H2020-EINFRA-2017

EINFRA-21-2017 - Platform-driven e-infrastructure innovation

DARE [777413] "Delivering Agile Research Excellence on European e-Infrastructures"

D4.8 Integrated Software Stack & Semantic Registry II

Project Reference No	777413 — DARE — H2020-EINFRA-2017 / EINFRA-21-2017
Deliverable	D4.8 Integrated Software Stack & Semantic Registry II
Work package	WP4: Big Data Processing and Analytics
Tasks involved	T4.5
Type	DEM: Demonstrator, pilot, prototype
Dissemination Level	PU = Public
Due Date	31/12/2020
Submission Date	30/12/2020
Status	Draft
Editor(s)	Sissy Themeli (NCSR-D), Iraklis Klampanos (NCSR-D)
Contributor(s)	
Reviewer(s)	KIT
Document description	This report accompanies the software deliverable of the Integrated Software Stack built in the framework of the DARE project. It provides a short description of the rationale and integration status while it indicates links with the work performed at other technical and architectural packages. In addition, it includes links to relevant code and documentation.

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
1	09/09/2020	Initial version	Sissy Themeli (NCSR-D)
2	1/12/2020	Additions on registries and modifications throughout	Iraklis Klampanos (NCSR-D)
3	30/12/2020	Final version	Iraklis Klampanos (NCSR-D)

Executive Summary

The document serves as a summary of the technical developments towards the final DARE integrated platform. Reported results span across all tasks of WP4, Big Data Processing and Analytics, where most of the technical work on individual constituents of the platform is carried out. It also refers to results obtained from WP3, Large-scale Lineage and Process Management, mainly pertaining to the incorporation of provenance into the platform and the establishment of API mechanisms for its various parts. Finally, the report specifies the relationship and conformance of the platform to the architectural design envisioned in the proposed DARE solution.

Table of Contents

1 Introduction *5*

1.1 Purpose and Scope **5**

1.2 Relationship with other Work Packages and Deliverables **5**

1.3 Methodology and Structure of the Deliverable **5**

2 DARE Software Stack & Components *6*

2.1 Core DARE Components **6**

2.1.1 dispel4py 6
2.1.2 s-ProvFlow 6
2.1.3 d4py-registry 6
2.1.4 CWL workflow registry 7
2.1.5 DARE deployment configuration (extending Big Data Integrator) 7
2.1.6 DARE Execution API 7

2.2 Supporting DARE Components **7**

2.2.1 Playground 7
2.2.2 MySQL 7
2.2.3 MongoDB 7
2.2.4 Virtuoso 7
2.2.5 SemaGrow 8
2.2.6 ExaSpark 8

2.3 Deployment Testbed **8**

2.3.1 Operational Deployment 8

3 DARE Registries *10*

3.1 DARE disel4py information registry **10**

3.2 DARE CWL registry **10**

3.3 Data Ontology **10**

3.4 Semantic data catalogue and search **11**

4 Summary *11*

5 References *11*

List of Terms and Abbreviations

Abbreviation	Definition
WaaS	Workflows-as-a-Service
DKB	DARE Knowledge-base
P4	Protected Pervasive Persistent Provenance
GDBMS	Graph Database Management System
SQL	Structured Query Language
RDF	Resource Description Framework
PE	Processing Element
API	Application Program Interface
FOAF	Friend-Of-A-Friend ontology
DCAT	Data Catalogue vocabulary

1 Introduction

As an update to deliverable D4.7 'Integrated Software Stack and Semantic Registry I' this document serves as an incremental summary of the technical developments leading to the implementation of the final DARE integrated platform – i.e., the complete DARE platform. Reported results span across all tasks of WP4, Big Data Processing and Analytics, where most of the technical work for individual constituents of the platform has been carried out. This report also refers to the results obtained from WP3, Large-scale Lineage and Process Management, mainly pertaining to the incorporation of provenance into the platform and the establishment of API mechanisms for its various parts. The integration process has been designed to set relatively low barriers and limits for integrating additional components and/or external resources. This design has made the integrated platform extensible to different scientific/user communities and adaptable to future evolutions in data science and e-infrastructure technologies.

1.1 Purpose and Scope

The purpose of this report is to provide a final account of the DARE platform in combination with the software delivered. It provides links to relevant code repositories and documentation, places the rationale for creating the Integrated Stack under the general approach of DARE and indicates the plan for evolving the platform.

1.2 Relationship with other Work Packages and Deliverables

The integrated software stack realised as the DARE platform is collectively one of the main technical outcomes of the DARE project as it implements the DARE architecture of WP2. Furthermore, it serves as the basis for technical improvements as well as for implementing the DARE use-cases of WPs 6 and 7. It constitutes an asset to be further developed and exploited by the DARE partners as described in deliverable D8.6 'Sustainability,
Exploitation and Commercialisation Plan'.

1.3 Methodology and Structure of the Deliverable

This report accompanies a software deliverable comprising the components incorporated in the DARE platform in accordance to the DARE architectural principles and the project goals.

Section 2 presents a high-level technical overview and summarises the components included in the first deployment of the DARE software stack. Section 3 presents the high-level ontologies designed for describing the respective components, as well as a preliminary implementation of the ontology used to describe data assets handled by DARE. Finally, Section 4 summarises the work described in this report.

2 DARE Software Stack and Components

This section describes the current set of components integrated in the DARE platform. We distinguish between *core* components - which are implementations of the major architectural elements of DARE as described in deliverable D2.2 'DARE Architecture and Technical Positioning II' - and *supporting* components - that encapsulate required functionalities towards a functioning platform - which also holds since the first version of this deliverable, i.e., D4.7 'Integrated software Stack and Semantic Registry I'. The latest stable version of the platform is also described in [1].

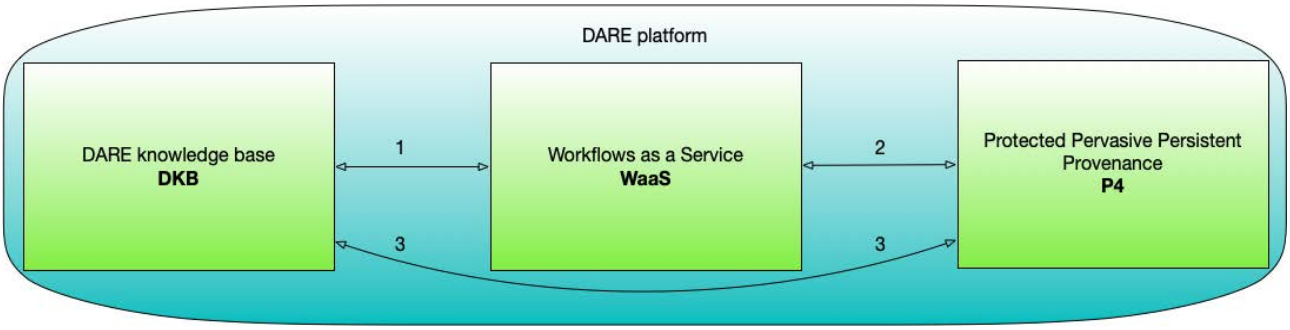

Figure 1: High-level DARE Architecture[1]

The DARE platform implements the DKB as a set of decoupled registries/stores: two separate workflow registries. Each of these exposes APIs for other components to interact with, following a microservices design[2]. The provenance store P4 is similar in the way it is implemented as a microservice.

WaaS drives interactions with the DKB and P4, for example obtaining components and workflow implementations to execute, information about the environment or providing information to the provenance system. It is worth noting that the Kubernetes[3] registries may also be interrogated by other DARE subsystems, therefore effectively being part of the DKB.

2.1 Core DARE Components

2.1.1 dispel4py

dispel4py acts as the current implementer of the WaaS concept in the DARE architecture. It is used to describe abstract workflows and enact them over diverse underlying infrastructures taking care of the distribution of the execution, its optimisation and the orchestration of the different components involved in the execution. The repository hosting the latest implementation of dispel4py is accessible at:
https://gitlab.com/project-dare/dispel4py.

2.1.2 s-ProvFlow

This component realises the P4 component of DARE that is responsible for collecting, preserving and reporting on provenance information from the workflow execution over the platform. Details on its functionality may be found at: https://gitlab.com/project-dare/s-ProvFlow.

2.1.3 d4py-registry

Through d4py-registry one can find a set of resources that include connections, function implementations, function parameters, groups, literals, PE implementations, user groups, users and workspaces, and a list of possible actions to apply on the resources. The interface is documented in

[1] D2.2 'DARE Architecture and Technical Positioning II'
[2] https://en.wikipedia.org/wiki/Microservices
[3] https://kubernetes.io/

order to ease the user experience while the implementation is stored at: https://gitlab.com/project-dare/d4py-registry.

2.1.4 CWL workflow registry

This component is used to register CWL workflows and relevant execution environments to enable the execution of CWL workflows. It uses a MySQL database in the backend and exposes its functionality via a RESTful API. The component is containerized and may be deployed using the relevant specification files provided in the dare-platform repository. The CWL Workflow Registry repository is available at: https://gitlab.com/project-dare/workflow-registry

2.1.5 DARE deployment configuration (extending Big Data Integrator)

The DARE platform is a customised, cloud-ready and modular integrator platform, bringing together commercial and research, production-ready components for big data analytics. It offers an easy-to-deploy, easy-to-use and adaptable framework for the execution of big data applications by exposing big data tools as ready-to-use Docker Compose[4] files while its installation and deployment are described in deliverable D4.2 'Big Data Analytics Toolkit II'. The DARE platform repository is available at: https://gitlab.com/project-dare/dare-platform that is the starting point for deployment and installation. Comprehensive documentation may be found at: https://project-dare.gitlab.io/dare-platform/.

2.1.6 DARE Execution API

This component exposes a RESTful API needed for the instantiation of the execution environments for the dispel4py and CWL workflows. It also provides functionality to the users for listing their folders and files, uploading and downloading files etc. The Execution API repository is available at: https://gitlab.com/project-dare/exec-api.

2.2 Supporting DARE Components

2.2.1 Playground

This component provides the domain developers with a testing environment for workflow development. It exposes a RESTful API for a dispel4py execution simulation as well as for terminal simulation allowing the users to execute simple terminal commands. The code repository for the DARE Playground module is available at: https://gitlab.com/project-dare/playground.

2.2.2 MySQL

MySQL[5] databases are used in the Dispel4py and CWL Workflow Registries to store workflow files, docker environments or workspaces, user information etc.

2.2.3 MongoDB

MongoDB[6] is used in the Provenance component to store provenance traces from the workflow executions.

2.2.4 Virtuoso

Virtuoso[7], due to its maturity and flexibility, was selected as the GDBMS to be used for the DARE platform. Virtuoso provides a hybrid server architecture for data access, virtualization, integration and multi-model relational database management (SQL Tables and/or RDF Statement Graphs). In the following link we provide a Docker image for the respective software component along with a Dockerfile and a script for configuration: https://gitlab.com/project-dare/docker-virtuoso.

[4] https://docs.docker.com/compose/
[5] https://www.mysql.com/
[6] https://www.mongodb.com/
[7] https://virtuoso.openlinksw.com/

2.2.5 SemaGrow

SemaGrow implemented a federation engine for DKB in the initial DARE versions. While not in use in later platform versions , from the architect point-of- view it remains an important component for providing a unifying view to multiple internal and external semantic resources, should the need arise in the future. The repository hosting the latest implementation of SemaGrow can be found at: https://gitlab.com/project-dare/semagrow.

2.2.6 ExaSpark

ExaSpark is available within the DARE architecture and can be potentially invoked via CWL though it is not currently used by any use-case. Related information may be found at: https://gitlab.com/project-dare/ExaSpark.

2.3 Deployment Testbed

DARE makes use of testbeds prepared and maintained by Fraunhofer-SCAI and GRNET. GRNET has provided a development testbed while SCAI has provided an operational testbed which has also been used during demonstrations and webinars, as reported in deliverables D6.4 'Pilot Tools and Services, Execution and Evaluation Report II', D7.4 'Pilot Tools and Services, Execution and Evaluation Report II' and D8.5 'Training and Consulting Report II'. The containers are managed and orchestrated via Kubernetes exposing information on the status and availability of each container via its native API. The API is used for updating the relevant entries in the Semantic Registry as summarised in the following section.

2.3.1 Operational Deployment

As far as it concerns the operational testbed provided by SCAI the DARE platform deployment can be performed using the instructions listed in deliverable D4.2 'Big Data Analytics Toolkit II', which are also available in the DARE platform microsite: https://project-dare.gitlab.io/dare-platform/installation/. The Kubernetes specification files for platform deployment in the operational environment are available at: https://gitlab.com/project-dare/dare-platform/-/tree/master/k8s-operational while users may interact with the platform with JupyterLab (hosted at: https://jupyter.dare.scai.fraunhofer.de/) using their DARE credentials. JupyterLab maintains a copy of the DARE repository with all use-cases (https://gitlab.com/project-dare/dare-examples).

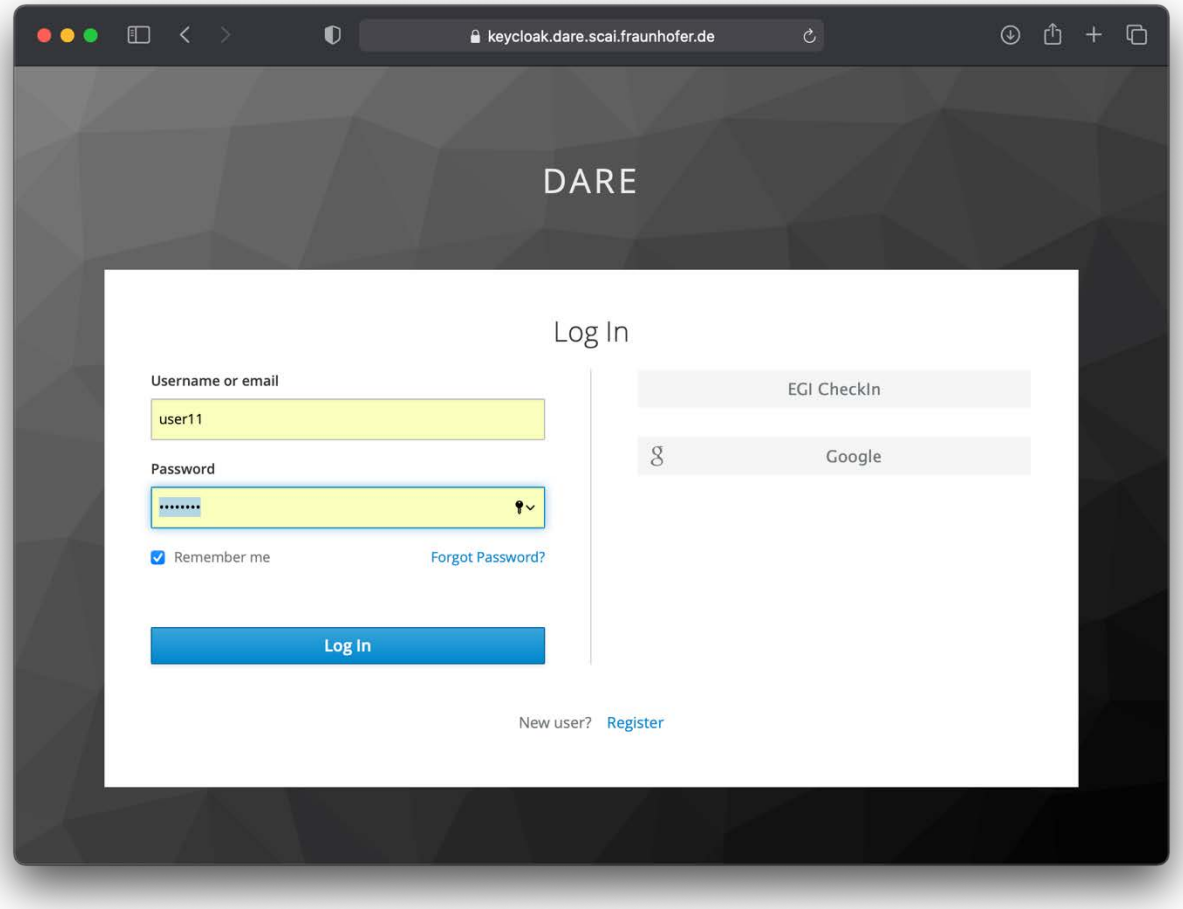

Figure 2: Login screen at the SCAI operational installation

3 DARE Registries

This section provides an overview of our approach for conceptualizing - at a generic level - the workflows and methods implemented as well as the data assets to be handled by these during the workflow execution on the platform. The DARE registries cover dispel4py and CWL workflows accompanied by an ontology for data assets used by DARE.

3.1 DARE disel4py information registry

The dispel4py Registry is a RESTful Web service providing functionality for registering workflow entities, such as PEs, functions and literals, while encouraging sharing and collaboration via groups and workspaces. More information is provided in [2] as well as at the corresponding repository: https://gitlab.com/project-dare/d4p-registry.

3.2 DARE CWL registry

The CWL Workflow Registry provides a similar functionality to the Dispel4py Registry while it is associated with CWL workflows. More information is provided at the corresponding source code repository: https://gitlab.com/project-dare/workflow-registry.

3.3 Data Ontology

The core entity of the ontology that conceptualizes data assets handled by DARE is the *Dataset*, that entails generic information about the asset. Its origin is determined via a link to a *Device* entity that models, at an abstract level, data-producing equipment (e.g. seismometers) placed in a specific *Location*.

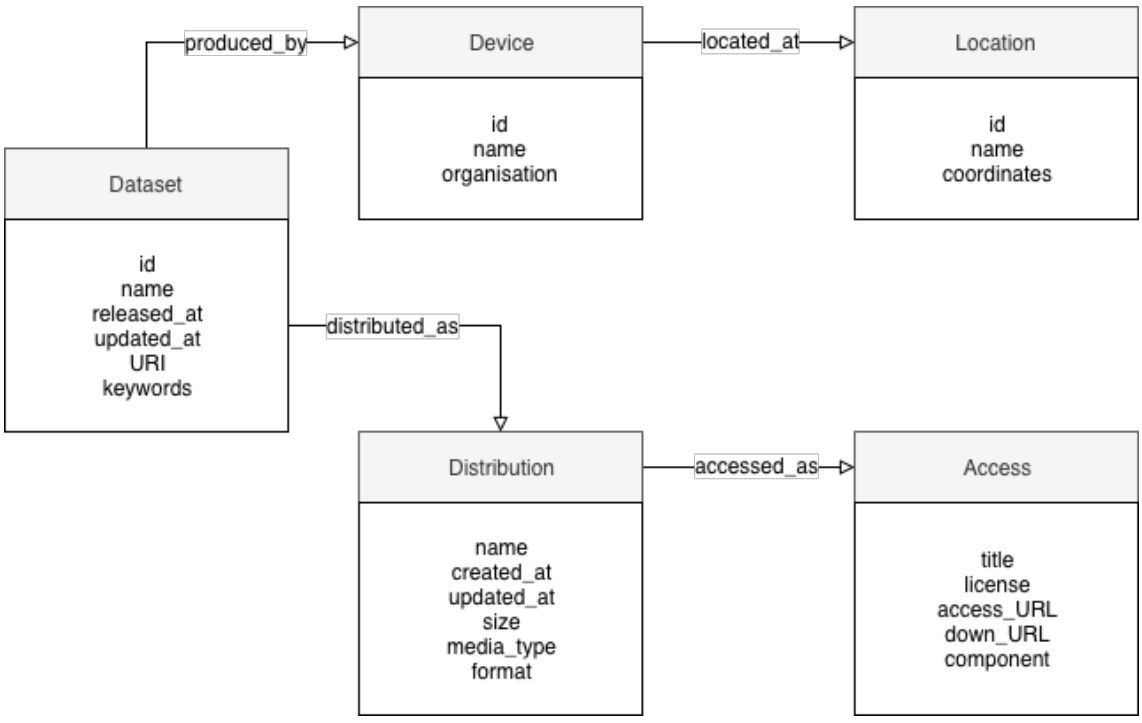

Figure 3: Data Ontology - Core Entities

Additional information that pertains to the rights and means for using the dataset is conceptualized as instances of the *Distribution* and *Access* classes. *Distribution* provides technical information for the dataset (format, type, size, etc.), while *Access* models licensing information and ways to consume the data resource (access URL, download URL). Furthermore, it links the resource with the *SoftwareComponent* that is able, and expected, to use the dataset.

The OWL ontology that materializes the described schema is available through the relevant Git repository of the project[8] and it is based on DCAT and FOAF. This led to the implementation of a semantics-based data catalogue and search component that is presented below.

3.4 Semantic data catalogue and search

The Semantic Data Catalogue is a Flask[9] Web Service which uses Virtuoso and Solr[10] as backends[11] where user data is stored in Virtuoso and indexed in Solr. The API exposes some functionalities to the users, such as searching data.

4 Summary

This report summarises the status of the Integrated Software Stack at the end of the DARE project. Stable versions of the core components, as reported in section 2, have been integrated in the final version of the DARE platform stack. All assets of the platform (code, containerized components, ontologies, schemas, etc.) are maintained as GitLab repositories along with their documentation.

5 References

[1] Klampanos et al., (2020). DARE Platform: a Developer-Friendly and Self-Optimising Workflows-as-a-Service Framework for e-Science on the Cloud. Journal of Open Source Software, 5(54), 2664, https://doi.org/10.21105/joss.02664

[2] Klampanos et al., (2019, August 6). Consistency and Collaboration for Fine-Grained Scientific Workflow Development: The dispel4py Information Registry. Zenodo. http://doi.org/10.5281/zenodo.3361395

[8] https://gitlab.com/project-dare/data-catalogue/-/blob/master/dare-data.owl

[9] https://flask.palletsprojects.com/en/1.1.x/

[10] https://lucene.apache.org/solr/

[11] https://gitlab.com/project-dare/semantic-data-discovery