Infrastructure Operation Report

May 31, 2018

Deliverable Code: D8.4
Version:1.0 – Final
Dissemination level: Public

D8.4 is an update of the Infrastructure Operation Report document that focuses in project months 31-36 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

D8.4 – Infrastructure Operation Report

WP8 – Operation and Maintenance	
WP participating organizations: ARC, GRNET	
Contractual Delivery Date: 5/2018	**Actual Delivery Date:** 5/2018
Nature: Report	**Version:** 1.0
Public Deliverable	

Preparation slip

	Name	Organization	Date
From	Stavros Sachtouris, Byron Georgantopoulos	GRNET	16/05/2017
Edited by	Byron Georgantopoulos	GRNET	16/05/2017
Reviewed by	Richard Eckart de Castilho Dimitris Galanis	UKP ARC	16/05/2018 17/05/2018
Approved by	Androniki Pavlidou	ARC	31/05/2018
For delivery	Mike Chatzopoulos	ARC	06/06/2018

Document change record

Issue	Item	Reason for Change	Author	Organization
V0.1	Draft version	ToC definition, Contributions in Section 2 and 3	Stavros Sachtouris, Byron Georgantopoulos	GRNET
V0.2	Draft for review	Appendix 1	Stavros Sachtouris	GRNET
V0.3	Draft version	Incorporate comments by reviewers	Richard Eckart de Castilho Dimitris Galanis	UKP ARC
V1.0	Final version	Finalising the document	Byron Georgantopoulos	GRNET

Table of Contents

TABLE OF CONTENTS 3

TABLE OF FIGURES 4

1. INTRODUCTION 8

2. CLUSTER ADJUSTMENTS FOR COMMUNITY USE CASES AND OPEN CALLS 9

2.1 GALAXY INTEGRATION WITH THE WORKFLOW SERVICE 9
2.2 THE "HEAVYWEIGHT" SUB-CLUSTER 10
2.3 GERANOS: A CLUSTER IMAGE MANAGER 12

3. OPERATIONS AND RESOURCE PROVISIONING 13

3.1 OVERALL RESOURCE UTILIZATION 13
3.1.1 OPENMINTED PROJECT AT ~OKEANOS LEGACY SERVICE 13
3.1.2 OPENMINTED PROJECT AT ~OKEANOS-KNOSSOS 13
3.1.2.1 PRODUCTION ENVIRONMENT 14
3.1.2.2 TESTING ENVIRONMENT 15
3.1.3 OPENMINTED AAI PROJECT 16
3.2 VIRTUAL MACHINE PROFILES 16
3.3 TOTAL RESOURCES OCCUPIED 17
3.4 SERVICE INCIDENTS AND DOWNTIMES 18

4. CONCLUSIONS 19

5. APPENDIX 1: OMTD WORKFLOW INFRASTRUCTURE GUIDE 20

5.1 SERVICES 20
5.2 PROVISION 21
5.3 INSTALL WITH ANSIBLE 21
5.4 EXECUTOR 23
5.5 LVM FOR EXECUTOR 24
5.6 EDITOR 25
5.7 MESOS AND CHRONOS 25
5.8 CLUSTER NODES 25
5.9 REVERSE PROXY AND SSL 25
5.10 SSL WITHOUT PROXY 27

6. REFERENCES 28

Table of Figures

Figure 1 - The new ~okeanos-knossos service landing page 8
Figure 2 - OpenMinTeD project resource utilization 9
Figure 3 –Project resource consumption on the ~okeanos-knossos cloud 10
Figure 4 - Resource allocation for the aai.openminted.grnet.gr project 11

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/)

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
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
GUI	Graphical User Interface
NFS	Network File System
TDM	Text and Data Mining

Publishable Summary

This document reports on the activities carried out by WP8 in the final months (M31-M36) of the project. During this period, work focused mainly on the issues arisen during integration testing and the exposition of the platform to the two Open Calls and the Community User Driven Applications [WP9]. This empirical process was the moment of truth for capacity planning and produced valuable feedback: storage requirements on the component/container layer proved to be more resource-hungry than the forecasted resources allocated. The solution required the utilization of supplementary storage and compute resources as well as a partial redesign of the workflow execution infrastructure.

The OMTD Workflow Backend comprises the necessary services and computing resources needed in order to facilitate the execution of TDM workflows on the ~okeanos cloud instances. These services include the Galaxy workflow execution and workflow editor instances, the Apache Chronos scheduler, the Apache Mesos resource manager, the Docker Engine and the private Docker Registry. These services have been deployed on top of the core ~okeanos IaaS services and have been successfully integrated in order to offer a seamless workflow execution environment that allows the execution of Docker-based containerized TDM applications.

During the previous reporting period, a new ~okeanos-knossos instance was installed across a state-of-the-art physical infrastructure located in a new GRNET data center at Knossos, Crete. We were one of the first users to allocate resources from this new instance with the goal of supporting increasing OpenMinTeD requirements and the original resource plan has been updated to accommodate new requirements from the real-life demands posed by community and Open Calls use cases. The added resources have been exploited in this context by both resource providers and developers/testers of the platform.

In order to support present and future operations, some housekeeping work has also been performed. Ansible installation scripts have been thoroughly tested and updated, while deployment and maintenance guides have been published[1]. Most importantly, the development and operation efforts are coordinated in a way that every new feature or bug fix is first tested on the testing platform (on ~okeanos cluster), before applied on the production cluster (on ~okeanos-knossos).

[1] https://github.com/openminted/omtd-stack-setup

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 reported period of work (M31-M36), our activity mainly focused on optimizing and maintaining the software stack to support the workflow execution on the cloud. As the community use cases and open calls materialized into running components in the production infrastructure being used by the Registry and Workflow Service layers, the WP main goal has been to support real life use cases seamlessly. The anticipation was to fine tune the cluster as well as to spot misconfiguration, bugs, security issues and performance bottlenecks. Though most issues were minor and were taken care off quickly, the stand-out issues were related to storage management on the image/container level of the Workflow infrastructure. Our solution was to distinguish components according to their docker image sizes and route them for execution in separate sub-clusters.

The structure of this report is as follows: Section 2 details the evolution of the supplied IaaS whereas Section 3 recaps the operation status and activities that took place during the previous period. Section 4 concludes with a short summary of the achievements. Finally, the Appendix contains the instructions on how the Workflow backend has been setup and configured in the project’s cloud environment.

2. *Cluster adjustments for Community use cases and Open Calls*

2.1 Galaxy integration with the Workflow service

Several changes have been applied to the Galaxy code base, in order to improve performance and tune Galaxy to better suite workflow creation and execution in the context of OMTD. The development team worked on a particular snapshot of the Galaxy software, which incorporated the changes applied by GRNET in previous reporting periods to support cluster integration [10]. The connectors to Synnefo [3] have been accepted by the Galaxy community via GitHub pull requests. On the other hand, the performance issues were already fixed by Galaxy developers in parallel, but on a different version of the tool than the one used in the project. Last but not least, the changes on the editor are restricted to the GUI, are related to branding and functionality restriction and are OMTD-specific, so there was no reason to push them back to Galaxy community. As a result, there are two Galaxy versions: the "editor" for designing workflows in a GUI environment and the "executor" for executing workflows, deployed on separate VMs. Arguments in favor of the separation include the following:

1. Security. The Galaxy editor is a GUI tool exposed to users, the Galaxy executor is a powerful service that can run complicated tasks on a multi-host cluster. The security level required for the two is completely different. Editor users should have more freedom to play around, while the executor should be strictly controlled by trusted services only (on this case, the Workflow service) and should not be accessible by external users.
2. User management. Keeping user management away from the infrastructure was a key decision when designing the OMTD stack from earlier phases as has been explained in previous reports. On the other hand, the GUI editor requires multiuser support, even if it is implemented on a separate level.
3. Modularity. On future versions of OMTD the editor and/or the executor could be replaced with other tools. Keeping them separate facilitates this option.
4. Performance and availability. Workflow execution and workflow editing have diverse performance needs. For instance, from a networking perspective, the editor must support multiple connections outwards (to the users), while the executor inwards (to the cluster). From a storage perspective, the editor handles only component descriptions, while the executor must have access to the corpora and the output of the execution. Hosting them on two separate VMs helps keep the services up and running.

The communication between the OMTD modules involved in workflow execution (Registry, workflow-service, Galaxy editor and executor, cluster nodes) is summarized below:

- TDM component information (a.k.a. Galaxy tool wrapper) is uploaded to a shared NFS directory ("tools") which is accessible by both Galaxy instances. The information in a wrapper is a) used for displaying a component in the Galaxy UI b) for calling/executing the component.
- Workflows are composed by the users on a browser using the Galaxy editor GUI. The resulting workflow (definition & name) is pulled by the Registry through the Galaxy REST API. When a workflow is invoked, OMTD Registry sends it (its name) to the Workflow service which in turn sends it to the Galaxy executor through the respective REST API.
- The corpora to be processed are available in OMTD Store service and are copied to a separate shared NFS space ("database" directory), accessible by the executor Galaxy and the cluster workers.
- Execution progress is monitored by polling the executor Galaxy REST API and execution output is downloaded via Galaxy API (from workflow-service) and stored to OMTD Store service.

In order to support storage needs, an NFS server offers two aforementioned shared spaces, "database" and "tools". Physically, they are both located on the executor Galaxy host, to reduce the number of over-the-network I/Os.

2.2 The "heavyweight" sub-cluster

Several components' docker images turned out to be larger than initially anticipated challenging both the ability to execute and workflow performance. The cluster was originally designed with small, flexible component images in mind, restricted to carry predominantly the TDM software, therefore topping 1-2 GBs per image would be rare. In reality, though, many component providers needed to store large auxiliary data (e.g., models, ontologies, multiple language instances) inside the component image, thus images of a few GB were not uncommon. The same applies for RAM needs, and to a lesser degree for CPU required.

On real tests, when these components were scheduled for execution, they always timed out during the "docker pull" phase (downloading the image to the worker). We overcame this obstacle by pre-downloading the component images manually on all workers. This solved the problem temporarily, but the thin workers run out of storage space very soon

(after 2 or 3 executions). In order to keep the cluster running, the WP8 team had to manually remove old images and continuously monitor storage space.

To cope with this issue, we considered two alternatives: (a) ask providers to separate software from auxiliary data or (b) distinguish components according to their need in storage and handle them in separate sub-clusters.

The first solution would require component providers to modify and re-submit their images, as well as the project team to perform significant changes on every layer of the OMTD platform (Registry, Storage), and effectively modify the policies in the component evaluation mechanism.

The second solution would entail modifications only on the workflow service and infrastructure layer, thus it is easier and safer to implement. The only disadvantage from an infrastructure point of view is that it may cause the underutilization of cluster resources. This drawback can be balanced by manually monitoring cluster utilization and moving resources between sub-clusters according to current needs. Adopting the second solution, components were separated in two categories, according to whether their docker image exceeds a threshold of 1GB in total size. Images smaller than that are tagged as “normal”, while the rest are “heavyweight”.

The “heavyweight” sub-cluster was created by modifying an existing cluster node, reinforcing it with extra storage space. The capacity of this sub-cluster can be increased horizontally (by adding more nodes) or vertically (by adding storage space on existing ones). The node is still connected to the same Mesos cluster, taking advantage of the fact the Mesos master redirects there all components with high size requirements.

The “Chronos runner”, a Galaxy enhancement developed by GRNET[2] to connect Galaxy with the Chronos scheduler, has been modified so that it automatically estimates the size of every component, before sending to Chronos for execution. Specifically, the runner code can now query the docker image registry about the component docker image, thus calculating the size of the image in MB. By convention, the component size is defined to be 256MB if the image size is below this threshold, or 50GB if higher, thus separating components into two categories, based on their image size. The (conventional) image size is attached on the request sent to Chronos to be used during the resource negotiation between Chronos and Mesos master.

[2] https://github.com/galaxyproject/galaxy/pull/4120, “Fix chronos client initialization”

The distinction between heavyweight and normal components is technically implemented in the Chronos runner in a way that no modifications are required on the configuration of Chronos or Mesos. The Mesos master does not distinguish between heavyweight and normal nodes, it just directs all heavyweight components to the nodes with adequate storage space. Normal nodes have typically less than 50GB storage in total, so they will never receive a heavyweight component. In order for this solution to work in practice, though, the images of the heavyweight components should already be pulled (downloaded) to each of the nodes used to serve heavyweight components.

2.3 Geranos: A cluster image manager

A new service, called "Geranos"[3] has been developed by GRNET[4] to manage the various types of component images on the cluster. The service provides a REST API which is used by the Workflow service or the Registry service to perform pre-execution operations on the image level.

Geranos allows services to check whether a particular image is available on the cluster, it can perform a "docker pull" on particular images across the cluster and it can distinguish between "heavyweight" and "normal" sub-clusters.

When a heavyweight component is to be executed, the Workflow service queries Registry to see if the latest version of the component image is already pulled to the heavyweight cluster. If not, Geranos will pull it. The Workflow service will wait until the image is pulled. The component is then executed through the Workflow service.

[3] https://github.com/openminted/geranos

[4] Geranos is developed in python and is published by GRNET under GPLv3.

3. Operations and Resource Provisioning

3.1 Overall Resource Utilization

Two ~okeanos projects[5] 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.

3.1.1 OpenMinTeD project at ~okeanos legacy service

The **openminted.grnet.gr** ~okeanos project provides cloud resources for generic usage in the project. Specifically, it is currently being used for testing and staging the platform software and deployment. The platform deployed on ~okeanos was exposed to the first open call and remains constantly in operation to assist any required software fixes and modifications. At the end of M36 a snapshot of resource consumption was as follows:

RESOURCES

	Max per member	*Total*	*Usage*
File Storage Space	500.0 GB	1.00 TB	**0% (1.35 GB)**
Hard Disk Storage	8.00 TB	12.0 TB	**42% (5.13 TB)**
CPUs	400	1600	**14% (234)**
RAM	800.0 GB	3.00 TB	**10% (317.0 GB)**
VMs	50	500	**8% (44)**
Private Networks	20	20	**20% (4)**
Public IPs	64	64	**98% (63)**

Figure 1 - Resource allocation for the OpenMinTeD project

Supplementary storage resources were needed in order to develop, test and address requirements posed by “heavyweight” component images. The cluster continues to be the testbed pool of resources for all tiers of the OMTD platform.

3.1.2 OpenMinTeD project at ~okeanos-knossos

The OpenMinTeD project at ~okeanos-knossos is dedicated to hosting the production services of the OpenMinTeD platform. The current deployment essentially reflects the implementation requirements of the platform architecture as it has been defined in the context of D8.3 [10] and D6.8 [11]. The following table summarizes the virtual machines instantiated and their roles within the deployment.

[5] https://accounts.okeanos.grnet.gr/ui/projects/how_it_works

3.1.2.1 Production Environment

VM #	Role	CPU cores	RAM (GB)	HD (GB)	Public IPv4	Private IP	OS
1	Galaxy & NFS	8	16	730	1	0	Ubuntu Server
2	Mesos/Chronos	8	8	30	1	1	Ubuntu Server
3	Worker 1	8	16	30	0	1	Debian 8
4	Worker 2	8	16	30	0	1	Debian 8
5	Worker 3	8	16	30	0	1	Debian 8
6	Worker 4	8	16	30	0	1	Debian 8
7	Worker 5 (heavyweight)	8	16	430	0	1	Debian 8
8	Docker Monitoring (Prometheus & Grafana)	4	8	30	1	1	Ubuntu Server
9	Docker registry	4	8	210	1	1	Debian 8
10	Galaxy Editor	8	16	30	1	0	Ubuntu Server
11	Jumbstart[6] host	2	2	30	1	0	Ubuntu Server
12	Workflow Service	8	16	30	1	0	Ubuntu Server
13	Testing Host	8	16	30	1	0	Ubuntu Server
	Total	**90 cores**	**186 GB**	**1,6 TB**	**7**	**7**	

Table 1: Detailed list of ~okeanos-knossos production VMs

[6] Jumbstart (or jump host) is the gateway host which provides access to the rest of the hosts. Instead of adding public keys to every host, we establish a secure PPK authentication between the Jumbstart and the rest of the hosts. Then, we give developers and operators access only to the Jumbstart.

3.1.2.2 Testing Environment

VM #	Role	CPU cores	RAM (GB)	HD (GB)	Public IPv4	Private IP	OS
1	Galaxy (executor) & NFS	8	8	460	1	0	Ubuntu Server
2	Galaxy (editor)	8	8	60	1	0	Ubuntu Server
3	Mesos/Chronos	8	8	40	1	1	Ubuntu Server
4	Worker 1	8	8	20	0	1	Debian 8
5	Worker 2	8	8	20	0	1	Debian 8
6	Worker 3	8	8	20	0	1	Debian 8
7	Worker 4 (heavyweight)	8	8	420	0	1	Debian 8
8	Docker Monitoring (Prometheus)	8	8	60	1	1	Ubuntu Server
9	Docker registry	8	8	260	1	1	Debian 8
10	Jumpstart host	2	1	20	1	0	Ubuntu Server
	Total	**74 cores**	**75 GB**	**1,4 TB**	**6**	**7**	

Table 2: Detailed list of testing VMs

Figure 2 reflects the current resource utilization of the project in ~okeanos-knossos. Initially a smaller fraction comparing to ~okeanos legacy resources was allocated, but response to Open Calls necessitated higher allocation and for some resource types they have already reached their limits. Due to this reason, additional resources will be allocated if needed in the future.

RESOURCES

	Max per member	Total	Usage
Hard Disk Storage	1.46 TB	1.46 TB	98% (1.44 TB)
CPUs	128	128	70% (90)
RAM	256.0 GB	256.0 GB	66% (170.0 GB)
VMs	32	32	40% (13)
Private Networks	32	32	0% (0)
Public IPs	32	32	62% (20)

Figure 2 – Project resource consumption on the ~okeanos-knossos cloud

3.1.3 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 almost full capacity having allocated a high percentage of the resources originally allocated for this purpose. 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 3 - Resource allocation for the aai.openminted.grnet.gr project

3.2 Virtual Machine profiles

The following table summarizes the resources utilized by OpenMinTeD projects as of the writing of this report, taking into account stable services besides the workflow backend, running either on production basis or permanent pre-production services currently used for experimentation and development.

VM #	Role	CPU cores	Main Memory (MB)	Volume Storage (GB)
1	Annotation editor server (WebAnno)	4	4096	20
2	WP9 Demonstrator 1	8	8192	60
3	WP9 Demonstrator 2	8	8192	60
4	WP9 Demonstrator 3 TDM linguistic pipeline	8	8192	60

5	WP9 Demonstrator 4 Virtuoso triple store	8	8192	60
6	WP9 Demonstrator 5	8	8192	60
7	WP9 Demonstrator 6	8	8192	60
8	Galaxy development (INRA)	2	6144	40
9	Galaxy development (GRNET)	2	4096	20
10	HTTP Load balancer (nginx)	4	2048	5
11	HTTP Load balancer (nginx/standby)	4	2048	5
12	Cache (memcached)	1	1024	5
13	Cache (memcached)	1	1024	5
14	DB (postgresql/master)	2	2048	20
15	DB (postgresql/hot standby)	2	2048	20
16	OIDC (MITREid Connect)	4	4096	10
17	OIDC (MITREid Connect)	4	4096	10
18	Backup (barman)	2	2048	60
19	Development-master	4	8192	60
20	Development-slave	4	8192	40
21	Development-services1	4	8192	60
22	Development-services2	4	8192	60
23	Development-env2-master	4	8192	60
24	Development-env2-slave	4	8192	60
25	Demo-master	4	8192	60
26	Demo-slave	4	8192	60
27	Registry Production	4	8192	60
28	Proxy.openminted.eu	4	8192	60
29	Repo.openminted.eu	4	8192	60
30	Ontology server	2	4096	40
	Total Resources	**126 (CPU cores)**	**174 GB**	**1,25 TB**

Table 3: VM profiles throughout ~okeanos projects beyond the Workflow engine

3.3 Total resources occupied

The following table summarizes the total utilization of cloud resources across ~okeanos and ~okeanos-knossos:

	CPU	RAM (GB)	Disk (GB)	VM
AAI	32	28	160	11
Services & testing	234	317	5130	44
Workflow backend Production	90	170	1830	13
TOTAL	356	515	7,1TB	68

Table 4: Total cloud resources for OpenMinTeD

3.4 Service incidents and downtimes

No unplanned downtime was experienced this period.

Synnefo update: on Mar 20, 2018, Synnefo [3], the underlying cloud software that powers ~okeanos IaaS instances, was scheduled for update. Due to this, ~okeanos and ~okeanos-knossos service were disrupted on that day between 9:00 and 10:00 EET. Virtual machines were not affected, while Astakos and Cyclades Web UIs and their APIs were unavailable during the maintenance window.

4. Conclusions

During months M31-M36 and towards the end of the project, the primary work of WP8 concentrated on the smooth running and improvements/configurations of the OpenMinTeD workflow backend functionality.

In the final months of the project, we continued working on the stabilization of the OpenMinTeD workflow backend functionality and maintenance of all the testing and execution environments across the two data centres. As the OpenMinTeD platform attracted new users from the Open Calls, demand on computing and storage needs increased the load and utilization of the services and necessitated the addition of a new sub-cluster for handling resource-heavy components.

WP8 has also continued to facilitate monitoring of resource consumption by Docker containers running as part of Galaxy workflows. As workflows are submitted from the OpenMinTeD Registry the infrastructure should allow appropriate 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.

Finally, on the core cloud operations aspect of WP8, no major issues have been noted and the provisioning of cloud resources will continue to timely and efficiently satisfy the requirements of the project technical activities. As with the integration activities, this task also intensified as OpenMinTeD has been operating, and continues to operate after the end of the project, on production status and the platform is opening to a greater audience. This will further increase the requirement for VMs and storage as more workloads will start running on the VMs and corpora will be transferred on the storage services. The WP8 team will continue to support this effort and dynamically provide the necessary cloud resources.

5. Appendix 1: OMTD Workflow Infrastructure guide

This is a guide to setup the OpenMinTeD Workflow Infrastructure stack on an IaaS with adequate resources. The installation procedure is based on configuring and running the "omtd-stack-setup" suite of ansible scripts [12] which has been developed and is being updated by GRNET. The guide and the script have been tested extensively on ~okeanos and ~okeanos-knossos IaaS installations, but they can be reproduced on any IaaS with Debian/Ubuntu VMs, public IPs and storage resources.

5.1 Services

All services are split as much as possible, then:

- Executor Galaxy) - executes TDM tools and workflows, used through REST API
- Editor Galaxy - offers an editor for creating and editing workflows
- NFS Server - offers a shared folder for the tools between the two galaxies, as well as a shared folder between the Executor Galaxy and the cluster nodes
- Chronos scheduler - schedulers the execution of tools
- Mesos cluster manager - manages the cluster that executes the tools
- Mesos slave - handles execution on slave nodes
- CAdvisor - monitors the execution of tools on the hosts they are executed
- Prometheus-node-exporter - exports the results of the monitoring to the Prometheus aggregator
- Prometheus - monitors the Mesos slaves and offers the results through a REST API
- Grafana - visualization of Prometheus monitoring
- Docker Registry - hosts the images of the TDM tools
- Geranos – a REST API to manage docker images and containers across the cluster

Suggested setup per host:

- Executor: Executor Galaxy, NFS server
- Server Editor: Editor Galaxy
- Cluster Manager: Chronos scheduler, Mesos master

- Cluster Nodes (normal + heavyweight): Mesos slave, CAdvisor, Node-Exporter
- Monitoring: Prometheus + Grafana
- Tool Registry: Docker Registry, Geranos

Note: You can start with 2 or 3 nodes. It is easy to scale up (or down) later.

5.2 Provision

Make sure to create the following VMs

- Executor VM: Ubuntu 16.04 LTS. Reasonable resources, with at least one extra volume for the NFS shares (may use LVM).
- Editor VM: Ubuntu 16.04 LTS with reasonable resources
- Cluster Manager: Ubuntu 16.04 LTS with reasonable resources
- Monitoring: Ubuntu 16.04 LTS with reasonable resources
- Tool Registry: Debian Jessie with reasonable resources and sizable disk space (e.g., a few hundred gigs)
- Cluster Nodes (normal): Debian Jessie with backports and as much CPU and RAM as possible.
- Cluster Nodes (heavyweight): Debian Jessie with backports and as much CPU and RAM as possible. Also, at least 0.5T of disk space.

Make sure you have your public key in */root/.ssh/authorized_keys* of each and every of the hosts above. Also, make sure you can log on with ssh without any obstacles or warnings.

Heavyweight cluster nodes are handled as regular nodes when deploying the stack. They are treated in a special way when the cluster is functional, though. The ratio of heavyweight to normal nodes depends on the expected size of OMTD components, but as a rule of a thump, a 1/3 ratio suffices.

5.3 Install with ansible

We assume you've got a copy of *omtd-stack-setup* repository and the IPs of the machines of the previous section.

Make a copy of *hosts*:

```
$ cp hosts hosts.production
```

Edit "hosts.production" so that:

```
[nfs_server]
<Executor IP>

[docker_registry]
<Tool Registry IP>

[cluster_master]
<Cluster Manager IP>

[cluster_nodes]
<Cluster Node 1 IP>
<Cluster Node 2 IP> ...

[executor]
<Executor IP>

[editor]
<Editor IP>

[monitor]
<Monitor IP>
```

Edit *group_vars/all* to set credentials for inter-service communication. Make sure to replace passwords and secrets with new ones that are long and hard to guess.

```
chronos_http_user: chronosadmin
chronos_http_password: chronospassword

chronos_principal: "chronos.omtd"
chronos_secret: "chronos.secret"
```

• • •

```
cluster_nodes:
    "83.212.XXX.XXX": {
        principal: "node1.omtd",
        secret: "node1.secret"

    }, "83.212.XXX.YYY": {
        principal: "node2.omtd",
        secret: "node2.secret"
    }
```

In *group_vars/executor*:

galaxy_admin: <email of the galaxy admin user>

In *group_vars/editor*:

remote_user_maildomain: <the domain of the user who is going to connect to the editor>

remote_user_secret: <a long, hard to guess string>

Now, install requirements and run the ansible playbook to install the services. It might take a while:

$ ansible-galaxy install -r requirements.yaml $ ansible-playbook -i hosts.production site.yaml

5.4 Executor

First go to Reverse proxy and SSL <proxy_ssl> to setup HTTP (and HTTPS).

Now, make sure the following lines are included in */etc/apache2/sites-enabled/vhosts-le-ssl.conf*:

```
<VirtualHost *:443>
    ...
    <Directory>
    Require all granted
    </Directory>
    ...
</VirtualHost>
```

Restart apache2.

Test if you can reach your host through HTTP(s). You should be able to reach Galaxy.

Galaxy requires to create an admin user first. To do this, you must change the Galaxy configuration to allow users to be created.

In /srv/executor/config/galaxy.ini find and commend out the following line: *allow_user_creation = False*
Restart galaxy: *$ service galaxy restart*
Connect to Galaxy through the web UI (just the fqdn of the host). On the top menu click *Login or Register* > *Register*. Fill out the form. The email field should be in the admin_users field in */etc/executor/config/galaxy.ini*.
Uncommend *allow_user_creation = False* and restart galaxy. You are good to go.

5.5 LVM for executor

You can hide the database volume as an LVM. This will allow easy (but manual) storage scaling (see the operations chapter).

1. Install lvm:

```
$ apt install lvm
```

2. Find the unused physical volume(s) and prepare it(them) :

```
$ fdisk -l
...
Disk /dev/vdb: 380 GiB, 408021893120 bytes, 796917760 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
...
$ pvcreate /dev/vdb [/dev/vdc [...]]
```

3. Create a new LVM group nfsstore, then create a logical partition and format it:

```
$ vgcreate nfsstore /dev/vdb [/dev/vdc [...]]
$ lvcreate -l 100%FREE -n nfs_logical nfsstore
$ mke2fs -t ext4 /dev/nfsstore/nfs_logical
```

4. Mount it:

```
$ echo "/dev/nfsstore/nfs_logical /srv/executor/database ext4 defaults,nofail 0 0">>/etc/fstab
$ mount -a
```

Make sure the target directory /srv/executor/database exists (in some deployments it is /srv/galaxy/database).

5.6 Editor

First go to Reverse proxy and SSL <proxy_ssl> to setup HTTP (and HTTPS).

Then, try to connect to the host (just the IP). You should be redirected to Galaxy, but get rejected with this message:

Access to Galaxy is denied.

This is because the editor is configured to accept only remote users from a specific domain (remote_user_maildomain), who authenticate themselves with a secret (remote_user_secret). All users from this domain can use the editor, as long as their requests contain the secret.

5.7 Mesos and Chronos

Everything is set up, but it is good to secure communication with SSL. You can do that with letsencrypt, if you follow the instructions in SSL without proxy <just_ssl>.

5.8 Cluster nodes

Everything is set up, but it is good to secure communication with SSL. You can do that with letsencrypt, if you follow the instructions in SSL without proxy <just_ssl>.

5.9 Reverse proxy and SSL

Our ansible scripts setup Apache2 as a reverse proxy on the hosts that need a reverse proxy, but only as HTTP.

On the editor host, make sure '/etc/apache2/vhosts.conf' looks like this:

```
DirectoryIndex index.html

<VirtualHost *:80>
        ServerName 123.45.67.89

        RewriteEngine on RewriteRule ^(.*)
        [http://localhost:8080$1](http://localhost:8080$1) [P]
</VirtualHost>
```

Make sure these apache modules are enabled: ssl, rewrite, proxy, proxy_http: *$ a2query -m <module>* To enable a disabled module: *$ a2enmod <module>*

Restart apache2:
$ service apache2 restart
At this point, thinks should work well without SSL, but that is going to change in the following lines.

First, remove the RewriteRule line from */etc/apache2/sites-enabled/vhosts.conf*
In the following we install "let's encrypt" free certificates. If you don't want to use these, you must figure some other way to setup your HTTPS proxy.

In Debian:

```
$ echo 'deb <http://ftp.debian.org/debian> jessie-backports main' |
sudo tee /etc/apt/sources.list.d/backports.list
$ sudo apt-get update
$ sudo apt-get install python-certbot-apache -t jessie-backports
```

In Ubuntu:

```
$ sudo apt-get install software-properties-common
$ sudo add-apt-repository ppa:certbot/certbot
$ sudo apt-get update
$ sudo apt-get install certbot
```

Automatically set up certificates:

```
$ sudo certbot --apache
: pick "vhosts.conf" and HTTPS only when asked
```

Restart apache2 and check that the host is redirecting to the correct place: *$ sudo service apache2 restart*

5.10 SSL without Proxy

In Debian:

```
$ echo 'deb <http://ftp.debian.org/debian> jessie-backports main' |
sudo tee /etc/apt/sources.list.d/backports.list
$ sudo apt-get update
$ sudo apt-get install certbot -t jessie-backports
```

In Ubuntu:

```
$ sudo apt-get install software-properties-common
$ sudo add-apt-repository ppa:certbot/certbot
$ sudo apt-get update
$ sudo apt-get install certbot
```

Now, install certbot and get the certificates. Make sure to replace example.com with your domain:

```
$ sudo certbot certonly --standalone -d example.com
$ sudo certbot renew
```

6. *References*

[1] ~okeanos service home page, https://okeanos.grnet.gr
[2] ~okeanos-knossos service home page, https://okeanos-knossos.grnet.gr
[3] Synnefo open source IaaS software stack, https://www.synnefo.org
[4] Chronos scheduler, https://mesos.github.io/chronos/
[5] Mesos resource allocator, http://mesos.apache.org/
[6] Grafana, https://grafana.com/
[7] Prometheus, https://prometheus.io/
[8] Geranos, https://github.com/openminted/geranos
[9] Galaxy platform, https://wiki.galaxyproject.org
[10] "D8.3 Infrastructure Operation Report", The OpenMinTeD consortium, available at http://openminted.eu/deliverables/
[11] "D6.8 Platform services distribution specification", The OpenMinTeD consortium, available at http://openminted.eu/deliverables/
[12] Workflow backend deployment guide: https://github.com/openminted/omtd-stack-setup