is moving to pre-production state and the platform will open to end users outside of the project consortium. This is expected to increase the requirement for cloud resources (VMs and storage) as actual workloads will start running on the VMs and corpora will be transferred on the storage services. The WP8 support team will continue to support this effort making sure that it will be carried unobstructed.

5. Annex A: Pithos+ as Galaxy storage backend

Galaxy has the potential to use various storage backends (e.g., local disk storage, shared file systems, cloud storage). A storage backend for Pithos+ is needed in order to fully deploy Galaxy on ~okeanos infrastructure. GRNET developed a Pithos+ driver for Galaxy which was offered and accepted as a contribution by the Galaxy community (see https://github.com/galaxyproject/galaxy/pull/3611). This plugin is a part of the core distribution since Galaxy version 17.05.

Galaxy provides a template class ("ObjectStore") allowing third party contributors to develop cloud-specific storage backends. The Pithos+ backend ("PithosObjectStore") is implemented as a subclass of "ObjectStore". The "ObjectStore" class describes a number of basic storage operations for handling storage objects e.g., "create", "delete", "get_data", "get_filename", "get_object_url" and others.

A glance at the internals of the Galaxy storage mechanism reveals a distinction between data sets stored at the same host as Galaxy and sets stored remotely i.e. on a cloud. The locally stored data are the ones used in actual data process, as they are faster to access, while the cloud is used for permanent storage. A simple caching mechanism is implemented to keep local and remote data synchronized.

In specific, connection with Pithos+ is handled through "kamaki"[1] [3], while the caching mechanism is implemented based on tested implementations for Microsoft Azure and Amazon S3, which are already included in Galaxy core.

Given an existing Galaxy deployment, to enable Pithos+ as backend storage, edit the "config/object_store_conf.xml" file accordingly. Assuming a Debian-based host, and the ~okeanos cloud as a storage provider, the addition to the config file would be:

```
<object_store type="pithos" order="1">
      <auth
            url="https://accounts.okeanos.grnet.gr/identity/v2.0"
            ca_certs="/etc/ssl/certs/ca-certificates.crt"
            token="Synnefo-User-Token"/>
      <container
            name="galaxy"
            project="User-project-id-to-provide resources" />

      <extra_dir type="job_work" path="database/job_working_directory_pithos"/>
      <extra_dir type="temp" path="database/tmp_pithos"/>
</object_store>
```

[1] Kamaki is the Python API library and a command line interface for Synnefo [2] which in turn is the software that powers the ~okeanos IaaS cloud

6. *Annex B: Mesos framework integration with Galaxy*

Galaxy provides a set of plugins (i.e. job runners) for running jobs on many systems such as clusters. One of the parts that was missing for the architecture illustrated on Figure 3 is the connection between Galaxy and Chronos. For this purpose, we have implemented a new Galaxy job runner for running jobs in a Mesos cluster via the Chronos scheduler. The primary responsibilities of this runner were:

- Creation of new jobs on Chronos.
- Tracking existing jobs of Chronos.
- Deletion of failed or succeeded jobs.

Specifically, when a Galaxy job is ready for launch, this plugin creates the appropriate specification of the job (e.g. command to be executed, required resources, docker image, data volumes), it is submitted to Chronos which is instructed to run the job immediately on a machine of the cluster. The machine is picked through negotiation between Chronos and Mesos. Once a job is registered on Chronos, the runner monitors the job state, i.e. running, succeeded or failed. If a job is succeeded then the runner marks it as finished and deletes it from Chronos. In the case of a failed job, the runner tries to execute this job again. Note that our runner allows Galaxy administrators to define an upper limit of retries after a failed job.

Typically, a Galaxy job consists of a wrapper shell script which Galaxy creates dynamically inside a working directory. The actual command which is executed on a container inside a machine of the cluster refers to this shell script. In order for the cluster machines to have access to all required data (i.e. shell scripts and datasets), the Galaxy working directory is shared between the Galaxy host and the execution nodes over an NFS filesystem. The shared directory is mounted on the containers, and the job runner adds persistent volumes to the them, so that data created during the execution of the job inside the containers persist even after container deletion. In this way, the output of the jobs is accessible from Galaxy and its users via the NFS server.

To activate our job runner in a Galaxy installation, we need to provide a configuration file. Below we list a typical configuration for our Chronos runner:

```
<?xml version="1.0"?>
<job_conf>
<plugins>
        <plugin id="chronos" type="runner"
load="galaxy.jobs.runners.chronos:ChronosJobRunner" workers="4">
            <param id="chronos">openminted.chronos.gr</param>
            <param id="owner">galaxy@grnet.gr</param>
            <param id="username">username</param>
            <param id="password">password</param>
</plugin>
</plugins>
<handlers>
        <handler id="main"/>
</handlers>
    <destinations default="chronos_dest">
        <destination id="chronos_dest" runner="chronos">
            <param id="docker_enabled">true</param>
            <param id="volumes">/srv/galaxy/working_directory/</param>
            <param id="docker_memory">2048</param>
            <param id="docker_cpu">2</param>
            <param id="max_retries">1</param>
</destination>
</destinations>
</job_conf>
```

The global parameters which are valid for all jobs are specified in the “<plugin>” tag, e.g. the host where Chronos runs, credentials, etc. Inside “<destinations>”, we can parametrize the specification of the jobs. For instance, the above configuration specifies that jobs should run a docker container, with 2048MB of memory, 2 CPUs, and that the directory `/srv/galaxy/working_directory/` (note that this directory is accessible from the Mesos cluster) should be mounted on the container.

Most notably, the job runner that was developed in the context of WP8 is now part of the official Galaxy repository on Github.

7. Annex C: Automated execution backend setup

As described in §3.3 a configuration management tool like Ansible is essential in order to automate the setup of services related to workflow execution over a significant number of VMs. Ansible is simple enough as it uses YAML syntax to declare the configuration of each machine.

A user of ~okeanos service can easily setup the architecture in a few minutes using the Ansible playbook developed in the context of WP8. The playbook is available from GitHub at the following address: https://github.com/openminted/omtd-stack-setup

Specifically, the steps required are:

1. Creation of virtual machines at the ~okeanos service using kamaki tool (CLI tool for interacting with Synnefo API) i.e. command: "`kamaki server create`". Note that the playbook has been tested with VMs running Ubuntu 14.04 LTS.
2. After cloning Ansible playbook, groups of hosts should be specified in a file named "hosts". This is necessary to map machines to services, e.g. machine which run Galaxy, or machines that constitute Mesos cluster, etc. Our playbook provides the following groups:
   - **nfs_server**: The host to run NFS server.
   - **nfs_clients**: A list of hosts with which NFS server shares directories, i.e. hosts running Galaxy and Mesos agents.
   - **mesos_masters**: Machines to act as Mesos masters. For a high-available Mesos cluster, multiple Mesos masters can be specified.
   - **mesos_slaves**: A list of Mesos slave machines. The machines which are actually are going to run the workloads. Since applications are packaged as Docker containers these machines provide a Docker Engine execution environment which allows the actual execution of submitted containers.
   - **chronos**: Machine to run Chronos framework.
   - **galaxy**: Machine that hosts the Galaxy service.
3. The setup process of the architecture is instantiated with the execution of ansible playbook:

   ```
   ansible-playbook -i hosts site.yaml
   ```

Note that there are no restrictions regarding the roles of the machines. For example, the same machine could host both Mesos master and Chronos.

8. References

[1] ~okeanos service home page, http://okeanos.grnet.gr
[2] Synnefo open source IaaS software stack, http://www.synnefo.org
[3] Kamaki project documentation, https://www.synnefo.org/docs/kamaki/latest/
[4] ~okeanos Cyclades Web GUI, http://cyclades.okeanos.grnet.gr
[5] ~okeanos Pithos+ Web GUI, http://pithos.okeanos.grnet.gr
[6] OGF DRMAA Working Group, https://www.drmaa.org/
[7] Docker Swarm, https://www.docker.com/products/docker-swarm
[8] Docker Compose, https://docs.docker.com/compose/
[9] Apache Mesos, http://mesos.apache.org/
[10] Chronos scheduler, https://mesos.github.io/chronos/
[11] Galaxy platform, https://wiki.galaxyproject.org
[12] What is Docker, https://www.docker.com/what-docker
[13] "D8.1 Infrastructure Operation Report", The OpenMinTeD consortium, available at http://openminted.eu/deliverables/
[14] Amazon S3 (Simple Storage Service), https://aws.amazon.com/s3/
[15] OpenStack Swift documentation, https://docs.openstack.org/developer/swift/
[16] Ansible, https://www.ansible.com