



# Large-scale EXecution for Industry & Society

## Deliverable D4.6

### Design and Implementation of the HPC-Federated Orchestration System - Final



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

<b>DELIVERABLE ID   TITLE</b>	D4.6   Design and Implementation of the HPC-Federated Orchestration System - Final
<b>RESPONSIBLE AUTHOR</b>	Laurent Ganne (Atos)
<b>WORKPACKAGE ID   TITLE</b>	WP4   Orchestration and Secure Cloud/HPC Services Provisioning
<b>WORKPACKAGE LEADER</b>	LINKS
<b>DATE OF DELIVERY (CONTRACTUAL)</b>	30/09/2021
<b>DATE OF DELIVERY (SUBMITTED)</b>	11/10/2021
<b>VERSION   STATUS</b>	V1.1   Final
<b>TYPE OF DELIVERABLE</b>	DEM (Demonstrator)
<b>DISSEMINATION LEVEL</b>	PU (Public)
<b>AUTHORS (PARTNER)</b>	Atos, LINKS
<b>INTERNAL REVIEW</b>	Thierry Goubier (CEA); Václav Svatoň (IT4Innovations)

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

## DOCUMENT VERSION

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
<b>0.1</b>	First version	15/06/2021	Laurent Ganne (Atos)
<b>0.2</b>	Review comments New Orchestration Service figure	23/06/2021	François Exertier (Atos), Laurent Ganne (Atos) Alberto Scionti (LINKS)
<b>0.3</b>	Updated sections 2.9 and 3.2 on Dynamic Allocation Module	23/08/2021	Giacomo Vitali (LINKS), Alberto Scionti (LINKS)
<b>0.4</b>	Review comments	24/08/2021	Thierry Goubier (CEA)
<b>1.0</b>	Review comments Updated executive summary Update section 1 and figure 1 Added section 2 on Demonstrator	29/09/2021	Václav Svatoň (IT4I) Alberto Scionti (LINKS) Alberto Scionti (LINKS) Laurent Ganne (Atos)
<b>1.1</b>	Final check of the deliverable	10/10/2021	Katerina Slaninova (IT4I)

## GLOSSARY

ACRONYM	DESCRIPTION
<b>ALIEN4CLOUD</b>	Application Lifecycle ENablement for Cloud
<b>A4C</b>	Alien4Cloud
<b>ADMS</b>	Atmospheric Dispersion Modelling System
<b>API</b>	Application Programming Interface
<b>CLI</b>	Command Line Interface
<b>DAM</b>	Dynamic Allocation Module
<b>DDI</b>	Distributed Data Infrastructure
<b>FPGA</b>	Field Programmable Gate Array
<b>GPU</b>	Graphics Processing Unit
<b>HEAPPE</b>	High-End Application Execution Middleware
<b>HPC</b>	High Performance Computing
<b>OIDC</b>	OpenID Connect
<b>REST API</b>	Application Interface that uses HTTP requests to GET, PUT, POST and DELETE data
<b>TLS</b>	Transport Layer Security
<b>TOSCA</b>	Topology and Orchestration Specification for Cloud Applications
<b>UI</b>	User Interface

<b>VPN</b>	Virtual Private Network
<b>VM</b>	Virtual Machine
<b>YORC</b>	Ystia orchestrator

## TABLE OF PARTNERS

<b>ACRONYM</b>	<b>PARTNER</b>
Avio Aero	GE AVIO SRL
Atos	BULL SAS
AWI	ALFRED WEGENER INSTITUT HELMHOLTZ ZENTRUM FUR POLAR UND MEERESFORSCHUNG
BLABS	BAYNCORE LABS LIMITED
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
EURAXENT	MARC DERQUENNES
GFZ	HELMHOLTZ ZENTRUM POTSDAM DEUTSCHESGEOFORSCHUNGSZENTRUM GFZ
ICHEC	NATIONAL UNIVERSITY OF IRELAND GALWAY / Irish Centre for High-End Computing
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

<b>EXECUTIVE SUMMARY .....</b>	<b>6</b>
<b>1 ORCHESTRATION SYSTEM OVERVIEW .....</b>	<b>7</b>
<b>2 DEMONSTRATOR .....</b>	<b>9</b>
2.1 WORKFLOW EXECUTION FROM LEXIS PORTAL .....	9
2.2 VALIDATION OF THE DEMONSTRATOR USING WP7 WORKFLOWS .....	16
2.2.1 <i>Workflows' description</i> .....	16
2.2.2 <i>Validation of workflows' execution</i> .....	20
2.3 ADVANCED FEATURE: WORKFLOW EXECUTION FAILOVER .....	21
<b>3 SOFTWARE DESCRIPTION .....</b>	<b>23</b>
3.1 ALIEN4CLOUD .....	23
3.2 ALIEN4CLOUD GO CLIENT LIBRARY .....	23
3.3 ALIEN4CLOUD YORC PROVIDER PLUGIN .....	23
3.4 YORC .....	24
3.5 YORC HEAPPE PLUGIN .....	24
3.6 YORC DDI PLUGIN .....	25
3.7 YORC DYNAMIC ORCHESTRATION PLUGIN .....	26
3.8 YORC OIDC CLIENT .....	27
3.9 DYNAMIC ALLOCATION MODULE .....	27
3.10 ORCHESTRATION SERVICE API .....	27
3.11 YSTIA FORGE .....	28
3.12 REPOSITORIES OF LEXIS TOSCA COMPONENTS AND APPLICATION TEMPLATES .....	28
3.12.1 <i>Public repository</i> .....	28
3.12.2 <i>Private repositories</i> .....	31
<b>4 INSTALLATION ON LEXIS INFRASTRUCTURES .....</b>	<b>31</b>
4.1 YSTIA INSTALLATION .....	31
4.2 DYNAMIC ALLOCATION MODULE INSTALLATION .....	31
<b>5 SUMMARY .....</b>	<b>32</b>
<b>REFERENCES .....</b>	<b>33</b>

## LIST OF FIGURES

FIGURE 1 ORCHESTRATION SYSTEM OVERVIEW .....	8
FIGURE 2 PORTAL WORKFLOWS .....	9
FIGURE 3 SELECT A WORKFLOW TEMPLATE .....	10
FIGURE 4 CREATE A WORKFLOW .....	10
FIGURE 5 WORKFLOW INFORMATION.....	11
FIGURE 6 CREATE WORKFLOW EXECUTION.....	12
FIGURE 7 WORKFLOW EXECUTION RUNNING .....	12
FIGURE 8 WORKFLOW EXECUTION DETAILS.....	13
FIGURE 9 WORKFLOW EXECUTION PROGRESS .....	14
FIGURE 10 WORKFLOW EXECUTION LOGS .....	14
FIGURE 11 WORKFLOW EXECUTION OUTPUTS.....	15
FIGURE 12 RESULT DATASET.....	15
FIGURE 13 RESULT DATASET CONTENTS .....	16
FIGURE 14 WP7 WORKFLOW PRE-PROCESSING PHASE .....	17
FIGURE 15 WP7 WORKFLOW HPC COMPUTATION PHASE.....	18
FIGURE 16 WP7 WORKFLOW POST-PROCESSING PHASE.....	20
FIGURE 17 LEXIS WORKFLOW RESULTS .....	20
FIGURE 18 OBSERVATIONS DATA.....	21
FIGURE 19 WORKFLOW WITH ON FAILURE STEPS.....	22

## LIST OF TABLES

TABLE 1 ALIEN4CLOUD REPOSITORY .....	23
TABLE 2 ALIEN4CLOUD GO CLIENT LIBRARY REPOSITORY .....	23
TABLE 3 ALIEN4CLOUD YORC PROVIDER PLUGIN REPOSITORY .....	24
TABLE 4 YORC REPOSITORY .....	24
TABLE 5 YORC HEAPPE PLUGIN REPOSITORY .....	24
TABLE 6 YORC DDI PLUGIN REPOSITORY .....	25
TABLE 7 YORC DYNAMIC ORCHESTRATION PLUGIN REPOSITORY.....	26
TABLE 8 YORC OIDC CLIENT REPOSITORY .....	27
TABLE 9 DYNAMIC ALLOCATION MODULE REPOSITORY .....	27
TABLE 10 ORCHESTRATION SERVICE API REPOSITORY .....	28
TABLE 11 YSTIA FORGE REPOSITORY .....	28
TABLE 12 PUBLIC REPOSITORY .....	29

## 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, Alien4Cloud [2] 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 using either a UI or a REST API.

### 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 and updates the deliverable D4.2 Design and Implementation of the HPC-Federated Orchestration System – Intermediate [3].

It enables the execution of workflows from WP8 LEXIS Portal, for WP5 (Aeronautics), WP6 (Earthquake and Tsunami) and WP7 (Weather and Climate) pilots, on Cloud/HPC infrastructures described in WP2 (LEXIS Requirements Definition and Architecture Design).

### Description of the deliverable

The purpose of this deliverable is twofold: on the one hand, it describes how the LEXIS Orchestration Service looks like in its final stage of implementation, by describing the software basic blocks that are at its basis and their installation on the LEXIS infrastructure; on the other hand, it provides a practical example (that is representative of the pilot application workflows served by the LEXIS platform) of its usage in a concrete context. Through this example, this deliverable clearly demonstrates the capability of the LEXIS Orchestration Service to manage the execution of complex workflows using federated resources (HPC and Cloud).

The Orchestration System is based on open-source software: the orchestrator back-end Yorc [4] and front-end AlienCloud [2], provided by Atos. Additional components were developed for the LEXIS project:

- Dynamic Allocation Module (DAM) developed by LINKS to implement policies of infrastructure resources placement as described in LEXIS Deliverables D4.3 [5] and D4.4 [6],
- Orchestrator back-end extensions developed by Atos to interact with:
  - WP3 LEXIS DDI API (Distributed Data Infrastructure API) for data management, described in Deliverable D3.3 [7],
  - IT4I HEAppE (High-End Application Execution Middleware) [8] to submit and monitor jobs on HPC infrastructures,
  - DAM, to dynamically set the locations where to allocate infrastructure resources during the workflow execution according to placement computations done by DAM.
- Orchestration Service API developed by ICHEC, used in the WP8 LEXIS Portal back-end to interact with the Orchestration System,
- Front-end Go client library developed by Atos, through which the Orchestration Service API can interact with Alien4Cloud.

Repositories of TOSCA components and application templates implementing Pilot workflows are also provided for:

- WP5 Aeronautics workflows - collaboration of Avio Aero, UNIFI, Atos and IT4I,
- WP6 Earthquake and Tsunami - collaboration of CEA, Atos and IT4I,
- WP7 Weather and Climate workflows - collaboration of CIMA, ECMWF, Atos and IT4I.

## 1 ORCHESTRATION SYSTEM OVERVIEW

The LEXIS project aims at providing the capability of executing complex application workflows that mix HPC, Big Data and Cloud processing, through an effective execution platform. Starting from this, the LEXIS platform has been designed around a set of services that allow the users to securely log-in, define their workflows and execute them on diverse infrastructural resources. As such, one of the key enablers is represented by the LEXIS Orchestration Service, which has been based on a set of technological building blocks. The LEXIS Orchestration Service has been designed in such a way that computational and storage resources (geographically) located on different HPC centres could be seamlessly used to efficiently (i.e., using the most appropriate set of resources that are available) execute users' workflows. Computational resources comprise those available through HPC clusters, virtual cloud instances (i.e., VMs), GPU accelerated systems (Nvidia DGX-2 system, accelerated nodes in Karolina cluster) and FPGAs. Similarly, storage resources are served through the DDI Service and comprise also the access to Burst Buffer nodes. The role of the LEXIS Orchestration Service is that of coordinating the access to these resources in order to execute users' workflows (thus, also Pilot workflows) by satisfying all their specific execution requirements. Also, maintenance period of the resources is taken into account to provide a reliable set of resources where to execute the workflow tasks, as well as the location of the datasets and the data transfer speeds between the locations. Finally, reliable execution is also supported by the capability of the LEXIS Orchestration Service to keep track of the execution failures. The LEXIS Orchestration Service makes decisions on coordinating the access to the resources keeping their usage balanced.

In order to provide all these features, the LEXIS Orchestration Service required a way to enable the dynamic allocation of the resources. To this end, DAM has been designed and successfully integrated with the other components. DAM is a Python-based module that receives the requests for assigning a location for job execution from the Yorc module and provides the most suited one to use. DAM is built upon the Flask framework and provides a REST-API to interact with. The computed locations are ranked in such a way there is a priority value (assigned to each location) in selecting which one to use first. Ranking the locations means to assign a priority level to each available location, and it allows DAM to easily support a failover mechanism: if a location failed in executing a workflow job for a certain number of times, DAM can provide another location with a lower priority. To this end, DAM evaluates each available location on the basis of a set of criteria; the evaluations of such criteria are then combined together to derive the final priority value. Deliverable D4.4 [6] provides a detailed description of the criteria used by DAM to compute this priority value and the way single evaluations are combined together. To have a better use of the resources that are assigned to a given user, DAM also uses information coming from the accounting and billing service for balancing purposes. Similarly, information regarding the status of the clusters (mainly provided by HEAppE middleware and OpenStack) are used for ranking the locations available.

An important information for making proper decision is represented by maintenance periods. Since HPC clusters are complex machines, it happens that sometimes there is a scheduled maintenance period, during which the machine is not accessible. As such, DAM exposes an API that allows to register the planned maintenance periods in a database. During the evaluation process of the locations, DAM excludes those ones that will go under maintenance if the maintenance period is overlapped with that of the execution of the workflow job. Elapsed maintenance period is also automatically removed in order to avoid the database growing too much. Similarly, DAM provides an API for updating the data transfer speeds between location pairs. These speed values (measured in bytes/s) are indeed not constant over the time and periodically can be adjusted to reflect the real performance of the networking.

The LEXIS Orchestration Service has been designed around the above-mentioned key points (i.e., supporting resource federation, managing failover, balancing the resource usage, keep tracking of execution failures, etc.), and it integrates the following technological building blocks (Figure 1 depicts the way LEXIS Orchestration Service components are architected):

- Orchestrator Service API used by the LEXIS Portal to interact with the Orchestration System,
- Internal API, Go client library used by the Orchestration Service API to interact with the Orchestrator front-end Alien4Cloud,
- 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 to the front-end A4C the ability to use the Ystia orchestrator (Yorc) to manage applications lifecycle and run workflows,
- DAM allowing to select locations where to allocate infrastructure resources,
- Yorc, the Ystia orchestrator managing the applications lifecycle and workflows executions,
- Yorc plugins, extending Yorc so it can use HEAppE as the framework managing HPC infrastructures in LEXIS, DDI to manage data transfers, and DAM to select dynamically the locations where to allocate infrastructure resources during the workflow execution.

In addition, the Ystia Forge, a collection of TOSCA components and application templates, provides TOSCA components used in Pilot workflows: Docker component, Docker container, Docker image, etc..

TOSCA components and application templates written for Pilot workflows to be executed on LEXIS Cloud and HPC infrastructure through HEAppE and OpenStack are also provided.

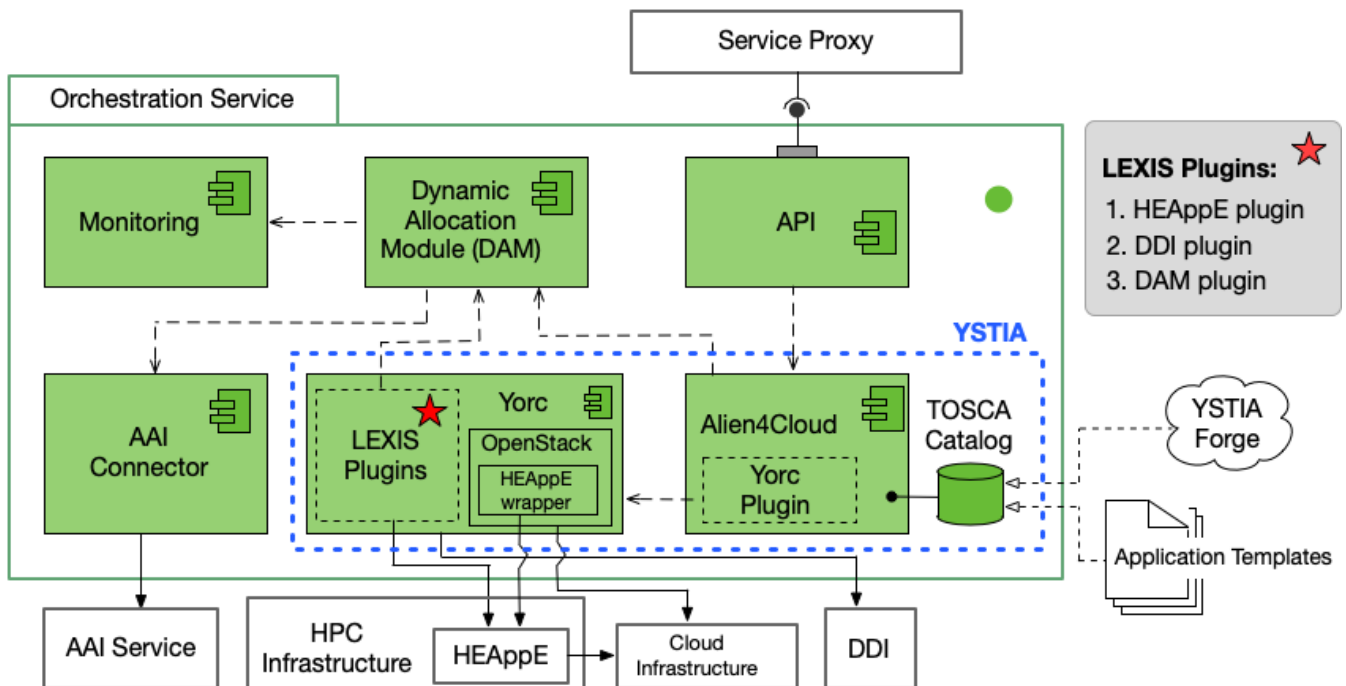


Figure 1 Orchestration System overview



## 2 DEMONSTRATOR

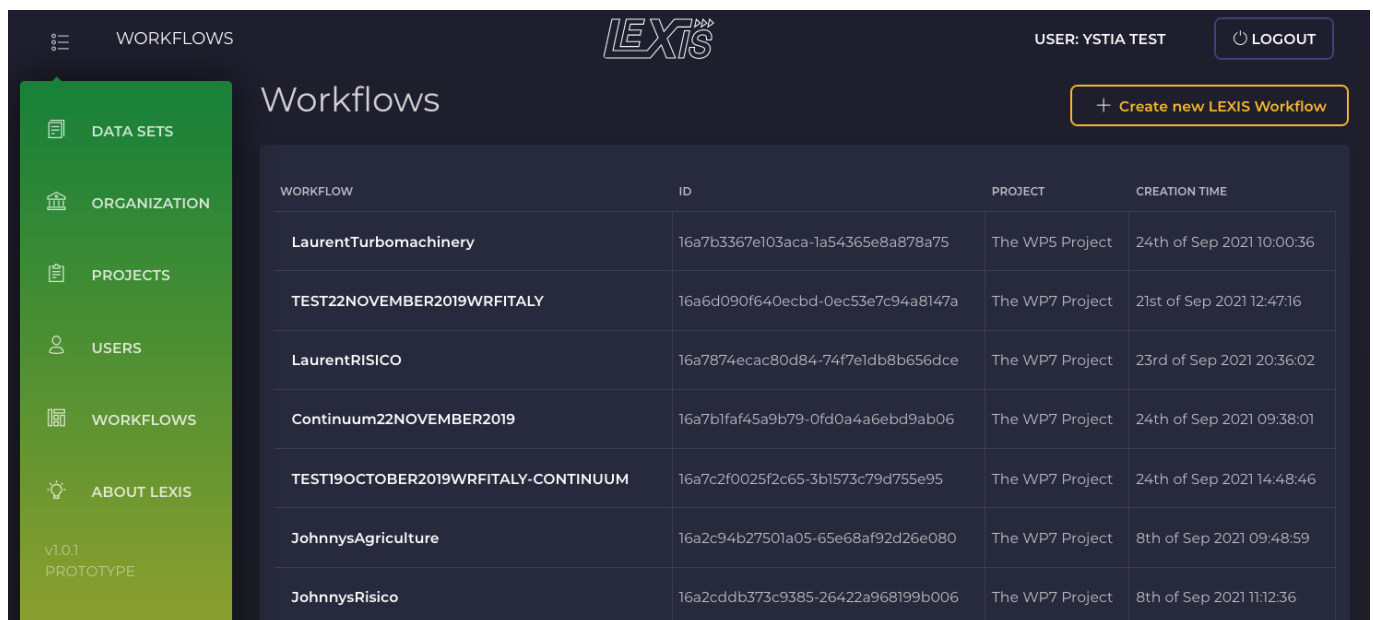
The demonstrator allows to execute workflows in a federated Cloud/HPC environment made of Cloud locations (OpenStack) and HPC locations (available through HEAppE) on both IT4I and LRZ, and using the LEXIS Distributed Data Infrastructure to manage data transfers between these locations.

This section describes how to execute and monitor a workflow from LEXIS Portal. It provides then an overview of the workflows that were executed to validate the demonstrator, and finally describes a more advanced feature to manage workflow executions failover.

### 2.1 WORKFLOW EXECUTION FROM LEXIS PORTAL

Workflows can be launched programmatically using the Orchestration Service API or can be launched from the User Interface provided by the LEXIS Portal.

From LEXIS Portal, the user needs first to login. He can then select **Workflows** on the left-hand pane to see the list of workflows already created, see Figure 2:



WORKFLOW	ID	PROJECT	CREATION TIME
LaurentTurbomachinery	16a7b3367e103aca-1a54365e8a878a75	The WP5 Project	24th of Sep 2021 10:00:36
TEST22NOVEMBER2019WRFITALY	16a6d090f640ecbd-0ec53e7c94a8147a	The WP7 Project	21st of Sep 2021 12:47:16
LaurentRISICO	16a7874ecac80d84-74f7e1db8b656dce	The WP7 Project	23rd of Sep 2021 20:36:02
Continuum22NOVEMBER2019	16a7b1faf45a9b79-0fd0a4a6ebd9ab06	The WP7 Project	24th of Sep 2021 09:38:01
TEST19OCTOBER2019WRFITALY-CONTINUUM	16a7c2f0025f2c65-3b1573c79d755e95	The WP7 Project	24th of Sep 2021 14:48:46
JohnnysAgriculture	16a2c94b27501a05-65e68af92d26e080	The WP7 Project	8th of Sep 2021 09:48:59
JohnnysRisico	16a2cddb373c9385-26422a968199b006	The WP7 Project	8th of Sep 2021 11:12:36

Figure 2 Portal workflows

When the user clicks on **Create new LEXIS Workflow** on the top right corner, he is presented with a list of workflow templates that he can instantiate, see Figure 3.

Users have access to common workflow templates, like Computational Fluid Dynamics simulation (OpenFOAM computation and visualization of results in Paraview), or generic workflow templates to perform basic cloud computation through a script or a public docker images to run provided in input by the user, HPC computation providing in input the HEAppE identifier of the computation to run, or a generic workflow combining cloud and HPC computations.

The user is also presented workflow templates associated to the LEXIS project he is allowed to use.

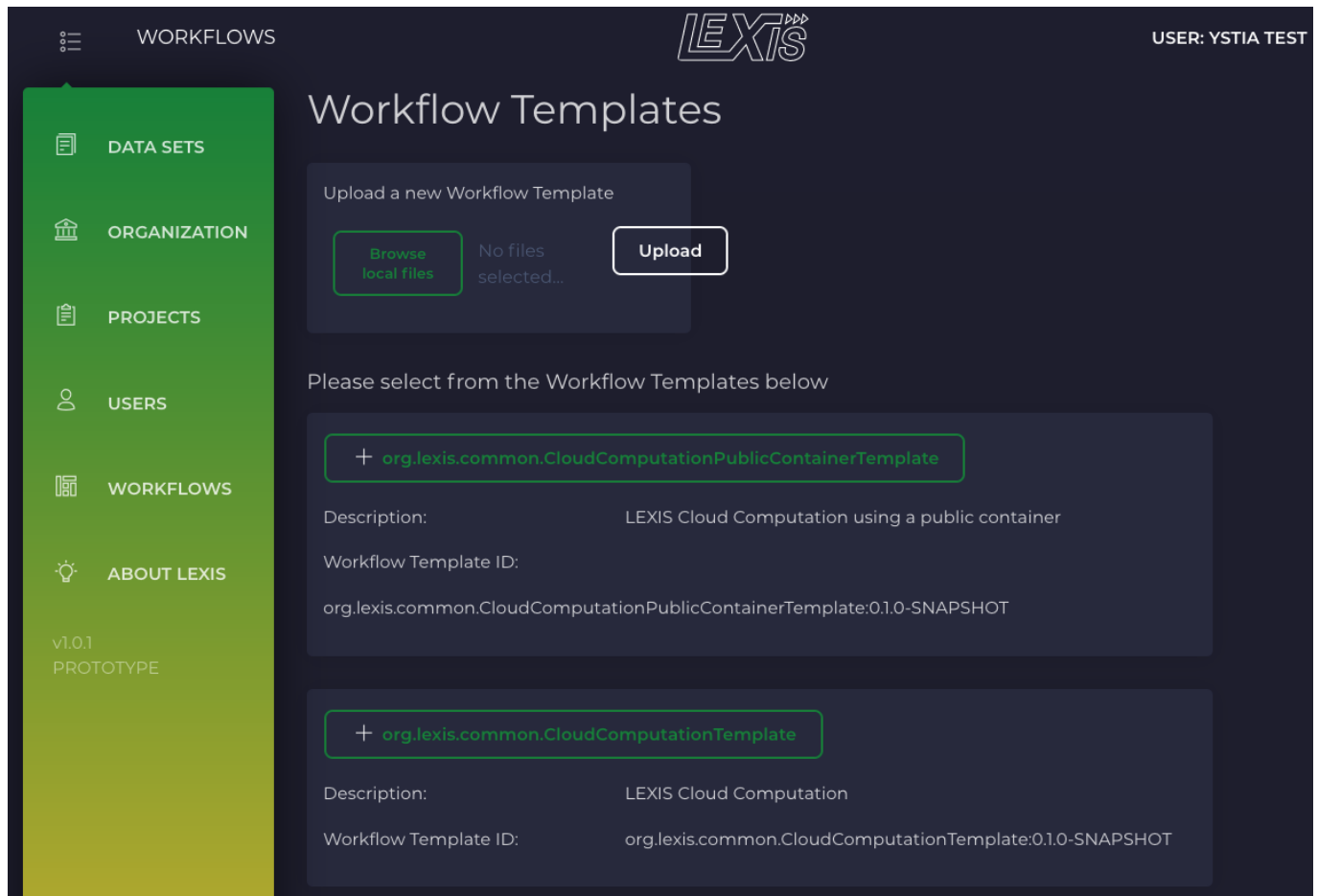


Figure 3 Select a Workflow template

Once the user has selected a template, he is asked to specify a name, the project under which he wants to instantiate this workflow template and a description, see Figure 4:

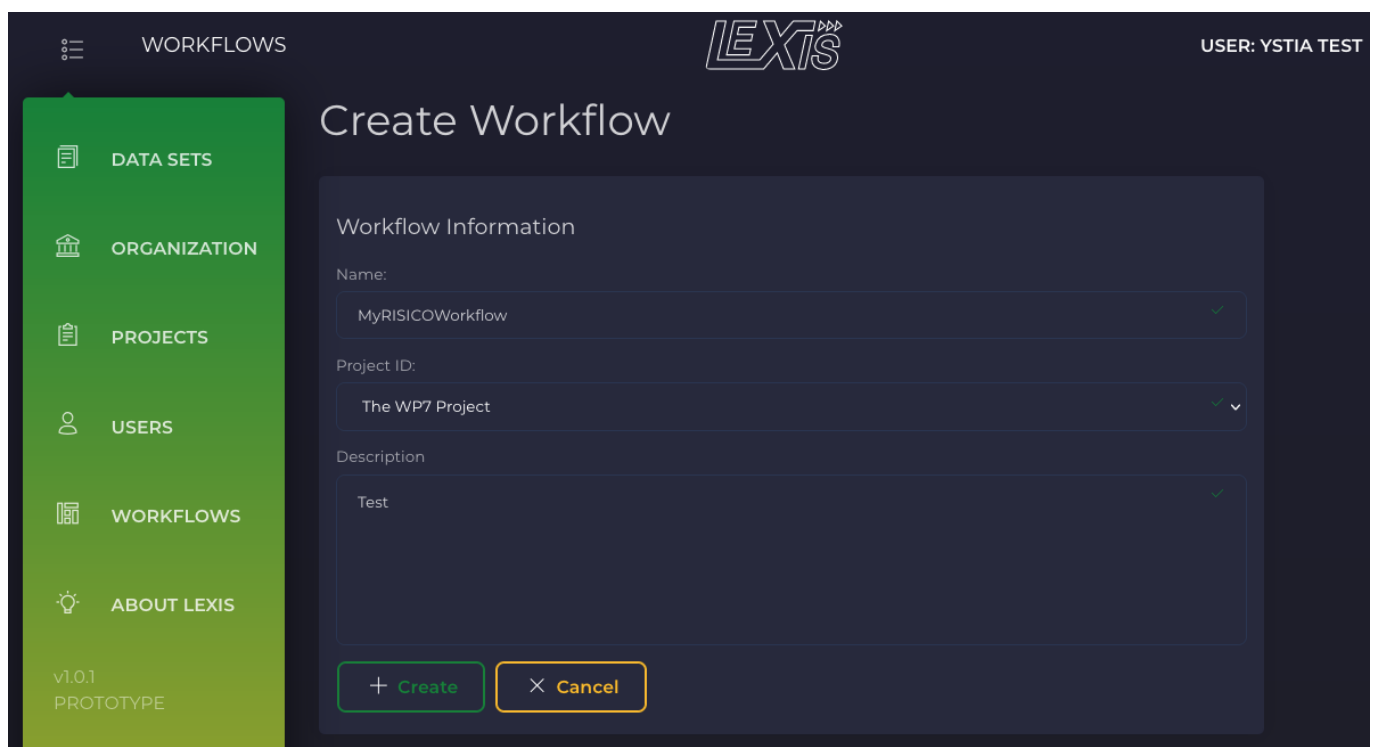


Figure 4 Create a Workflow

Once done, clicking on **Create** creates a workflow from the selected template, and a page providing a short description of the workflow appears (see Figure 5):

The screenshot shows the LEXIS interface for a workflow named 'MyRISICOWorkflow'. The left sidebar contains navigation links: DATA SETS, ORGANIZATION, PROJECTS, USERS, WORKFLOWS (highlighted), and ABOUT LEXIS. The main content area displays 'Workflow Information' and 'Workflow Tasks'. At the bottom, there are three buttons: '+ Create Workflow Execution', 'Remove Workflow', and 'View Workflow Executions'.

Workflow Information	
Workflow Name:	MyRISICOWorkflow
Workflow ID:	16a89ac2925e8ca3-578e266c6420bf56
Project:	The WP7 Project
Workflow Template:	org.lexis.wp7.RisicoTemplate:0.1.0-SNAPSHOT
Description:	Test
Created By:	a59ce88b-9448-4c27-ad1e-412b1e4ecf56
Time Created:	27th of Sep 2021 08:43:45

Workflow Tasks	
preprocessing:	WPS_GFS    HPCToDDIJob
computation:	WRF
postprocessing:	RISICO

Figure 5 Workflow information

Clicking on **Create Workflow Execution** will display a page where the user can provide input parameters for the workflow execution to run, see Figure 6:

Figure 6 Create Workflow execution

For example, in Figure 6 we see input parameters for a WP7 workflow:

- Geographical data Path is the path to a dataset in DDI providing geographical data, it has a default value in the workflow template, and the user can select another dataset in DDI through a drop-down list showing the datasets in DDI to which he has access,
- Public container images used to do the pre-processing and download observations data, with default values,
- Start date, with no default value, that the user must provide to specify the date for which the workflow will get meteo data and compute a simulation.

Once done, the user clicks on **Run** at the end of the page (cropped in the screenshot in Figure 6) to run the workflow execution.

A page appears with the new workflow execution shown as currently running, see Figure 7. The user can click on a workflow execution to get more details.

WORKFLOW EXECUTION	WORKFLOW	CREATION TIME	STATUS
0-wp7	MyRISICOWorkflow	27th of Sep 2021 08:48:03	Running

Figure 7 Workflow execution running

After having clicked on the workflow execution, the page with three tabs at the top, **DETAIL**, **PROGRESS** and **LOGS** appears, see Figure 8.

The **DETAIL** tab provides the current status of the workflow execution, here **Running**), workflow inputs, as well as workflow outputs when they will be computed:

Figure 8 Workflow execution details

The **PROGRESS** tab shows a graphical view of the workflow execution. Steps in green have been executed successfully, steps in red have failed, steps in blue are in progress, steps in black are not yet done.

In Figure 9, a WP7 workflow is executing in parallel:

- Step **WPS\_GFS\_create**, downloading a docker image and creating the container that will perform the pre-processing,
- Step **GFSData\_start**, downloading Global Forecast System meteo data for the date specified in input from an external site,
- Step **DDIToCloudGEOGDatasetJob\_run**, monitoring the asynchronous execution of the staging of an input dataset from DDI to the cloud staging area, performed by the DDI staging API:

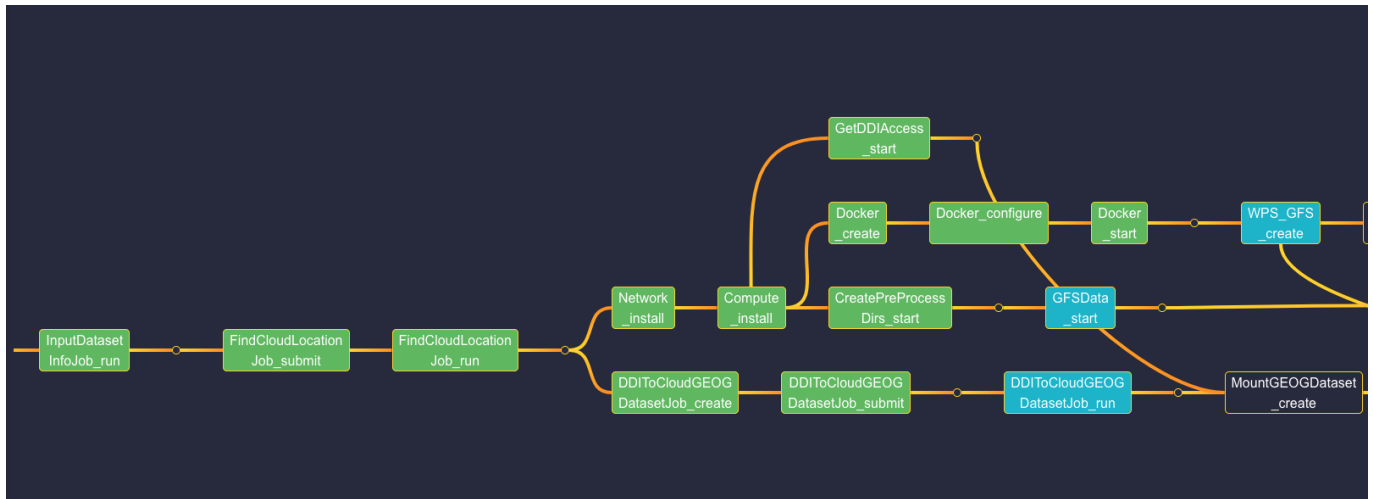


Figure 9 Workflow execution progress

The **LOGS** tab provides workflow execution logs of the orchestrator back-end, see Figure 10. Here, logs show the creation of an OpenStack instance on the location selected previously in the workflow by DAM:

TIME	LOG MESSAGE CONTENT
2021-09-27 08:50:25	openstack_compute_floatingip_associate_v2.FIPyorc-RisicoVM-0: Creation complete after 3s (ID: 138.246.236.9/130b1e16-8e75-43ce-b528-c7098a15a4ff) null_resource.yorc-RisicoVM-0-ConnectionCheck: Creating... null_resource.yorc-RisicoVM-0-ConnectionCheck: Provisioning with 'remote-exec'... null_resource.yorc-RisicoVM-0-ConnectionCheck (remote-exec): Connecting to remote host via SSH... null_resource.yorc-RisicoVM-0-ConnectionCheck (remote-exec): Host: 138.246.236.9 null_resource.yorc-RisicoVM-0-ConnectionCheck (remote-exec): User: ubuntu null_resource.yorc-RisicoVM-0-ConnectionCheck (remote-exec): Password: false null_resource.yorc-RisicoVM-0-ConnectionCheck (remote-exec): Private key: true null_resource.yorc-RisicoVM-0-ConnectionCheck (remote-exec): SSH Agent: false null_resource.yorc-RisicoVM-0-ConnectionCheck (remote-exec): Checking Host Key: false
2021-09-27 08:50:20	openstack_compute_instance_v2.yorc-RisicoVM-0: Still creating... (40s elapsed) openstack_compute_instance_v2.yorc-RisicoVM-0: Creation complete after 44s (ID: 130b1e16-8e75-43ce-b528-c7098a15a4ff) openstack_compute_floatingip_associate_v2.FIPyorc-RisicoVM-0: Creating... floating_ip: "" => "138.246.236.9" instance_id: "" => "130b1e16-8e75-43ce-b528-c7098a15a4ff" region: "" => "RegionOne"
2021-09-27 08:50:10	openstack_compute_instance_v2.yorc-RisicoVM-0: Still creating... (30s elapsed)
2021-09-27 08:50:03	Status for node "DDIToCloudGEOGDatasetJob", instance "0" changed to "PENDING"

Figure 10 Workflow execution logs

When the workflow is done, back to the **DETAIL** tab, this tab provides outputs of the workflow, see Figure 11. In most cases, these outputs are links to result datasets.

For workflows with a visualization phase, like the OpenFOAM workflow, a URL is provided to access a remote session on a Cloud Instance and visualize results.

Below in the **DETAIL** tab, the section **Output Properties** provides the unique identifier of two result datasets on which you can click:

project\_id: /wps\_data/output/inputs/output /mnt/cloud\_staging\_area/risico: /geogrid 5f248333-a8d7-62af-4484-ef763ff2b331

Workflow Tasks

preprocessing: HPCToDDIJob WPS\_GFS

computation: WRF

postprocessing: RISICO

Output Properties

HPCToDDIJob: 90056ba2-1d01-11ec-a2ff-b8599f568450

CloudToDDIJob: bf2d41d8-1d02-11ec-bf8c-0050568fc9b5

[Delete](#) [Back](#) [Recreate workflow](#)

Figure 11 Workflow execution outputs

Clicking on the first Unique Identifier, a page describing the dataset appears, see Figure 12:

Data Set Detail: *RISICO Workflow WRF result - 2019101100* [Update dataset metadata](#)

Data Set ID:

Data Set Access Mode: project

Project: The WP7 Project

Data Publication Year: 2021

Data Creator(s): RISICO workflow

Encryption: no

Compression: yes

[Go to dataset list and refresh](#) [Advanced](#) [List files](#)

Figure 12 Result dataset

Clicking on **List files** at the end of the page allows to check the content of the dataset, see Figure 13:

The screenshot displays the LEXIS web interface. At the top, there is a navigation bar with the LEXIS logo, a user profile 'USER: YSTIA TEST', and a 'LOGOUT' button. A sidebar on the left contains menu items: DATA SETS, ORGANIZATION, PROJECTS, USERS, WORKFLOWS, and ABOUT LEXIS. The main content area is titled 'RISICO Workflow WRF result - 2019101100'. Below the title, there are three buttons: 'Refresh', 'Check size of dataset', and 'Upload file(s) here'. A table lists the dataset contents:

NAME	TYPE	CREATION TIME	SIZE		
results.tar.gz	gz	24th of Sep 2021 06:35:10	9.89 GiB	Delete	Download

At the bottom left of the sidebar, it says 'v1.0.1 PROTOTYPE'.

Figure 13 Result dataset contents

Here, the **Download** action allows to download the corresponding file.

## 2.2 VALIDATION OF THE DEMONSTRATOR USING WP7 WORKFLOWS

This section describes the WP7 workflows that were executed to validate the demonstrator, the platform on which these workflows were run and the observed behaviour regarding the execution of computations on the federated infrastructure.

### 2.2.1 Workflows' description

WP7 workflows,

- RISICO - risks of wildland fires simulations over Italy,
- Continuum - risks of flooding simulation over Italy,
- ADMS - Air quality over France.

have a similar structure made of 3 phases:

- Download and pre-processing of input data on a cloud instance,
- Computation on HPC,
- Post-processing on cloud instance,
- Upload of results to a SFTP server for analysis, in addition to the staging of results in DDI.

The pre-processing phase of each of these workflows (see Figure 14) requires a common input dataset available in DDI containing static geographical data.

The first step of the pre-processing will consist in gathering info on this dataset from the DDI API: the size of the dataset, its number of files, the locations where it is available (on one of the locations IT4I/LRZ, or on both if the dataset is replicated).

The next step is to send a request to DAM to get a location where to create a Cloud Instance for the pre-processing, providing these details on the input dataset (so that the DAM can estimate the cost of the data transfer), as well as



requirements on the Cloud instance to create for the pre-processing. These workflows require a linux cloud instance of at least 10 CPUs, 45 GB of memory and 150 GB of disk.

The DAM response will provide the URL of an OpenStack instance, OpenStack project name, as well as a HEAppE URL used to authenticate and provide OpenStack credentials to the orchestrator from the access token of the user who launched the workflow. The orchestrator will then proceed to the creation of a cloud instance on this location.

Once done, the workflow will proceed to the staging of the input dataset from DDI to a cloud staging area that will then be mounted on the cloud instance. It will also proceed to the download of data from external sites, from ECMWF for data over France, and from Research Data Archive<sup>1</sup> for input data over Italy. The pre-processing container will then be executed, and in parallel radar and sensors observation data will be downloaded from CIMA for a later use in HPC computation and post-processing.

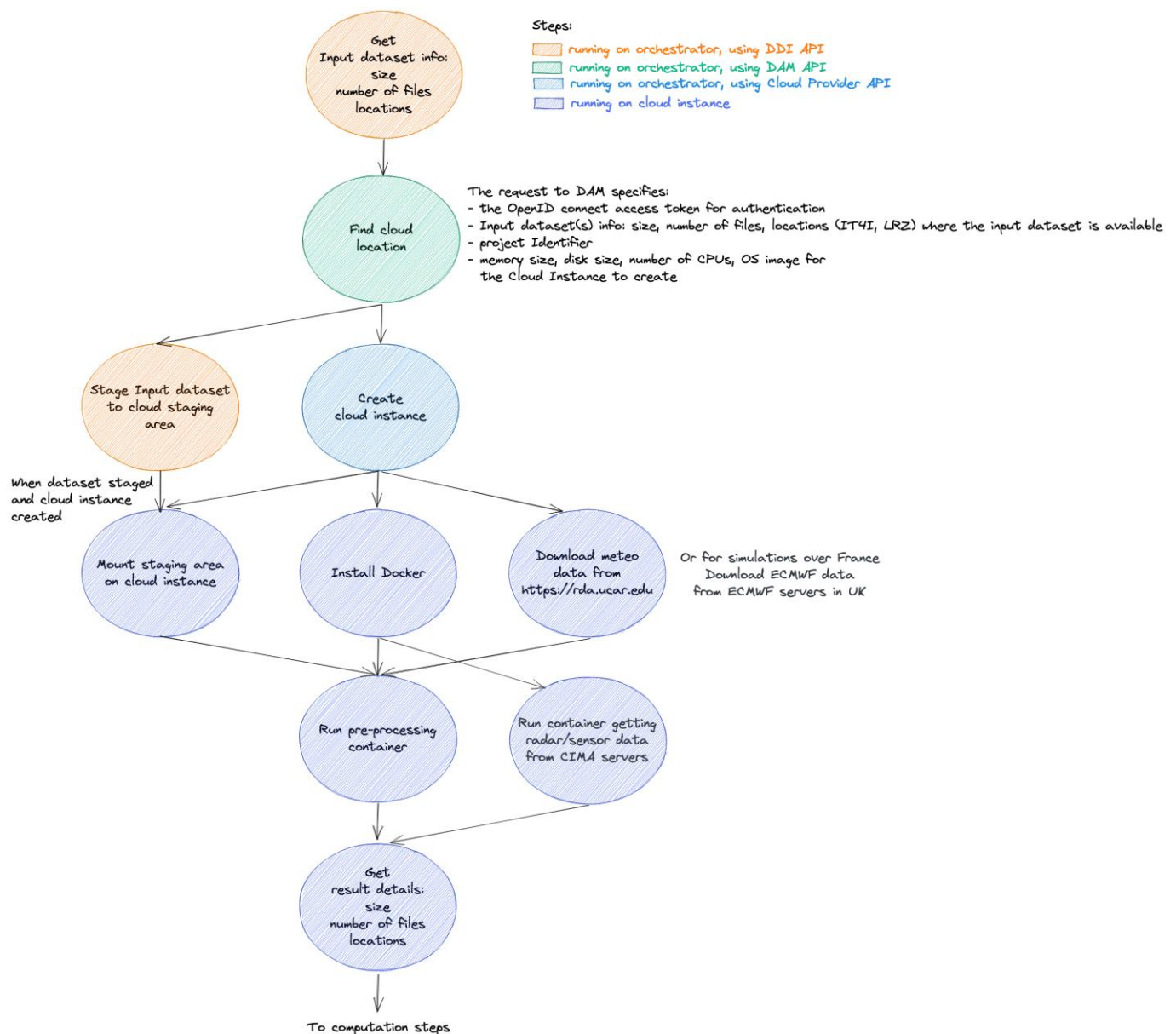


Figure 14 WP7 workflow pre-processing phase

<sup>1</sup> Research Data Archive data: <https://rda.ucar.edu>

When the pre-processing is done, the HPC computation phase starts by sending a request to DAM to find the best HPC location according to the location, size, number of files of pre-processing results, the HPC computation to perform, the number of cores and max wall time required.

The DAM response will provide the URL of a HEAppE instance, a project name, a cluster identifier and an identifier of the HPC computation to perform (WRF in the case of WP7 workflow). The orchestrator will then proceed to the creation of a HEAppE job, see Figure 15.

Once the job created, its file system is accessible to DDI for data transfers. The orchestrator will use the DDI staging API to stage pre-processing results, from the cloud staging area where they are available to a directory provided by HEAppE where the job expects to find its input data.

Once the data transfer is done, the orchestrator uses the HEAppE API to submit the job, then monitors its status until the job ends.

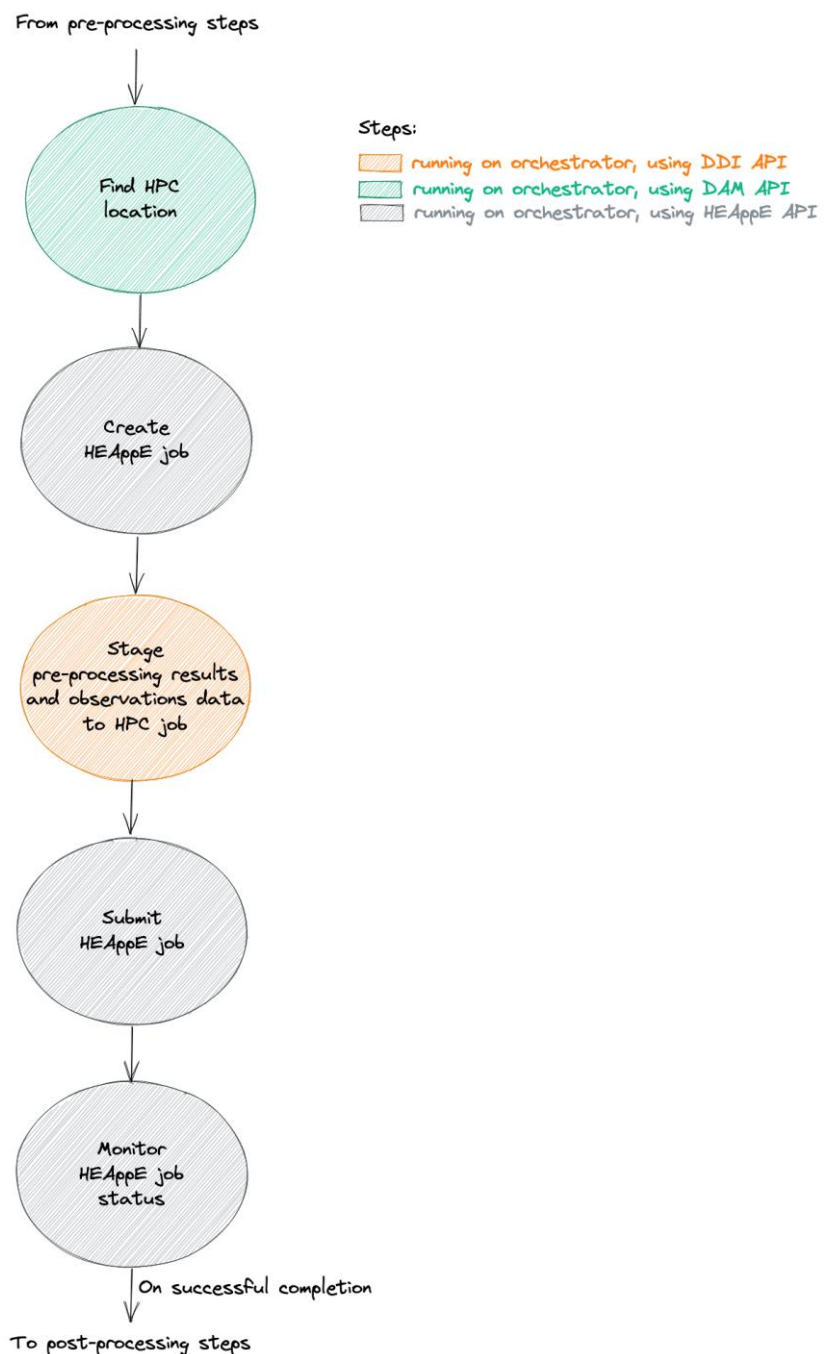


Figure 15 WP7 workflow HPC computation phase

When the HPC computation is done, results are transferred to the cloud staging and to DDI, see Figure 16.

The post-processing computation is done, and post-processing results are transferred to DDI.

These results are also uploaded to a CIMA SFTP server (and to a NUM SFTP server for ADMS - Atmospheric Dispersion Modelling System).

Finally, infrastructure resources are cleaned up: HEAppE job deleted, Cloud staging area cleaned, and Cloud instance deleted.

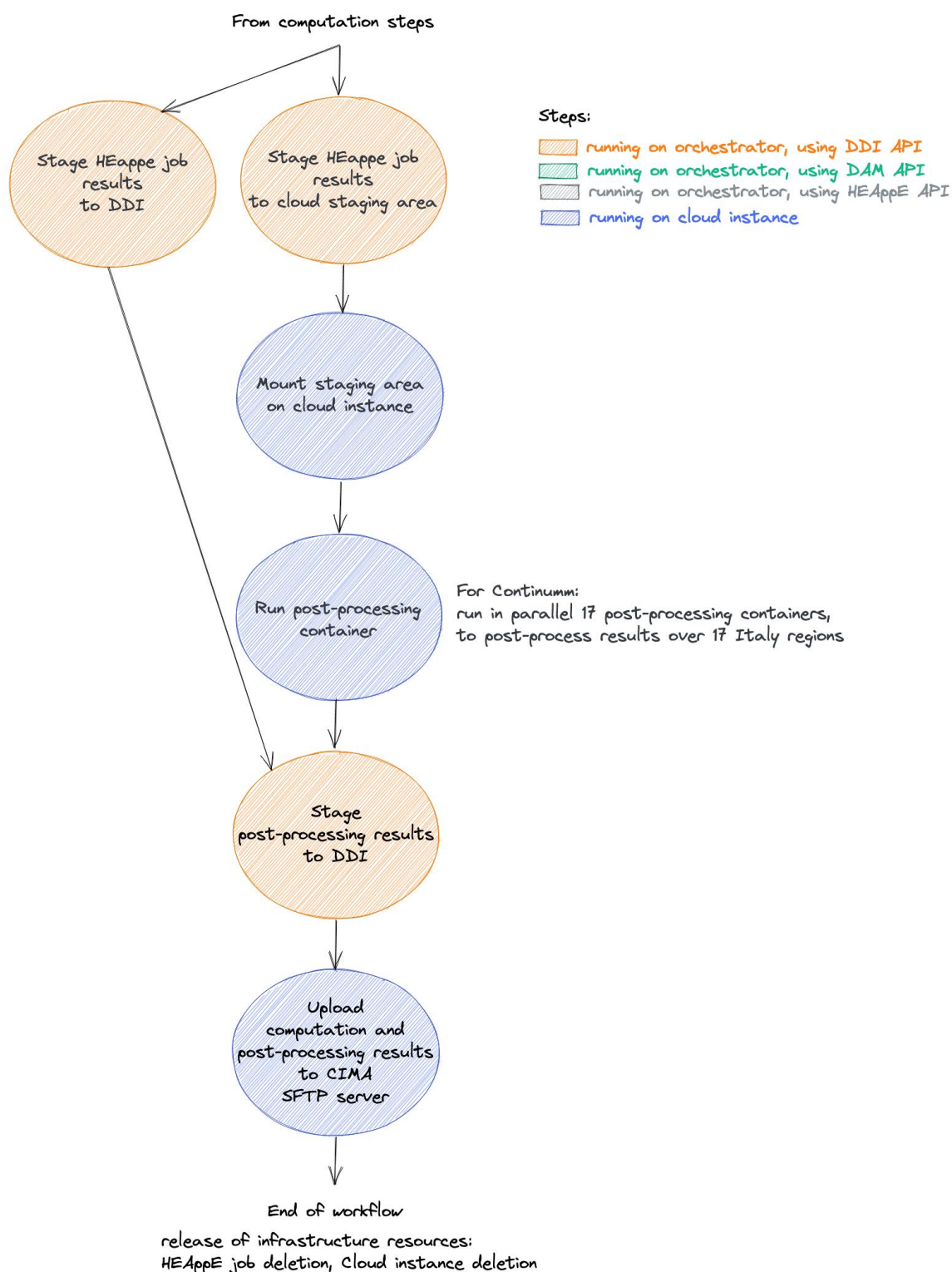




Figure 16 WP7 workflow post-processing phase

## 2.2.2 Validation of workflows' execution

WP7 workflows were executed on a platform configured to have:

- Two instances of OpenStack available for the project, one instance at IT4I and one instance at LRZ,
- The HPC computation Weather Research and Forecasting (WRF) available on two clusters in IT4I, Barбора and Karolina.

The CIMA team could run the workflows, selecting dates of remarkable known events to validate the simulation results by comparing these results with the observed data.

During the workflows' executions, we could verify that the orchestration could execute each step of the workflow as planned, on the locations selected by the DAM.

WP7 workflows use 50 GB of geographical static data stored in DDI. The use of a DDI feature allows to transfer and uncompress a compressed dataset through the internal use of a Burst Buffer, improved by 60% the time to transfer this input dataset.

The workflow executions allowed to verify that DAM selects a location to minimize the cost of the data transfer when the dataset is available on the single location, and that DAM bases its selection on the amount of free resources in the Cloud/HPC locations, when the dataset is replicated on both locations IT4I and LRZ.

The CIMA team could as well directly get the results uploaded by the workflow on their SFTP server and visualize the results on their platform myDEWETRA [9] to assess the validity of the simulation by comparing the results of the simulation to the observed data.

As an example, Figure 17 the result provided by a LEXIS WP7 Continuum Workflow execution for rainfall on October 2019, 19 to 21:

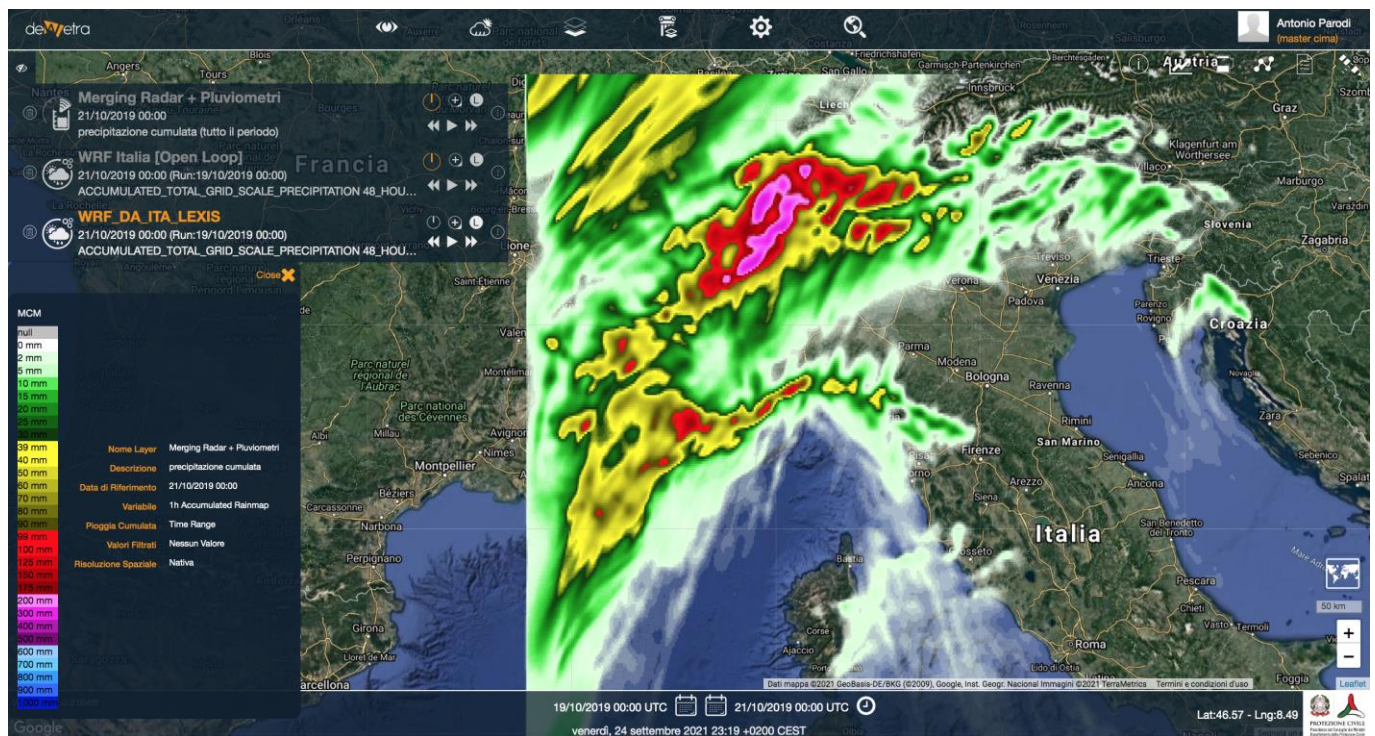


Figure 17 LEXIS workflow results

And Figure 18 shows the map for real observation data for the same dates:

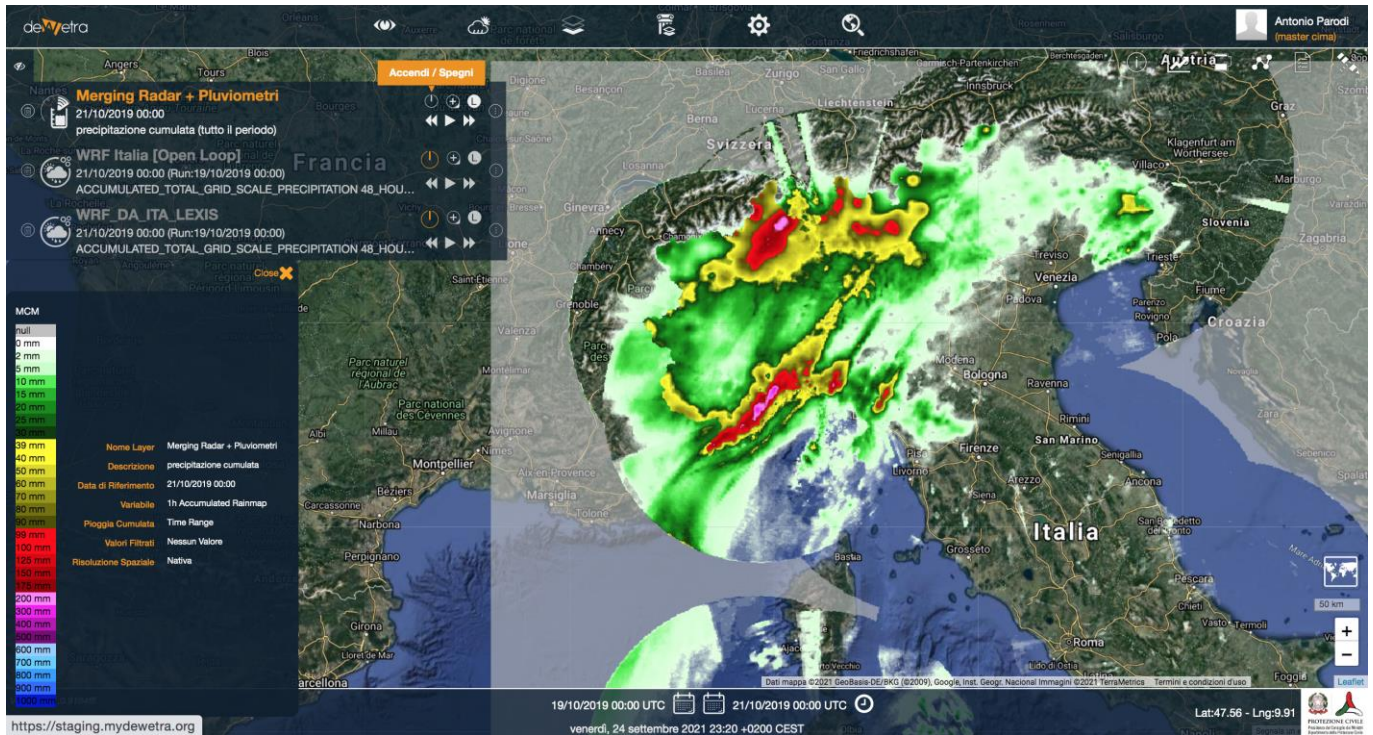


Figure 18 Observations data

### 2.3 ADVANCED FEATURE: WORKFLOW EXECUTION FAILOVER

For WP5 workflows like Turbomachinery, the HPC computation step can take several days and can benefit from a failover feature in case of failure in this step, as the software performing the HPC computation generates checkpoint files that can be used to restart from the state where it failed.

TOSCA workflows support the definition of steps to execute on failure of a given step, see Figure 19. So, steps to perform on failure of the HPC computation are defined in the workflow to manage the failover.

A TOSCA component was implemented to store in DDI the checkpoint files created by the HPC computation. For the duration of the HPC computation, this component monitors the files created by the HPC computation, that it gets from a HEAppE API endpoint providing the list of files generated by a given HPC computation job, and their last modification date. The TOSCA component identifies checkpoint files from patterns provided in parameter. And after a given elapsed time without any modification to a checkpoint file (5 minutes in the Turbomachinery case), this component considers the checkpoint file is complete and uses the DDI API to store this checkpoint file in a dataset in DDI.

Whenever a failure of the HPC computation is reported by HEAppE, the workflow will execute the “on failure” steps.

The first step to execute on failure will send a request to DAM to get a HPC location where to run the HPC computation. This is like what was done at the beginning of the workflow, but this second request to DAM here in the “on failure” branch of the workflow will contain a reference to the first request done. So, DAM will be able to retrieve the previous location that was selected, and will return a new location if available, even if this new location appears as less preferable than the location where the failure occurred, to avoid any potential issue on this previous location that could have caused the computation failure.

A new HEAppE job will be created on this location, the input dataset and checkpoint dataset will be transferred in input of this new job, and the new HPC computation job will be submitted.



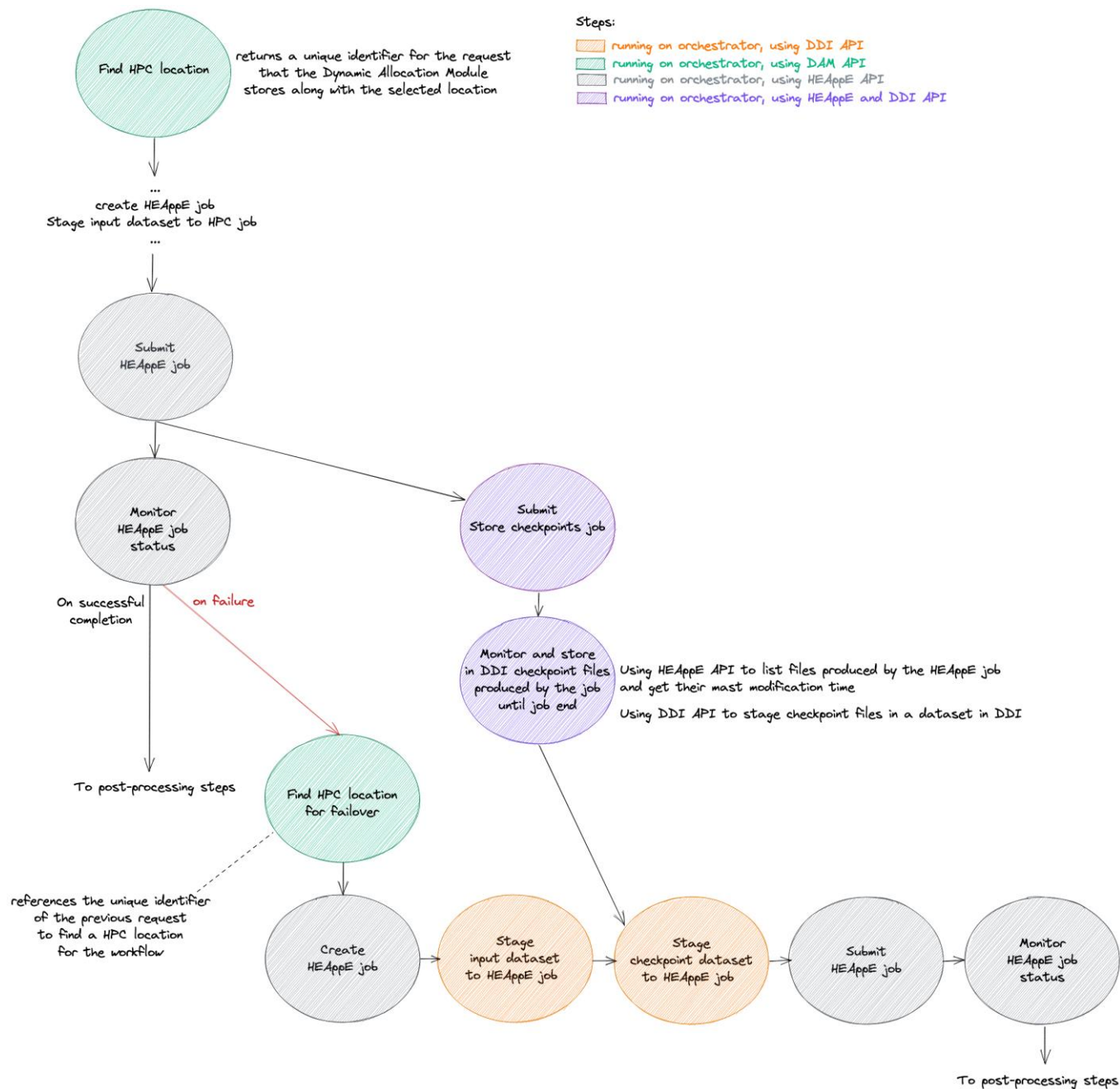


Figure 19 Workflow with on failure steps

### 3 SOFTWARE DESCRIPTION

This section describes the software repositories for each LEXIS Orchestration System component.

#### 3.1 ALIEN4CLOUD

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

<b>SOURCE CODE</b>	<a href="https://github.com/alien4cloud/alien4cloud">https://github.com/alien4cloud/alien4cloud</a>
<b>LICENSE</b>	Apache 2.0
<b>DOCUMENTATION</b>	<a href="http://alien4cloud.github.io/">http://alien4cloud.github.io/</a>
<b>RELEASE</b>	version 3.3.0 <a href="https://www.portaildulibre.fr/nexus/repository/opensource-releases/alien4cloud/alien4cloud-premium-dist/3.3.0/alien4cloud-premium-dist-3.3.0-dist.tar.gz">https://www.portaildulibre.fr/nexus/repository/opensource-releases/alien4cloud/alien4cloud-premium-dist/3.3.0/alien4cloud-premium-dist-3.3.0-dist.tar.gz</a>

Table 1 Alien4Cloud repository

#### 3.2 ALIEN4CLOUD GO CLIENT LIBRARY

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

It was implemented in order to ease the integration of the orchestration service front-end into the LEXIS Portal.

<b>SOURCE CODE</b>	<a href="https://github.com/alien4cloud/alien4cloud-go-client">https://github.com/alien4cloud/alien4cloud-go-client</a>
<b>LICENSE</b>	Apache 2.0
<b>DOCUMENTATION</b>	<a href="https://github.com/alien4cloud/alien4cloud-go-client/blob/master/README.md">https://github.com/alien4cloud/alien4cloud-go-client/blob/master/README.md</a>
<b>RELEASE</b>	version 3.0.0-milestone.5 <a href="https://github.com/alien4cloud/alien4cloud-go-client/releases/tag/v3.0.0-milestone.5">https://github.com/alien4cloud/alien4cloud-go-client/releases/tag/v3.0.0-milestone.5</a>

Table 2 Alien4Cloud Go Client Library repository

#### 3.3 ALIEN4CLOUD YORC PROVIDER PLUGIN

The A4C Yorc provider plugin allows 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).

<b>SOURCE CODE</b>	<a href="https://github.com/alien4cloud/alien4cloud-yorc-provider">https://github.com/alien4cloud/alien4cloud-yorc-provider</a>
<b>LICENSE</b>	Apache 2.0
<b>DOCUMENTATION</b>	<a href="https://alien4cloud.github.io/#/documentation/3.3.0/orchestrators/yorc/index.html">https://alien4cloud.github.io/#/documentation/3.3.0/orchestrators/yorc/index.html</a>
<b>RELEASE</b>	version 3.3.0 bundled with A4C

	<a href="https://www.portaildulibre.fr/nexus/repository/opensource-releases/alien4cloud/alien4cloud-yorc-provider/3.3.0/alien4cloud-yorc-provider-3.3.0.zip">https://www.portaildulibre.fr/nexus/repository/opensource-releases/alien4cloud/alien4cloud-yorc-provider/3.3.0/alien4cloud-yorc-provider-3.3.0.zip</a>
--	---

Table 3 Alien4Cloud Yorc Provider plugin repository

### 3.4 YORC

The Ystia Orchestrator (Yorc) is a TOSCA **Je zadán neplatný pramen.** 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	<a href="https://github.com/ystia/yorc">https://github.com/ystia/yorc</a>
LICENSE	Apache 2.0
DOCUMENTATION	<a href="https://yorc.readthedocs.io/en/latest/">https://yorc.readthedocs.io/en/latest/</a>
RELEASE	version 4.1.1 <a href="https://github.com/ystia/yorc/releases">https://github.com/ystia/yorc/releases</a>

Table 4 Yorc repository

### 3.5 YORC HEAPPE PLUGIN

The Yorc HEAppE plugin extends the Ystia Orchestrator (Yorc) so it can use HEAppE to submit and monitor jobs, get the list of files produced by the jobs and their last modification date.

SOURCE CODE	<a href="https://github.com/lexis-project/yorc-heappe-plugin">https://github.com/lexis-project/yorc-heappe-plugin</a>
LICENSE	Apache 2.0
DOCUMENTATION	<a href="https://github.com/lexis-project/yorc-heappe-plugin/blob/master/README.md">https://github.com/lexis-project/yorc-heappe-plugin/blob/master/README.md</a>
RELEASE	version 1.0.6 <a href="https://github.com/lexis-project/yorc-heappe-plugin/releases">https://github.com/lexis-project/yorc-heappe-plugin/releases</a>

Table 5 Yorc HEAppE plugin repository

This plugin provides an implementation for the following TOSCA components used to interact with HEAppE in LEXIS workflows:

- `org.lexis.common.heappe.nodes.Job`

A HEAppE job implementing the standard TOSCA operations lifecycle *create*, *delete*, and Job extensions lifecycle operations *submit*, *run*, *cancel*.

Custom operations are also implemented to provide the corresponding HEAppE API features:

- `enable_file_transfer`: Enables files transfer to/from the job,
- `disable_file_transfer`: Disables files transfer to/from the job,
- `list_changed_files`: lists the files created/updated by the job, and their modification date.

The list of files created/updated by the job is provided by the component attribute `changed_files`.



- **org.lexis.common.heappe.nodes.JobWithRuntimeTaskParameters**

A HEAppE job for which the task parameters are not properties configured before the deployment, but are attributes defined at runtime by another TOSCA component responsible for computing tasks parameters at runtime during the workflow execution.

- **org.lexis.common.heappe.nodes.WaitFileAndGetContentJob**

A component associated to a HEAppE job, allowing to wait for a given file to be generated by the job, and to get the content of this file. The content of the file is provided by the component attribute `filecontent`.

### 3.6 YORC DDI PLUGIN

The Yorc DDI plugin extends the Ystia Orchestrator (Yorc) so it can use LEXIS DDI API (Distributed Data Infrastructure) to manage asynchronous dataset transfer requests.

<b>SOURCE CODE</b>	<a href="https://github.com/lexis-project/yorc-ddi-plugin">https://github.com/lexis-project/yorc-ddi-plugin</a>
<b>LICENSE</b>	Apache 2.0
<b>DOCUMENTATION</b>	<a href="https://github.com/lexis-project/yorc-ddi-plugin/blob/master/README.md">https://github.com/lexis-project/yorc-ddi-plugin/blob/master/README.md</a>
<b>RELEASE</b>	version 1.0.0 <a href="https://github.com/lexis-project/yorc-ddi-plugin/releases">https://github.com/lexis-project/yorc-ddi-plugin/releases</a>

Table 6 Yorc DDI plugin repository

This plugin provides an implementation for the following TOSCA components used to interact with the LEXIS DDI (Distributed Data Infrastructure) in LEXIS workflows:

- **org.lexis.common.ddi.nodes.DDIToCloudJob**

Job (TOSCA component implementing the standard TOSCA operations lifecycle *create*, *delete*, and Job extensions lifecycle operations *submit*, *run*, *cancel*) executing the transfer of dataset from DDI to Cloud staging area.

- **org.lexis.common.ddi.nodes.CloudToDDIJob**

Job executing a transfer of dataset from Cloud staging area to DDI.

- **org.lexis.common.ddi.nodes.DeleteCloudDataJob**

Job deleting a dataset from Cloud staging area.

- **org.lexis.common.ddi.nodes.DDIToHPCTaskJob**

Job executing a transfer of dataset from DDI to HPC in a directory for a given task in the job.

- **org.lexis.common.ddi.nodes.HPCToDDIJob**

Job executing a transfer of data from a HPC job directory to DDI.

- **org.lexis.common.ddi.nodes.StoreRunningHPCJobFilesToDDIJob**

Job monitoring a HEAppE job and transferring new files produced by this HEAppE job to DDI, until this HEAppE job ends.

- **org.lexis.common.ddi.nodes.StoreRunningHPCJobFilesToDDIGroupByDatasetJob**

Job monitoring a HEAppE job and transferring new files produced by this HEAppE job to DDI until this HEAppE job ends, and grouping these files in datasets according to a pattern.

- **`org.lexis.common.ddi.nodes.WaitForDDIDatasetJob`**  
Job waiting for a dataset to appear in DDI, and optionally waiting for files of a given pattern to appear in this dataset.
- **`org.lexis.common.ddi.nodes.DDIRuntimeToCloudJob`**  
Job executing a transfer of dataset from DDI to Cloud staging area, the dataset being provided at runtime by an associated component (while `org.lexis.common.ddi.nodes.DDIToCloudJob` has the DDI dataset path to transfer as a property, statically defined before the execution of the workflow).
- **`org.lexis.common.ddi.nodes.DDIRuntimeToHPCTaskJob`**  
Job executing a transfer of dataset from DDI to HPC in a directory for a given task in the job.
- **`org.lexis.common.ddi.nodes.GetDDIDatasetInfoJob`**  
Job executing a request to get a DDI dataset info (size, number of files, number of small files of size <= 32MB).
- **`org.lexis.common.ddi.nodes.GetComputeInstanceDatasetInfo`**  
Component providing info on a directory in a compute instance (size, number of files, number of small files of size <= 32MB).
- **`org.lexis.common.ddi.nodes.GetHPCJobTaskDatasetInfo`**  
Component providing info on files produced by a HEAppE job (size, number of files, number of small files of size <= 32MB).
- **`org.lexis.common.ddi.nodes.SSHFSMountStagingAreaDataset`**  
Component mounting a dataset in the Cloud staging area on a compute instance directory through SSHFS.

### 3.7 YORC DYNAMIC ORCHESTRATION PLUGIN

The Yorc Dynamic Orchestration plugin extends the Ystia Orchestrator (Yorc) so it can use the LEXIS DAM API during the workflow execution to select at runtime Cloud and HPC locations where to allocate infrastructure resources. It provides TOSCA components as well, taking care of exchanging/refreshing LEXIS AAI tokens during the workflow execution.

<b>SOURCE CODE</b>	<a href="https://github.com/lexis-project/yorc-dynamic-orchestration-plugin">https://github.com/lexis-project/yorc-dynamic-orchestration-plugin</a>
<b>LICENSE</b>	Apache 2.0
<b>DOCUMENTATION</b>	<a href="https://github.com/lexis-project/yorc-dynamic-orchestration-plugin/blob/master/README.md">https://github.com/lexis-project/yorc-dynamic-orchestration-plugin/blob/master/README.md</a>
<b>RELEASE</b>	version 1.0.0 <a href="https://github.com/lexis-project/yorc-dynamic-orchestration-plugin/releases">https://github.com/lexis-project/yorc-dynamic-orchestration-plugin/releases</a>

**Table 7 Yorc Dynamic Orchestration plugin repository**

This plugin provides an implementation for the following TOSCA components used to interact with the LEXIS DAM and to exchange/refresh LEXIS AAI tokens in LEXIS workflows:

- **`org.lexis.common.dynamic.orchestration.nodes.SetLocationsJob`**  
Requests DAM to provide the best locations where to allocate Cloud or HPC infrastructure resources and update the TOSCA components in the workflow so that they will be allocated on the selected locations.

- `org.lexis.common.dynamic.orchestration.nodes.ValidateAndExchangeToken`  
Validates an input token and uses the OpenID Connect "Token exchange" feature to exchange the token received from the caller (LEXIS Portal), for a token accepted by the orchestrator.
- `org.lexis.common.dynamic.orchestration.nodes.RefreshTargetTokens`  
Refreshes the access token attribute of the associated target.

### 3.8 YORC OIDC CLIENT

The Yorc OIDC client is a Go client library used by Yorc LEXIS plugins described above to manage OpenID Connect tokens: exchange token, check if a token is still valid, refresh a token, get user info from this token.

<b>SOURCE CODE</b>	<a href="https://github.com/lexis-project/yorcoidc">https://github.com/lexis-project/yorcoidc</a>
<b>LICENSE</b>	Apache 2.0
<b>DOCUMENTATION</b>	<a href="https://github.com/lexis-project/yorcoidc/blob/master/README.md">https://github.com/lexis-project/yorcoidc/blob/master/README.md</a>
<b>RELEASE</b>	version 1.0.0 <a href="https://github.com/lexis-project/yorcoidc/releases">https://github.com/lexis-project/yorcoidc/releases</a>

Table 8 Yorc OIDC client repository

### 3.9 DYNAMIC ALLOCATION MODULE

The Dynamic Allocation Module (DAM) embeds the orchestration logic needed to dynamically determine the best location for the execution of a given task. To this end, it provides an API to request the best location where to allocate Cloud/HPC infrastructure resources according to various criteria collected from the LEXIS platform (performance, usage, planned maintenance, etc.).

This API is used by the Yorc Dynamic Orchestration plugin during a workflow execution to select dynamically (at runtime) the locations where to allocate infrastructure resources just before these resources are needed in the workflow. At the time of writing, DAM is in a private repository as the code is under quality check. It will be released to public at the end of this process, while the availability status will be reported in D2.5 [10].

<b>SOURCE CODE</b>	<a href="https://github.com/lexis-project/dynamic-allocation-module">https://github.com/lexis-project/dynamic-allocation-module</a>
<b>LICENSE</b>	MIT
<b>DOCUMENTATION</b>	<a href="https://github.com/lexis-project/dynamic-allocation-module/blob/main/README.md">https://github.com/lexis-project/dynamic-allocation-module/blob/main/README.md</a>
<b>RELEASE</b>	version 1.0.0 <a href="https://github.com/lexis-project/dynamic-allocation-module/releases">https://github.com/lexis-project/dynamic-allocation-module/releases</a>

Table 9 Dynamic Allocation Module repository

### 3.10 ORCHESTRATION SERVICE API

The orchestration Service API developed by ICHEC is used by the WP8 LEXIS Portal to interact with the Orchestration System.

This API translates LEXIS concepts exposed by the Portal (workflow, workflow execution) to Ystia concepts used in Alien4Cloud/Yorc (application template, deployment, workflow run), as described in the glossary and terminology section of deliverable D4.4 [6]. At the time of writing, the Orchestration Service API code is under quality check. It will be released to public at the end of this process, while the availability status will be reported in D2.5 [10].

<b>SOURCE CODE</b>	<a href="https://code.it4i.cz/lexis/wp8/alien4cloud-interface">https://code.it4i.cz/lexis/wp8/alien4cloud-interface</a>
<b>LICENSE</b>	Apache 2.0
<b>DOCUMENTATION</b>	<a href="https://code.it4i.cz/lexis/wp8/alien4cloud-interface/-/blob/develop/a4c-service-swagger-definition.yaml">https://code.it4i.cz/lexis/wp8/alien4cloud-interface/-/blob/develop/a4c-service-swagger-definition.yaml</a>
<b>RELEASE</b>	<a href="https://code.it4i.cz/lexis/wp8/alien4cloud-interface/-/tags">https://code.it4i.cz/lexis/wp8/alien4cloud-interface/-/tags</a>

Table 10 Orchestration Service API repository

### 3.11 YSTIA FORGE

The Ystia Forge is a repository for the TOSCA Types and Topology templates used by the Ystia project.

<b>SOURCE CODE</b>	<a href="https://github.com/ystia/forge">https://github.com/ystia/forge</a>
<b>LICENSE</b>	Apache 2.0
<b>DOCUMENTATION</b>	<a href="https://github.com/ystia/forge/blob/develop/org/ystia/README.rst">https://github.com/ystia/forge/blob/develop/org/ystia/README.rst</a>
<b>RELEASE</b>	version 3.0.0-milestone.1 <a href="https://github.com/ystia/forge/releases">https://github.com/ystia/forge/releases</a>

Table 11 Ystia Forge repository

The following TOSCA components from the Ystia Forge are used in LEXIS workflows:

- **org.ystia.docker.ansible.nodes.Docker**  
Component for installing, configuring and starting Docker on a Cloud compute instance.
- **org.ystia.docker.containers.docker.generic.nodes.GenericContainer**  
Component for creating and starting a docker container.
- **org.ystia.docker.images.ArchiveLoader**  
Component for loading a docker image from a tar archive.

### 3.12 REPOSITORIES OF LEXIS TOSCA COMPONENTS AND APPLICATION TEMPLATES

In addition to the Orchestration System, generic workflows are provided to cover basic use cases of workflows on Hybrid Cloud/HPC infrastructures.

Components created for the pilot workflows are provided as well in the public lexis-project GitHub repository or in a private repository, based on their requirements which are provided either in the public lexis-project GitHub repository, or in a private repository.

#### 3.12.1 Public repository

<b>SOURCE CODE</b>	<a href="https://github.com/lexis-project/application-templates">https://github.com/lexis-project/application-templates</a>
<b>LICENSE</b>	Apache 2.0
<b>DOCUMENTATION</b>	<a href="https://github.com/lexis-project/application-templates/blob/master/README.md">https://github.com/lexis-project/application-templates/blob/master/README.md</a>
<b>RELEASE</b>	version 0.1.3 <a href="https://github.com/lexis-project/application-templates/releases">https://github.com/lexis-project/application-templates/releases</a>

Table 12 Public repository

The Public repository provides TOSCA implementations for the following workflows.

**Generic workflows:**

- **`org.lexis.common.LEXISTemplate`**

Generic workflow performing the following Cloud/HPC Operations:

- Transfer of an input dataset from DDI to the Cloud staging area,
- Pre-processing of the input data by a docker container executed on a LEXIS Cloud Compute instance created on demand,
- Transfer of pre-processing results to a HEAppE job created on a LEXIS HPC cluster,
- Transfer of HEAppE job results to DDI and to the Cloud Compute instance,
- Post-processing of the computation results, running a post-processing docker container,
- Transfer of post-processing results to DDI.

See a detailed description of this workflow at:

<https://github.com/lexis-project/application-templates/tree/master/examples/applications/cloudHPCComputation>

- **`org.lexis.common.CloudComputationPublicContainerTemplate`**

Workflow allowing to transfer a dataset from DDI to a Cloud Compute instance on which a computation will be done by a Docker container, and to transfer results in the DDI.

See detailed description at:

<https://github.com/lexis-project/application-templates/tree/master/examples/applications/cloudComputationPublicContainer>

- **`org.lexis.common.CloudComputationTemplate`**

Workflow allowing to transfer a dataset from DDI to a Cloud Compute instance on which a user-defined script ins run to produce results. Results are then transferred to the DDI.

See details description at:

<https://github.com/lexis-project/application-templates/tree/master/examples/applications/cloudComputation>

- **`org.lexis.common.HPCComputationTemplate`**

Workflow allowing to transfer a dataset from DDI to a HEAppE Job, and to transfer the HEAppE job results to the DDI.

See detailed description at:

<https://github.com/lexis-project/application-templates/tree/master/examples/applications/hpcComputation>

**Computational Fluid Dynamics workflow:**

- **`org.lexis.common.OpenFOAMTemplate`**

Workflow allowing to perform an OpenFoam computation and to visualize the results.

See detailed description at:

<https://github.com/lexis-project/application-templates/tree/master/computational-fluid-dynamics/applications/openfoam>

**Remote visualization workflow:**

- **`org.lexis.common.VisualizationTemplate`**

Workflow allowing to transfer a dataset from DDI to a Cloud Compute instance on which the data analysis and visualization application ParaView is installed and accessible through a remote session.

See detailed description at:

<https://github.com/lexis-project/application-templates/tree/master/visualization/applications/xpra>

### Weather and Climate workflows implemented for WP7:

- **org.lexis.wp7.RisicoTemplate**

This workflow performs risks of wildlands fires simulations:

- Downloading the required input data (static geographical data and Global Forecast System data) for a date specified by the user,
- Pre-processing these data on a Cloud Compute instance using a pre-processing container provided by CIMA,
- Transferring this data to a WRF HEAppE job on a HPC cluster,
- Post-processing WRF on a Cloud Compute instance using a post-processing RISICO container provided by CIMA,
- Transferring RISICO results to LEXIS DDI.

See detailed description at:

<https://github.com/lexis-project/application-templates/tree/master/weather-climate/applications/risico>

- **org.lexis.wp7.ContinuumTemplate**

This workflow performs hydrology simulations:

- Downloading the required input data (static geographical data and Global Forecast System data) for a date specified by the user,
- Pre-processing the data on a Cloud Compute instance using a pre-processing container provided by CIMA,
- Transferring this data to a WRF HEAppE job on a HPC cluster,
- Post-processing WRF results on a Cloud Compute instance using two post-processing containers provided by CIMA:
  - A container which prepares datasets for the Hydrological model for a given region - 17 instances of this container are run in parallel for 17 regions of Italy,
  - A container running the Continuum hydrological model for a given region - 17 instances of this container are run in parallel for 17 regions of Italy.
- Transferring these results to LEXIS DDI and optionally on a SFTP server when specified by the user.

See detailed description at:

<https://github.com/lexis-project/application-templates/tree/master/weather-climate/applications/continuum>

- **org.lexis.wp7.ADMSTemplate**

This workflow is running ADMS:

- Downloading the required input data (static geographical data and ECMWF data) for a date specified by the user,
- Pre-processing the data on a Cloud Compute instance using a pre-processing container provided by CIMA,
- Transferring this data to a WRF HEAppE job on a HPC cluster,
- Post-processing WRF results on a Cloud Compute instance using a container executing a script provided by NUM retrieved from LEXIS DDI,
- Running ADMS on a Cloud Compute Instance using a specific Windows images available on the OpenStack Cloud locations provided by LEXIS,
- Transferring the results to LEXIS DDI and optionally on a SFTP server when specified by the user.

For convenience, an ADMS post-processing workflow is also implemented. This workflow takes in as an input a WRF results dataset that was stored in DDI by a previous ADMS template workflow execution and

runs only the post-processing described above without having to execute the pre-processing and HPC computation phases of the ADMS workflow.

See detailed description at:

<https://github.com/lexis-project/application-templates/tree/master/weather-climate/applications/adms>

- **org.lexis.wp7.AgricultureTemplate**

This workflow executes the same pre-processing on Cloud Compute instance and same HPC computation as ADMS above, and then stores these HPC computation results in DDI and optionally on a SFTP server when specified by the user, for an offline post-processing.

See detailed description at:

<https://github.com/lexis-project/application-templates/tree/master/weather-climate/applications/agriculture>

### 3.12.2 Private repositories

Private repositories are used for WP5 Aeronautics workflows in the IT4I GitLab repository:

<https://code.it4i.cz/lexis/wp4/application-templates>

and WP6 Earthquake and Tsunami workflows:

<https://code.it4i.cz/lexis/wp6/pilottaskstemplates>

## 4 INSTALLATION ON LEXIS INFRASTRUCTURES

### 4.1 YSTIA INSTALLATION

The Ystia stack was installed on both datacenters at IT4I and LRZ, on a 2 CPUs/8GB host, [ystia.msad.it4i.lexis.tech](https://ystia.msad.it4i.lexis.tech) at IT4I, and [sikplr-z-lexis-orchestrator.srv.mwn.de](https://sikplr-z-lexis-orchestrator.srv.mwn.de) at LRZ, both hosts are accessible only through a VPN.

No system configuration change was required, except from the definition of a route on the LRZ host (172.16.0.0/16 via 141.40.145.101) so that the orchestrator can access systems at IT4I.

The Ystia stack was installed using Yorc bootstrap command-line, which allows the deployment of the full Ystia stack Alien4Cloud/Yorc as described in Yorc documentation [11].

The secured deployment option was selected: TLS with mutual authentication between the Ystia stack components is configured.

Yorc LEXIS plugins described above were then installed on the system and Yorc restarted to take them into account.

TOSCA components and application templates described above were then added to the Alien4Cloud catalogue of TOSCA components, so that they can be used by LEXIS Portal to create workflows.

### 4.2 DYNAMIC ALLOCATION MODULE INSTALLATION

The DAM stack was installed on LRZ datacenter, on a 2 CPUs/8GB host VM host, accessible only through a VPN.

No system configuration was required. The DAM stack was installed by cloning the GitHub repository linked in Section 3 of this document. All the dependencies were installed as described in the repository documentation. Finally, DAM has been launched after properly editing the configuration file. The whole process is described in the documentation in the repository.

## 5 SUMMARY

This deliverable describes the final 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 Atos.

Yorc plugins were developed for the LEXIS project to extend the orchestrator so that it supports the management of HPC infrastructure resources through the middleware developed by IT4I (HEAppE), the management of data transfers through the DDI API provided by WP3, and the selection of locations where to allocate infrastructure resources dynamically during the workflow execution using DAM developed by LINKS.

A Go client was also developed by Atos to ease the integration with Alien4Cloud. It is used by the Orchestration Service API developed by ICHEC, this API being used by WP8 LEXIS Portal to interact with the Orchestration System.

The Orchestration System was installed at IT4I and LRZ, allowing to deploy applications and run workflows on LEXIS Cloud infrastructure and HPC infrastructures provided by LRZ and IT4I.

Generic workflows covering basic and common usages were implemented, as well as Pilot workflows in cooperation with Work Packages partners, so that they can run their applications on LEXIS.

To validate the demonstrator, the CIMA team could run WP7 workflows performing computations on the LEXIS federated Cloud/HPC environment and validate results of these computations.



## 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](https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca).
- [2] "Orchestrator front-end Alien4Cloud," 2021. [Online]. Available: <https://github.com/alien4cloud/alien4cloud>.
- [3] LEXIS Deliverable, *D4.2 Design and Implementation of the HPC-Federated Orchestration System - Intermediate*.
- [4] "Ystia Orchestrator Yorc," 2021. [Online]. Available: <https://github.com/ystia/yorc>.
- [5] LEXIS Deliverable, *D4.3 Definition of Data Access Priority, Analytics Policies, and Security Assessment*.
- [6] LEXIS Deliverable, *D4.4 Definition of workload management policies in federated cloud/HPC environments*.
- [7] LEXIS Deliverable, *D3.3 Mid-Term Infrastructure (Deployed System Hard/Software)*.
- [8] "HEAppE Middleware," 2021. [Online]. Available: <https://heappe.eu>.
- [9] "myDEWETRA," 2021. [Online]. Available: <https://www.cimafoundation.org/foundations/research-development/my-dewetra.html>.
- [10] LEXIS Deliverable, *D2.5 Final Assessment of the Co-Designed LEXIS Architecture*.
- [11] "Yorc Bootstrap documentation," 2021. [Online]. Available: <https://yorc.readthedocs.io/en/latest/bootstrap.html>.