Large-scale EXecution for Industry & Society

Deliverable D4.2

Design and Implementation of the HPC-Federated Orchestration System - Intermediate

Co-funded by the Horizon 2020 Framework Programme of the European Union
Grant Agreement Number 825532
ICT-11-2018-2019 (IA - Innovation Action)

DELIVERABLE ID | TITLE	D4.2 | Design and Implementation of the HPC-Federated Orchestration System - Intermediate
RESPONSIBLE AUTHOR	Laurent Ganne (Bull/Atos)
WORKPACKAGE ID | TITLE	WP4 | Orchestration and Secure Cloud/HPC Services Provisioning
WORKPACKAGE LEADER	LINKS
DATE OF DELIVERY (CONTRACTUAL)	29/02/2020 (M14)
DATE OF DELIVERY (SUBMITTED)	01/03/2020 (M14)
VERSION | STATUS	V1.1 | Final
TYPE OF DELIVERABLE	O (Other)
DISSEMINATION LEVEL	PU (Public)
AUTHORS (PARTNER)	Laurent Ganne (Bull/Atos)
INTERNAL REVIEW	Emanuele Danovaro (ECMWF); Stephen Blair-Chappell (BLABS)

Project Coordinator: Dr. Jan Martinovič – IT4Innovations, VSB – Technical University of Ostrava
E-mail: jan.martinovic@vsb.cz, **Phone:** +420 597 329 598, **Web:** https://lexis-project.eu

DOCUMENT VERSION

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
0.1	First Draft	14/01/2020	Laurent Ganne (Bull/Atos)
0.2	Review Added a workflow screenshot	28/01/2020	Alberto Scionti (LINKS); François Exertier (Bull/Atos); Laurent Ganne (Bull/Atos)
0.3	Review	14/02/2020	Marc Levrier (Bull/Atos)
0.4	Changed figure 1	17/02/2020	Alberto Scionti (LINKS); Jan Křenek (IT4I)
0.5	Review	26/02/2020	Stephen Blair-Chappell (BLABS)
0.6	Review	27/02/2020	Emanuele Danovaro (ECMWF)
1.0	Updated github links, figure 1	28/02/2020	Laurent Ganne (Bull/Atos)
1.1	Final check by the coordinator	29/02/2020	Katerina Slaninova (IT4I)

GLOSSARY

ACRONYM	DESCRIPTION
ALIEN4CLOUD	Application LIfecycle ENablement for Cloud
A4C	Alien4Cloud
API	Application Program Interface
CLI	Command Line Interface
DDI	Distributed Data Infrastructure
HA DEPLOYMENT	High Availablilty Deployment
HEAPPE	High-End Application Execution Middleware
HPC	High Performance Computing
IAAS	Infrastructure as a Service
REST API	Application Interface that uses HTTP requests to GET, PUT, POST and DELETE data
SAML	Security Assertion Markup Language
TLS	Transport Layer Security
TOSCA	Topology and Orchestration Specification for Cloud Applications
UI	User Interface
VM	Virtual Machine
YORC	Ystia ORChestrator
WCDA	Weather and Climate Data API

TABLE OF PARTNERS

ACRONYM	PARTNER
Avio Aero	GE AVIO SRL
AWI	ALFRED WEGENER INSTITUT HELMHOLTZ ZENTRUM FUR POLAR UND MEERESFORSCHUNG
BLABS	BAYNCORE LABS LIMITED
Bull/Atos	BULL SAS
CEA	COMMISSARIAT A L ENERGIE ATOMIQUE ET AUX ENERGIES ALTERNATIVES
CIMA	Centro Internazionale in Monitoraggio Ambientale - Fondazione CIMA
CYC	CYCLOPS LABS GMBH
ECMWF	EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS
GFZ	HELMHOLTZ ZENTRUM POTSDAM DEUTSCHESGEOFORSCHUNGSZENTRUM GFZ
IT4I	VYSOKA SKOLA BANSKA - TECHNICKA UNIVERZITA OSTRAVA / IT4Innovations National Supercomputing Centre
ITHACA	ASSOCIAZIONE ITHACA
LINKS	FONDAZIONE LINKS / ISTITUTO SUPERIORE MARIO BOELLA ISMB
LRZ	BAYERISCHE AKADEMIE DER WISSENSCHAFTEN / Leibniz Rechenzentrum der BAdW
NUM	NUMTECH
O24	OUTPOST 24 FRANCE
TESEO	TESEO SPA TECNOLOGIE E SISTEMI ELETTRONICI ED OTTICI

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

EXECUTIVE SUMMARY

The LEXIS HPC/Cloud Orchestration System provides an application deployment platform for Hybrid HPC/Cloud applications. The applications to be deployed are modelled using TOSCA [1] (Topology and Orchestration Specification for Cloud Applications), an OASIS consortium standard language to describe an application made of components, with their relationships, requirements, capabilities, operations.

A front-end is provided in which applications can be constructed from an extensible catalogue of TOSCA components. The front-end is also used to describe workflows - sequences of operations on application components - that can be launched and monitored using either a UI or REST API commands.

Position of the deliverable in the whole project context

This deliverable is a part of Task 4.4 - Integration of the Overall HPC/Cloud Orchestration System.

It is an intermediate deliverable. The final version will be deliverable **D4.6 Design and Implementation of the HPC-Federated Orchestration System – Final** [2], due in month 24.

The final version will enable the building and execution of workflows for WP5 (Aeronautics), WP6 (Earthquake and Tsunami), WP7 (Weather and Climate) pilots on Cloud/HPC infrastructures described in WP2 (LEXIS Requirements Definition and Architecture Design), through WP8 (LEXIS Portal), using LEXIS Distributed Data Infrastructure (DDI) provided by WP3 (LEXIS Data System) and LEXIS Weather and Climate Data API (WCDA) provided by WP7 to manage datasets.

Description of the deliverable

This deliverable provides details of the software implementing the intermediate version of the Orchestration system and its installation on LEXIS infrastructure.

The intermediate version is based on open source software: the orchestrator back-end (Yorc) and front-end (AlienCloud), provided by Bull/Atos. Additional components were developed under the LEXIS project: a plugin extending the orchestrator to support the management of HPC infrastructure resources through the middleware developed by IT4I (HEAppE[1]), and a Go client providing an API through which the LEXIS Portal can interact with the orchestration system.

This intermediate version provides the basic functionality needed to deploy and run application workflows on a hybrid HPC/Cloud infrastructure. The infrastructure contains both an IaaS OpenStack location (where the orchestrator creates Virtual Machines on demand, installs software components, executes scripts), and an HPC location (where jobs are submitted and monitored through HEAppE).

The requirements of the orchestration system are described in deliverables: **D4.3 Definition of data access priority, analytics policies and security assessment** [3], and **D4.4 Definition of workload management policies in federated cloud/HPC environments** [4]**.**

Some of the orchestration system requirements are not yet fulfilled by this intermediate version, for example: the use of DDI API to manage datasets or synchronize checkpointing data on remote infrastructures for job failover; policy to launch the same job over several infrastructures in order to get resources as soon as possible for urgent computing; dynamic orchestration (currently locations where to deploy application components are statically defined before the deployment is started, whereas in the final version, the orchestrator will be able to compute dynamically these locations during the workflow execution, through the execution of placement policies).

Contributors

LINKS as the Work Package leader, defined the needs and gathered requirements from other Work Packages. Bull/Atos provided the orchestrator software and implemented the orchestrator enhancements/innovations needed for LEXIS. IT4I provided access and resources to the IT4I infrastructure, as well as the HPC as a service

[1] High-End Application Execution Middleware: http://heappe.eu

middleware HEAppE used on both sites IT4I and LRZ. LRZ provided access and resources to the LRZ Infrastructure. O24 provided a Keycloak environment to which the Orchestrator front-end delegates its user authentication. CYC and BLABS provided a list of requirements and code contributions to the orchestrator client library used by LEXIS Portal.

1 ORCHESTRATION SYSTEM OVERVIEW

The current version of the LEXIS HPC/Cloud orchestration system provides:

- an API, Go client library used by the LEXIS Portal to interact with the Orchestration system
- a front-end Alien4Cloud (or A4C - Application LIfecycle ENablement for Cloud) providing a UI and REST API, allowing the creation of applications and workflows from an extensible catalogue of components,
- A4C Yorc plugin that provides front-end A4C the ability to use the Ystia orchestrator (Yorc) to manage applications lifecycle and run workflows
- Yorc, the Ystia orchestrator managing the application lifecycle and workflow execution
- Yorc HEAppE plugin, extending Yorc so it can use HEAppE (High-End Application Execution Middleware) as the framework managing HPC infrastructures in LEXIS.

In addition, the Ystia Forge, a collection of TOSCA components and application templates, provides examples of Big Data applications such as Hadoop, log analysis, stream processing, data science development environments, databases, a load balancer/reverse proxy, docker registry, etc...

TOSCA components and an application template example showing a Hybrid HPC/Cloud workflow over LEXIS HPC infrastructure through HEAppE and OpenStack are also provided.

The final version of the orchestration system in addition to the above will also contain additional Monitoring, Business Logic, Authentication and Authorization Infrastructure connector components. These components will implement metrics collection and workload management policies described in deliverable D4.4 Definition of workload management policies in federated cloud/HPC environments [4].

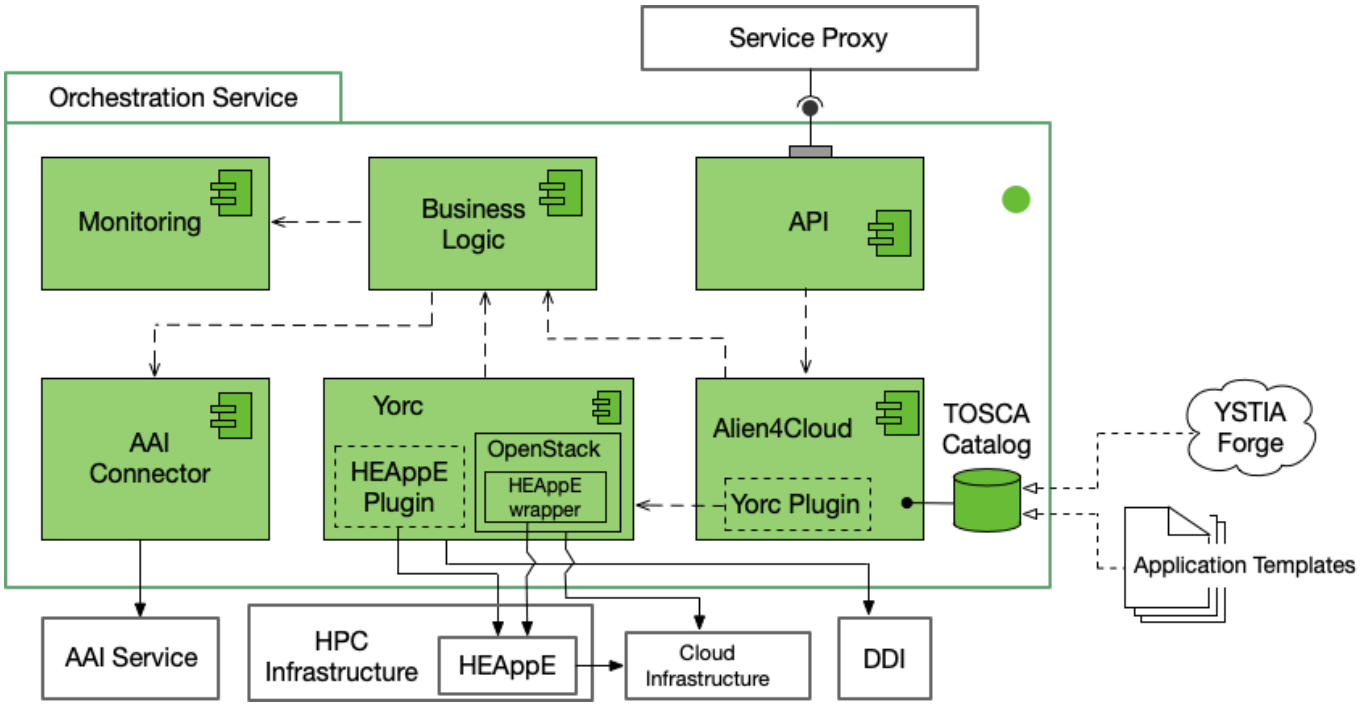

Figure 1 Orchestration System Overview

2 SOFTWARE REPOSITORIES

This section describes the software repositories for each LEXIS Orchestrator system component. The release versions contained in this document are the versions installed in the intermediate delivery. These versions will be updated regularly during the project time-line, each time a new feature fulfilling a LEXIS requirement is delivered.

2.1 ALIEN4CLOUD

Alien4Cloud (A4C) is the orchestration front-end (UI and REST API). It contains an extensible catalogue of TOSCA components, a studio to design applications from this catalogue, and associated workflows. It relies on an orchestrator (by default Yorc - Ystia orchestrator) to manage application lifecycles and to run workflows on the infrastructures supported by the orchestrator.

SOURCE CODE	https://github.com/alien4cloud/alien4cloud
LICENSE	Apache 2.0
DOCUMENTATION	http://alien4cloud.github.io/
RELEASE	version 2.2.0-SM10 https://fastconnect.org/maven/content/repositories/opensource/alien4cloud/alien4cloud-dist/2.2.0-SM10/alien4cloud-dist-2.2.0-SM10-dist.tar.gz

Table 1 Alien4Cloud Source Code

2.2 ALIEN4CLOUD GO CLIENT LIBRARY

The A4C Go client library is a library implemented in Golang, using the A4C REST API.

It was implemented after discussion with the WP8 team in order to ease the integration of the orchestration service front-end into the LEXIS Portal.

SOURCE CODE	https://github.com/alien4cloud/alien4cloud-go-client
LICENSE	Apache 2.0
DOCUMENTATION	Usage examples in the repository: https://github.com/alien4cloud/alien4cloud-go-client/blob/master/README.md
RELEASE	2.0.0-M1, available to go programs as a go module by pointing to this repository and version.

Table 2 Alien4Cloud GO Client Library Source Code

2.3 ALIEN4CLOUD YORC PROVIDER PLUGIN

The A4C Yorc provider plugin is a plugin allowing A4C to rely on the Ystia Orchestrator (Yorc) to manage application lifecycles and to run workflows on infrastructures supported by the orchestrator.

Yorc being the default A4C orchestrator, this plugin is bundled within the A4C distribution (both A4C and this plugin are released under the same version number).

SOURCE CODE	https://github.com/alien4cloud/alien4cloud-yorc-provider
LICENSE	Apache 2.0
DOCUMENTATIO N	https://alien4cloud.github.io/#/documentation/2.1.0/orchestrators/yorc/index.html
RELEASE	version 2.2.0-SM10 https://fastconnect.org/maven/content/repositories/opensource/alien4cloud/alien4cloud-yorc-provider/2.2.0-SM10/alien4cloud-yorc-provider-2.2.0-SM10.zip

Table 3 Alien4Cloud Yorc Provider Plugin Source Code

2.4 YORC

The Ystia Orchestrator (Yorc) is a TOSCA [1] orchestrator supporting application/job lifecycle management over hybrid infrastructures (HPC scheduler SLURM, Kubernetes, OpenStack, several public clouds). It provides an implementation of operations allowing the creation/deletion of infrastructure resources (jobs, compute instances, block-devices, etc.) on demand. It also allows the installation of applications and to run workflows on these infrastructures. It can support additional infrastructures through the plugins described in the next section.

SOURCE CODE	https://github.com/ystia/yorc
LICENSE	Apache 2.0
DOCUMENTATION	https://yorc.readthedocs.io/en/latest/
RELEASE	version: 4.0.0-M6 https://bintray.com/ystia/yorc-engine/download_file?file_path=4.0.0-M6%2Fyorc-4.0.0-M6.tgz

Table 4 Yorc Source Code

2.5 YORC HEAPPE PLUGIN

Yorc HEAppE plugin extends the Ystia Orchestrator (Yorc) so it can use HEAppE as the middleware. This middleware is used to submit and monitor jobs, copy files to the relevant input directories, and to get result files on HPC infrastructures.

SOURCE CODE	https://github.com/lexis-project/yorc-heappe-plugin
LICENSE	Apache 2.0
RELEASE	version: 1.0.0 https://github.com/lexis-project/yorc-heappe-plugin/releases/download/v1.0.0/heappe-plugin.zip
DOCUMENTATION	https://github.com/lexis-project/yorc-heappe-plugin/blob/master/README.md

Table 5 Yorc HEAppE Plugin Source Code

2.6 YSTIA FORGE

The Ystia Forge provides TOSCA components and application templates that can be used in the orchestration service, by being uploaded in A4C catalogue of components. For example, it provides the implementation of a

TOSCA component that allows the installation of Docker on a host, and a TOSCA component to support the running of a docker container, used in the example described in the next section.

SOURCE CODE	https://github.com/ystia/forge
LICENSE	Apache 2.0
RELEASE	version: 2.1.0 https://github.com/ystia/forge/releases/tag/v2.1.0 (the development version can also be used, importing directly the GitHUB repository in Alien4Cloud catalogue)
DOCUMENTATION	https://github.com/ystia/forge/blob/develop/org/ystia/README.rst

Table 6 Ystia Forge Source Code

2.7 EXAMPLE OF HYBRID HPC/CLOUD DEPLOYMENT

An example of hybrid Cloud/HPC application is provided. This example is similar to the ones used in the pilot case studies and combines both job execution on HPC infrastructures and the running of software in OpenStack compute instances.

The example provides a workflow that first submits a job on the HPC infrastructure, then waits for the job to be run. Once the job completes, a Virtual Machine is created on the Cloud infrastructure, Docker is installed on this Virtual Machine, and finally an Apache Docker container is run (using TOSCA components available in the Ystia Forge described in the previous section):

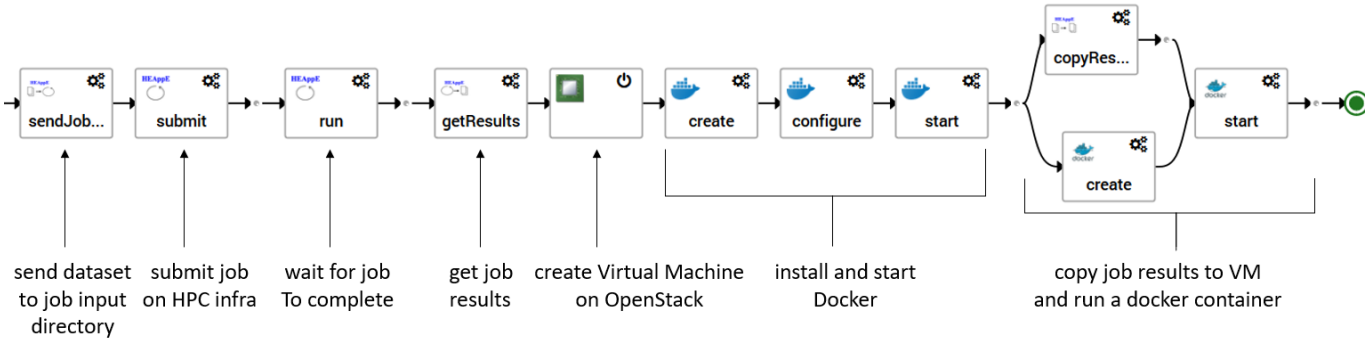

Figure 2 Screenshot of the Hybrid HPC/Cloud Example Workflow in Alien4Cloud UI

SOURCE CODE	https://github.com/lexis-project/application-templates/tree/master/sample
LICENSE	Apache 2.0
DOCUMENTATION	https://github.com/lexis-project/application-templates/blob/master/sample/README.md

Table 7 Example of Hybrid HPC/Cloud Deployment - Source Code

3 INSTALLATION ON LEXIS INFRASTRUCTURES

This intermediate version of the LEXIS Orchestration system has been installed similarly on both centres at IT4I and LRZ. One Virtual Machine was allocated at each of these centres to host the front-end A4C and the Ystia Orchestrator with the plugins described above.

This was done using Yorc bootstrap command-line, which allowed the deployment of the full orchestration system as described in the Yorc documentation at [5]. The following installation options were selected for this intermediate version: no HA deployment (one instance of each components, all installed on a single Virtual Machine), secured deployment (TLS with mutual authentication between components, HashiCorp Vault is installed to store infrastructures credentials).

A4C was then configured to delegate its user authentication to Keycloak, a software providing Single Sign-on with Identity and Access Management, supporting standard protocols Open ID Connect, OAuth 2, SAML 2.

A new client (Service Provider) was configured in Keycloak for A4C. This client is using SAML protocol, as this is the only protocol currently supported by A4C.

Three users were created, each having a distinct role:

- *administrator*, who can configure which resources which user can create on each infrastructure,
- *architect*, who can add TOSCA components in A4C catalog but can't deploy applications,
- *appmanager*, who can't add new components in A4C catalog, but can create new applications from these components, deploy/undeploy applications and run workflows.

The roles for these users were duplicated in A4C by the administrator as A4C can delegate user authentication to Keycloak but cannot currently map roles defined in Keycloak.

Then the Yorc HEAppE plugin was added in Yorc plugins directory and Yorc was restarted, to add the support of HEAppE to Yorc. Yorc CLI was then used to add new locations (infrastructures) so that Yorc was able to deploy applications on the OpenStack setup and run/monitor jobs on the HPC infrastructures through HEAppE.

TOSCA components and applications template from the example described in Section 2.7 above were added to the A4C catalogue of TOSCA components (see the application template TOSCA description [6]), so that users can create an application from this template, deploy it, and run a workflow on OpenStack and HPC using this orchestration service intermediate version.

4 SUMMARY

This deliverable describes the intermediate version of the orchestration system allowing to run hybrid Cloud/HPC workflows on LRZ and IT4I infrastructures.

It is based on the open source versions of the orchestrator back-end Yorc and front-end Alien4Cloud, provided by Bull/Atos.

A Yorc plugin was developed under the LEXIS project, to extend the orchestrator so that it supports the management of HPC infrastructure resources through the middleware developed by IT4I (HEAppE). A Go client was also developed to provide an API through which the LEXIS Portal can interact with the orchestration system.

This intermediate version was installed at IT4I and LRZ, allowing to deploy applications and run workflows on LEXIS Cloud infrastructure and HPC infrastructures on LRZ and IT4I.

The development and tests of workflows implementing Pilot use cases have started on this platform.

Next steps will consist in having these workflows fully implemented, integrating new features when they will be available, like the DDI API to manage data.

Requirements on orchestration as described in deliverables D4.3 [3] and D4.4 [4] will also be implemented, like the dynamic orchestration, where placement policies will provide the best location to allocate an infrastructure resource just before it is needed in a workflow.

The final version of the orchestration system will be provided with deliverable D4.6 Design and Implementation of the HPC-Federated Orchestration System – Final [2], due in month 24.

REFERENCES

[1] „OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA),“ 2020. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca.

[2] LEXIS Deliverable *D4.6 Design and Implementation of the HPC-Federated Orchestration System – Final.*

[3] LEXIS Deliverable *D4.3 Definition of data access priority, analytics policies and security assessment.*

[4] LEXIS Deliverable *D4.4 Definition of workload management policies in federated cloud/HPC environments.*

[5] „Yorc Bootstrap documentation,“ 2020. [Online]. Available: https://yorc.readthedocs.io/en/latest/bootstrap.html.

[6] „Application template TOSCA description,“ 2020. [Online]. Available: https://github.com/lexis-project/application-templates/blob/master/sample/testapp_template.yaml.