eFlows4HPC

Enabling dynamic and Intelligent workflows in the future EuroHPC ecosystem

D2.5 Final Report on Data Logistics Implementation

Version 1.0

Documentation Information

Contract Number	9555558
Project Website	www.eFlows4HPC.eu
Contratual Deadline	29.02.2024
Dissemination Level	PU
Nature	R
Author	Jedrzej Rybicki (FZJ)
Contributors	
Reviewer	Rosa M Badia (BSC)
Keywords	Data, Workflows, HPC

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955558. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Germany, France, Italy, Poland, Switzerland, Norway.

Change Log

Version	Description Change
V0.1	Initial version of the document layout
V0.2	After the internal review
V1.0	Final version, formatted for submission

Table of Contents

1 Executive Summary

This report gives an overview of the implementation process of the Data Logistics Service. The service is based on the open source Apache Airflow, which was extended, customized and integrated into the eFlows4HPC software stack. This document describes the changes. In particular, we describe the implementation of the service, the implementation of the data movement pipelines, and the integration of the service with the eFlows4HPC software stack.

The project has implemented a number of data movement pipelines to address the specific needs of the pillars and their workflows. A complete list of these pipelines is provided in this document. Thanks to the use of the Data Catalogue, they are quite generic and can be used to move different datasets. The pipelines are available in the open repository and can be reused outside of the project.

Many scientific services suffer from the problem of not reaching the maturity required by users. They are rarely updated and become obsolete. With eFlows4HPC, and the Data Logistics Service in particular, we invested extra effort to make the automatic deployment of the service as easy as possible. This allowed us to quickly add new features and keep the service up to date. This was achieved by developing a set of Continuous Deployment (CD) pipelines and using dynamic IaaS cloud resources. These pipelines ensure that every change to the service is tested and deployed quickly and in a reproducible manner. Users can also view the status of the service using a monitoring solution. All that contributes strongly to establishing users' trust in the service.

The Data Logistics Service developed in the project was widely used by the workflow developers, made the created workflows portable and contributed significantly to the success of the project. The integration with the workflow orchestration engine and HPC-Workflow-as-a-Service (HPCWaaS) provided a seamless experience during the workflow development and execution. The addition of automatic deployments and tests increased user confidence in the service and is an important point for service reuse beyond the project lifetime.

2 Introduction

Data Logistics Service is part of the eFlows4HPC software stack and is responsible for facilitating data movements required by the workflows. To set up the stage for further discussion let us show where to place the Data Logistics Service (DLS) in the solution created in the eFlows4HPC project. The components overview can be found in Figure 1.

From the workflow perspective the data pipelines also play a well-defined role. Where they are located can be seen on Figure 2.

More details on this subject can be found in D1.5 “eFlows4HPC interfaces and final software stack release”, whereas the internal workings of the Data Logistics Service were covered in the D2.3 “First version of Data Logistics”, and will be omitted here.

In short, the Pillars workflows can include the action of calling particular data movement (defined as pipelines in the DLS). The workflow orchestrator upon executing such an action contacts DLS and triggers the particular pipeline and monitors its execution.

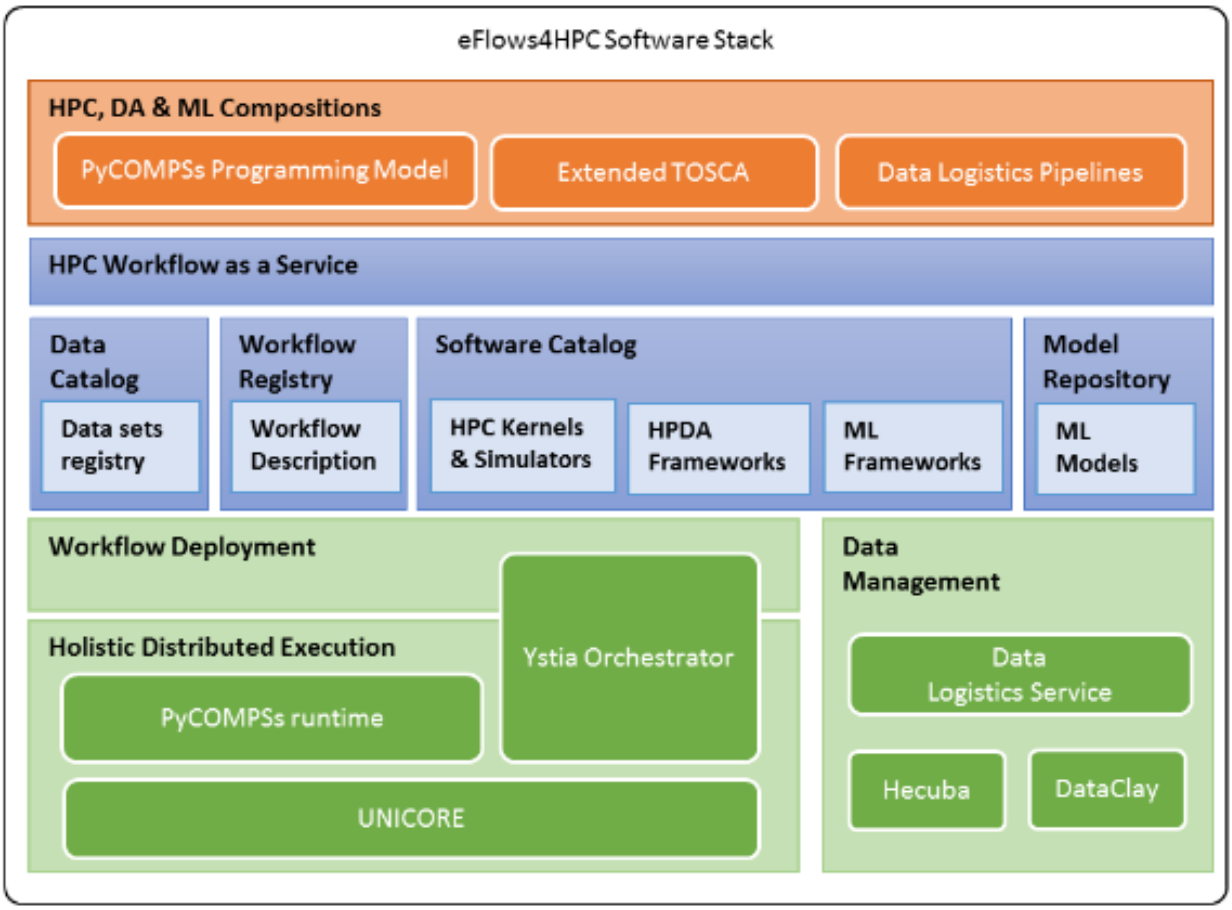

Figure 1 Components overview

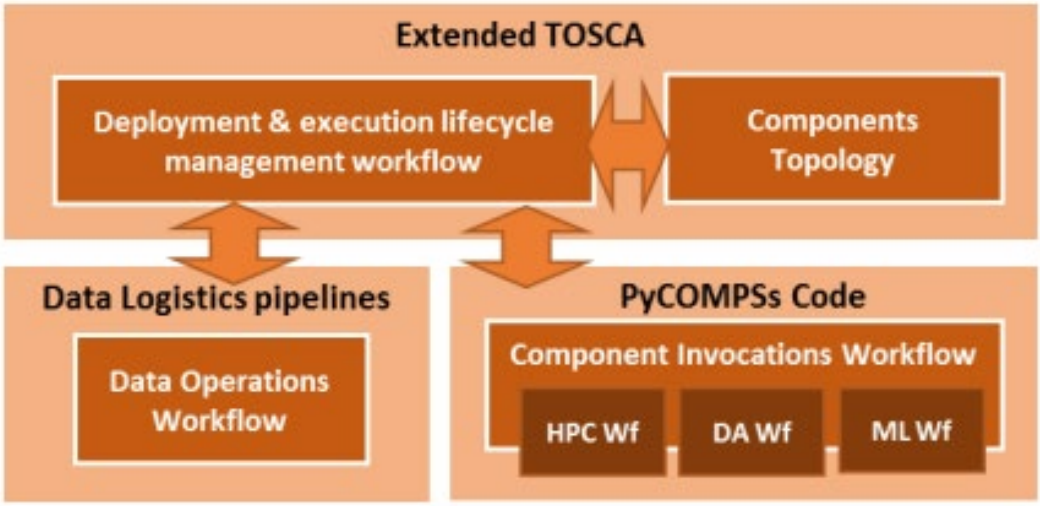

Figure 2 Workflow overview

3 Implementation

This section describes the implementation around the DLS undertaken in eFlows4HPC project. We start with a description of all the pipelines created during the implementation of the project workflows, then describe the integration with external services (including solutions for credentials and access management). We also describe how the process of automatic service and pipelines deployment was set up. The DLS service is based on Apache Airflow. We will not describe the original software in depth but rather focus on the project-specific modification and extensions.

3.1 Implementation of the pipelines

The DLS pipelines are formalizations of the data movements. They are programmed with Python and comprise a set of tasks that need to be executed and the definition of a DAG (directed acyclic graph) that defines the order of those operations. The pipelines can be found in [1]. Here we present a short list with comments on their functionality (see Table 1).

Table 1 List of data movement pipelines created in the project

Name	Functionality
git2ssh	Allows to stage-in data from a gitlab repository to target location via SSH protocol
mlflow_download	Pipeline to stage-in ML models from the model repository into HPC facility
mlflow_model_transfer mlflow_upload_model	Pipelines to stage-out trained model from a local model repository into the global model repository. One uses a local serialized model, other uses a local model repository instance.
ssh2ssh	Pipeline to copy data between two locations accessible through SSH. This is a recursive copy (with subdirectories).
taskflow_example	The pipeline copies data objects from B2SHARE to an SSH target location.
taskflow_stream	The pipeline copies data objects from B2SHARE to an SSH target location by using streaming.
testdag	Testing pipeline used in the service deployment phase to verify the correctness of the service update.
transfer_image	Pipeline to transfer Singularity images from the image repository to the HPC facility.
upload_example	Pipeline to upload data from HPC resource to B2SHARE repository.
verify_config	Pipeline to verify the config of airflow is correct. Used in the setup of new instances of DLS.
webdav2unicore	Stages-in the data from a WebDAV repository (e.g. B2DROP) to an HPC system accessible through Unicore.

webdav_stagein	Stages in the data from a WebDAV repository (e.g. B2DROP) to an HPC system using SSH. This is a recursive copy with subdirectories.
webdav_stageout	Moves data from a SSH source to a WebDAV repository (like B2DROP). This is a recursive copy with subdirectories.

Each pipeline can consist of multiple tasks. Since some of the tasks are common across different pipelines (e.g. SSH-based access), we encapsulated them in separate programming modules which are shared between the pipelines. These parts were also optimized to achieve high transfer speeds.

3.2 Integration with other services

3.2.1 Alien4Cloud and Ystia

The eFlows4HPC workflows are defined with help of Alien4Cloud and described in TOSCA standard. They can include execution of data pipelines. Ystia is then responsible for the orchestration of the workflows' execution. It parses the TOSCA description and interprets the actions as defined by the workflow developer. Upon encountering an action requiring execution of a DLS pipeline, the Ystia issues a request to the Airflow API. This is HTTP(s)-based RestAPI. In the project, we only used a subset of its functionality. To trigger a pipeline, a request with the pipeline's parameters (like input data sets, etc), a JSON document needs to be sent to the respective endpoint. Through this API also the status of the pipeline execution can be retrieved and presented to the user.

3.2.2 Data Catalogue

The pipelines developed in the project were designed to be as flexible as possible, facilitating the resume of the pipelines across the workflow implementations. To this end, rather than hard coding the location of the dataset in the pipeline, the data that need to be moved (e.g. for stage-in) was registered in the external Data Catalogue and then the pointer to this record can be passed to a pipeline. Also, the data stage-out by the eFlows4HPC pipelines are automatically registered with the Data Catalogue. This serves two purposes. Firstly, the results produced in the pipeline can be used as inputs for a different pipeline. Secondly, the results of the project are visible for external users facilitating reuse of the data (in spirit of FAIR principles).

3.3 Access and credentials management

The execution of the workflows naturally requires interaction of multiple components. For instance, a call to the HPCWaaS to start a workflow execution Is passed to Alien4Cloud/Ystia (which is responsible for instantiation of the workflow). This service in turn needs to contact the DLS to start data transfers required for instance to stage-in the data. The DLS then needs to contact at least the data repository and the target HPC system to facilitate the data transfer. The process gets even more complicated for the eFlows4HPC workflows as they include image building, and transfer, execution of multiple processing commands, stage-out and registration of the data output, etc. One of the main challenges here is that the interaction of the components needs credentials management. To orchestrate the data movement on behalf of the users, the DLS needs to have user and service specific credentials. Rather than developing a new solution for that, and also due to the fact that the Alien4Cloud has the same requirement, we decided to use the industry-standard solution Vault [1]. With help of the HPCWaaS client, the user is able to securely store her credentials into the Vault from which DLS or Alien4Cloud can access them and facilitate

the actions on behalf of the user. We have prepared a set of functions that are able to interact with the Vault and can be incorporated into data movement pipelines. Even for a situation where a new pipeline needs to be developed, the developer only needs to focus on the actual data movement and simply reuse the tasks responsible for accessing the Vault and setting up the required connections.

The above described solution tackles the challenge of integrating multiple service components. But for the end-users direct access to Vault would not make much sense. Yet still they need to access DLS for instance to view information about how the data transfer was completed. This is especially relevant in case of an error, where debugging across services to pinpoint the action that failed in the workflow is often needed. The DLS also gives some basic information about performance of the data movement (time required by particular tasks), which may also be of interest to end-users. To this end, users should be able to access the web interface of the DLS service. Preferably, the same credentials should be used to access Alien4Cloud and other components, to avoid a situation where a user needs a long list of credentials to access all the services. Therefore, the original Airflow software was extended and integrated with a Single-Sign-On solution. To this end we started to manage an Identity Provider (IdP) based on the software Unity [3]. Focusing on the relevant details of unity for the DLS, Unity can act as an OAuth2 provider, which is a protocol for authenticating users via an external user management system. Using this, we could then implement another login path for users and developers, where Unity verifies the credentials and provides the required access levels to the DLS. This was then also extended to some other eFlows4HPC services, resulting in a true Single-Sign-On experience for the users. The solution was successfully evaluated with the Data Logistics Service and Data Catalogue, and rolled out to other eFlows4HPC services.

3.4 Continuous deployment, testing, and monitoring

One of the challenges that we faced is to provide an easy way of updating and deploying new versions of pipelines. Also, periodic updates of services (e.g. caused by the availability of new versions of Airflow) shall be conducted in a quick yet safe manner. To this end, we decided to implement an automatic deployment strategy with help of Gitlab pipelines. The pipelines consist of some unit tests to verify the correctness of the pipeline implementations. Subsequently a new Docker image containing the new version of the software is prepared, and then deployed first to a test instance (to conduct integration tests and make sure that the new version does not break some hidden dependencies). After this verification the test version is promoted to become the production version. Similar pipeline is used to keep the instance of Data Catalogue up-to-date.

To further reinforce the trust of the users to the infrastructure we deployed a simple monitoring solution [4]. It gives basic information about the availability (and health) of critical infrastructure components. Each user can thus verify that the components required for workflow deployment are indeed in place and working correctly. How this solution looks can be viewed in Figure 3. Such a monitoring solution is a must in a distributed environment.

At some point we noticed that the update frequencies for service and pipelines differ. The service became stable while new and updated pipelines needed to be deployed more frequently. Therefore, we decided to move the pipelines to a separate repository [1], and deploy them to the service without the need of laborious process of image building. Every 10 minutes the DLS instance checks if new versions of pipelines are available in the pipelines repository and deploy them in the service. The code repository for the pipelines also includes a verification pipeline. This pipeline conducts unit tests, but also more sophisticated integration tests, for instance the ssh2ssh pipeline

is verified to work against an instance of SSH server. With this approach it is possible to provide much farther-reaching guarantees of the functionality and reduce risks of modification that break down working pipelines.

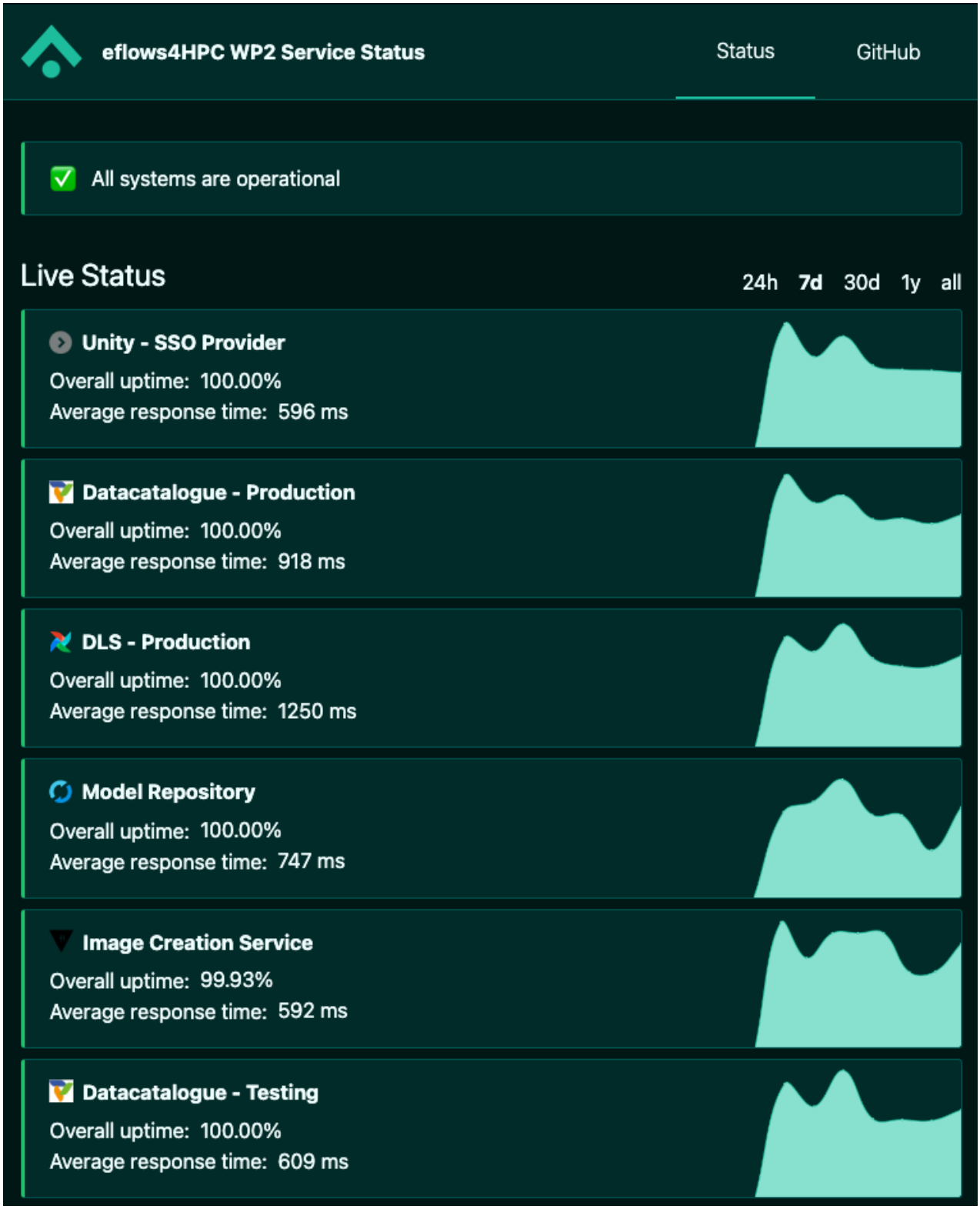

Figure 3 Monitoring solution

To put the benefits of the CI solution and the usage of the DLS in the project, let us analyze some basic numbers. We have two deployments of the DLS in eFlows4HPC. One in FZJ and one in BSC. Single pipelines are composed of multiple tasks, below we show the number of executed pipelines and number of tasks.

FZJ instance (https://datalogistics.eflows4hpc.eu/)

Pipelines runs: 479

Executed tasks: 2398

BSC instance (https://eflows4hpc.bsc.es/datalogistics)

Pipelines runs: 320

Executed tasks: 870

As explained above the development of the DLS and its pipelines was split in two repositories to account for different development speeds and access rights.

DLS-pipelines (https://github.com/eflows4hpc/dls-dags)

178 commits

218 CI/CD runs for the integration and unit tests

The DLS service (https://gitlab.jsc.fz-juelich.de/eflows4hpc-wp2/data-logistics-service/)

427 Commits

447 Deployment

4 Conclusion

This deliverable summarizes the development of the Data Logistics Service in eFlows4HPC. The developments can be subsumed in following groups. Firstly, a set of data pipelines (formalization of the data movements) were implemented. They are then used in the Pillars' workflows to describe what data and where needs to be moved. Secondly, a set of changes in the service alone was implemented (e.g., changes in the UI). Lastly, the DLS was integrated with external services to provide a seamless experience of running the workflows. This included integration with the workflow orchestrator, the image creation service, number of data repositories, etc. but also a Single Sign On solution for authentication. Lastly, an effort was invested in creating continuous deployment solutions and monitoring to achieve service maturity.

5 Acronyms and Abbreviations

- D – Deliverable
- DSL – Data Logistics Service
- HPC – High Performance Computing
- HPCWaaS – HPC Workflow-as-a-Service
- M – Month
- WP – Work Package
- WPL – Work Package Leader
- SSH – Secure Shell Protocol
- SSO – Single Sign-On

6 References

[1] eFlows4HPC, "DLS Pipelines repository," [Online]. Available: https://github.com/eflows4hpc/dls-dags.

[2] HashiCorp, [Online]. Available: https://www.vaultproject.io.

[3] "Unity Identity Relationship Management Software," [Online]. Available: https://unity-idm.eu/.

[4] eFlows4HPC, "eFlows4HPC infrastructure monitoring," [Online]. Available: https://cboettcher.github.io/eflows4HPC_WP2_Service_Monitor/.