H2020-EINFRA-2017
EINFRA-21-2017 - Platform-driven e-infrastructure innovation
DARE [777413] "Delivering Agile Research Excellence on European e-Infrastructures"

Integrated Software Stack & Semantic Registry I

Project Reference No	777413 — DARE — H2020-EINFRA-2017 / EINFRA-21-2017
Deliverable	D4.7 Integrated Software Stack & Semantic Registry I
Work package	WP4: Big Data Processing and Analytics
Tasks involved	T4.5, T2.1, T3.2, T3.3, T3.4, T4.1, T4.2, T4.3, T4.4, T5.1
Type	DEM: Demonstrator, pilot, prototype
Dissemination Level	PU = Public
Due Date	31/12/2018
Submission Date	31/12/2018
Status	Draft v1.0
Editor(s)	Antonis Koukourikos (NCSR-D)
Contributor(s)	NCSR-D, UEDIN, KNMI, ATHENA-RC
Reviewer(s)	Federica Magnoni (INGV)
Document description	The report accompanies the software deliverable for the Integrated Software Stack delivered by DARE. It provides a short description on the rationale and state of the integration and indicates links with work on other technical and architectural packages. It also provides links to relevant code and documentation.

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
0.1	30/11/2018	ToC	Antonis Koukourikos
0.5	07/12/2018	Sections 2 and 3	Antonis Koukourikos
0.7	14/12/2018	Section 4	Antonis Koukourikos
0.8	20/12/2018	Submitted for internal review	Antonis Koukourikos
0.9	21/12/2018	Internal Review	Federica Magnoni
1.0	31/12/2018	Final version to be submitted to the EC	Mariana Markouli

Executive Summary

The document serves as a summary of the technical developments towards the final DARE outcome, the DARE integrated platform.
Reported results span across all tasks of Work Package 4, Big Data Processing and Analytics, where most of technical work for individual constituents of the platform is carried out.
It also refers to results obtained from Work Package 3, Large-scale Lineage and Process Management, mainly pertaining to the incorporation of provenance in 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 for the proposed DARE solution.
The whole integration process is designed to set relatively low barriers and limits for integrating additional components and/or external resources. Hence the adoption of containerization and the requirement initial metadata at a high level for the incorporation of a resource.
The design is expected to help technical work during the duration of the project as well as – and more importantly – to constituting the platform extensible to further, different scientific and user communities and adaptable to future evolutions in data science and e-infrastructures technologies.

Table of Contents

List of Figures

List of Terms and Abbreviations

Abbreviation	Definition
WaaS	Workflows-as-a-Service
C4	Common Conceptual Core Catalogue
P4	Protected Pervasive Persistent Provenance
GDBMS	Graph Database Management System
SQL	Structured Query Language
RDF	Resource Description Framework
PE	Processing Element
Fraunhofer-SCAI	Fraunhofer Institute for Algorithms and Scientific Computing
API	Application Program Interface
FOAF	Friend Of A Friend

1 Introduction

The present document is an accompanying report to the first deployment of the DARE Software Stack.

1.1 Purpose and Scope

The purpose of the report is to summarize the progress on the DARE platform, clarify decisions, identify risks and critical points, and present the plan for the forthcoming months.
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 Approach and relationship with other Work Packages and Deliverables

D4.7, a direct deliverable of T4.5, *DARE Software Stack Release*, is the culmination of the technical work in DARE, as it encapsulates the activities in multiple work packages and tasks towards an integrated platform.
The blueprint for the Integrated Software Stack is provided by WP2 and specifically, D2.1, *DARE Architecture and Technical Positioning*.
Resources and components pertaining to the physical layer used by the platform are made available via work carried out in WP5 and are reported in detail in D5.1, *Platform Infrastructure, Usage and Deployment*.
The distinct components incorporated in the Integrated Software Stack are developed, updated and maintained in various tasks of Work Packages 3 and 4, namely:
T3.2 – Reproducibility and Lineage Services
T3.3 – User-driven Control-enabling API
T3.4 – Tooling Integration
T4.1 – Big Data Analytics Toolkit
T4.2 – Data-driven Abstraction Specification Toolkit
T4.3 – Data Consolidation and Linking Toolkit
T4.4 – Execution Mapping Services

1.3 Methodology and Structure of the Deliverable

The report accompanies a software deliverable comprising the components thus far incorporated in the DARE platform, under the guidelines and design set by the DARE architectural principles. It aims to showcase the conformance with the goals of the project and the designed architecture for realizing these goals.
Section 2 presents a high-level technical overview and summarizes 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 that will be used to describe data assets to be handled by DARE. Section 4 reports on future steps, foreseen risks and the respective mitigation measures.

2 DARE Software Stack & Components

The section describes the current set of components integrated in the DARE platform. We can distinguish between *core* components, which are implementations of the major architectural elements of DARE, as described in deliverable D2.1, and *supporting* components, that encapsulate required functionalities towards a functioning platform.

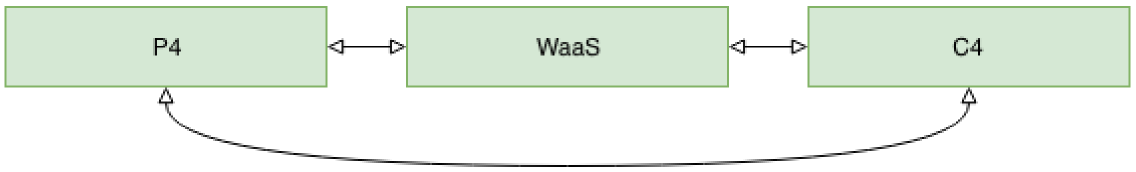

Figure 1: High-level DARE Architecture

From a technical standpoint (see Figure 2), C4 comprises a Federator module which is responsible for accessing the underlying database where DARE-internal information is stored, as well as, external collections and repositories of relevance to the use cases executed through DARE.
P4 communicates with WaaS to be informed about execution details that are included in the respective provenance record and stored through the Federator in C4. Similarly, WaaS is informed by C4 (again via the Federator) for invoking data and components that are required for the execution of a given workflow.

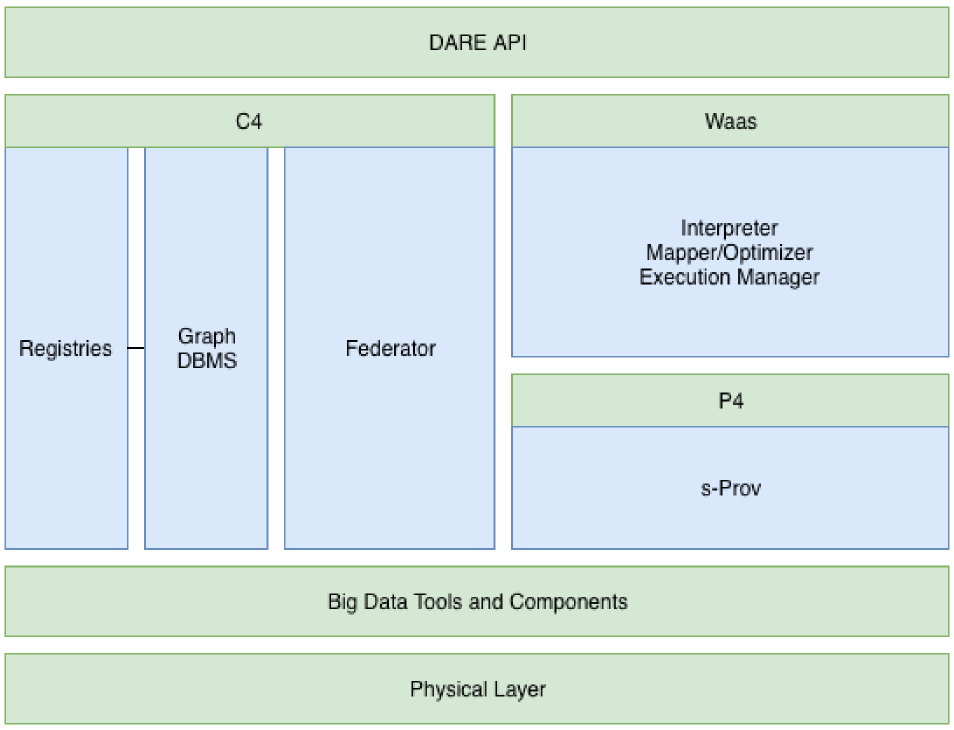

Figure 2: DARE Technical Overview

2.1 Core DARE Components

2.1.1 dispel4py

dispel4py acts as the current implementor 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 optimization 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.
The master branch can be cloned from https://gitlab.com/project-dare/dispel4py.git

2.1.2 s-ProvFlow

The component realizes the P4 component of DARE, responsible for collecting, preserving and reporting on provenance information from the workflow execution over the platform. Details on its functionality can be seen in the respective repository at: https://gitlab.com/project-dare/s-ProvFlow.

The master branch can be cloned from https://gitlab.com/project-dare/s-ProvFlow.git

2.1.3 SemaGrow

SemaGrow is the realization of the Federation engine for C4 in the first version of DARE.
The repository hosting the latest implementation of SemaGrow can be found at:
https://gitlab.com/project-dare/semagrow.
The master branch can be cloned from https://gitlab.com/project-dare/semagrow.git.

2.1.4 ExaSpark

ExaSpark is also part of the WaaS implementation within the DARE architecture. For more information search the GitLab repository at: https://gitlab.com/project-dare/ExaSpark.
The master branch can be cloned from https://gitlab.com/project-dare/ExaSpark.git.

2.2 Supporting DARE Components

2.2.1 Virtuoso

Virtuoso, 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.

2.2.2 d4py-registry

Through d4py-registry, one can find a set of resources, which 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 order to ease the user's experience and the implementation can be found at:
https://gitlab.com/project-dare/d4py-registry.
The master branch can be cloned from https://gitlab.com/project-dare/d4py-registry.git.

2.2.3 Components registry

The components registry is responsible for gathering and preserving information on the status and availability of components deployed or accessible from the DARE platform. Its model and communication with the deployment environment is summarized in section 3.1 of this document.
Information on the registry can be found at: https://gitlab.com/project-dare/components-registry
The master branch of the relevant Git repository can be cloned from https://gitlab.com/project-dare/components-registry.git.

2.2.4 BDI

The Big Data Integrator 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 files.
A list of applications provided by BDI is reported in D5.1, and indicatively includes technologies like Hadoop, Spark, Flink, Kafka, GeoTriples, and others.

2.3 Deployment Testbed

For the first year of the project, DARE utilized infrastructures provided, prepared and maintained by Fraunhofer-SCAI and GRNET, as reported in deliverable D5.1. The Docker images of the aforementioned components, along with the services provided by the Big Data Integrator platform, were deployed over the infrastructures.

The containers are managed and orchestrated via Kubernetes, which exposes 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.

3 DARE Semantic Registries

The section provides an overview of our approach for conceptualizing at a generic level the components incorporated in the DARE platform and the data assets to be handled by these during the execution of workflows over the platform. It is expected that the core model will be extended as the component collection is enriched or modified and specific data assets are requested by the use cases.

Furthermore, an important part of the conceptualization task is the linking with existing ontologies used by external repositories that are relevant to DARE. The linking will be carried out mainly on demand (as data sources are identified by the use cases) and in some cases, proactively (links to major Linked Data sources and widely used schemas like schema.org, FOAF, etc.).

All registries are part of the Federation controlled by SemaGrow in the C4 conceptual component of the DARE architecture and are exposed through the appropriate APIs, as reported in deliverable D3.5, DARE API.

3.1 Components Ontology

The main entity in the ontology is *SoftwareComponent*, which conceptualizes a generic software entity deployable and usable within the platform. From an implementation point of view, the class is not expected to have direct instances, as each software component will be an instantiation of a specialization of the *SoftwareComponent* class (indicatively, a Database, a Filesystem, etc.).

In accordance with the DARE approach, all components are deployed as Docker containers. Thus, a *SoftwareComponent* individual is related to its Container, an entity carrying the basic characteristics that define an image. The container is hosted in a specific *Host*, from where it is actually accessible.

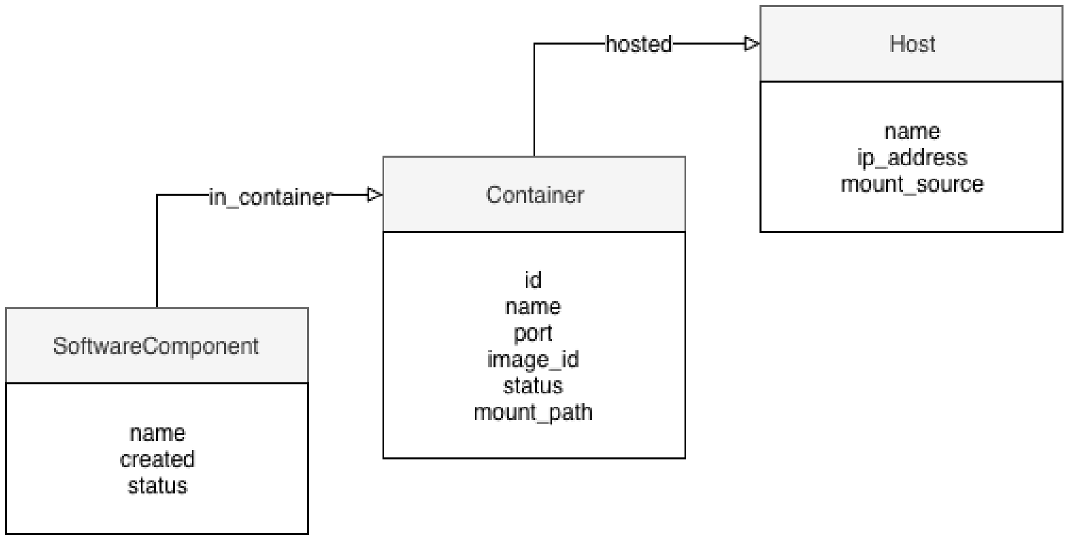

Figure 3: Components Ontology - Basic Entities

At the functional level, the information on the component registry is updated via the relevant Kubernetes managers. Hence, a mapping between the metadata exposed by Kubernetes and the Components ontology has been designed and can be found at:
https://gitlab.com/project-dare/components-registry/blob/master/kubectl-components-mapping.docx
The ontology itself is available at:
https://gitlab.com/project-dare/components-registry/blob/master/components.owl

3.2 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, which models at an abstract level data-producing equipment (e.g. seismometers), placed in a specific *Location*.

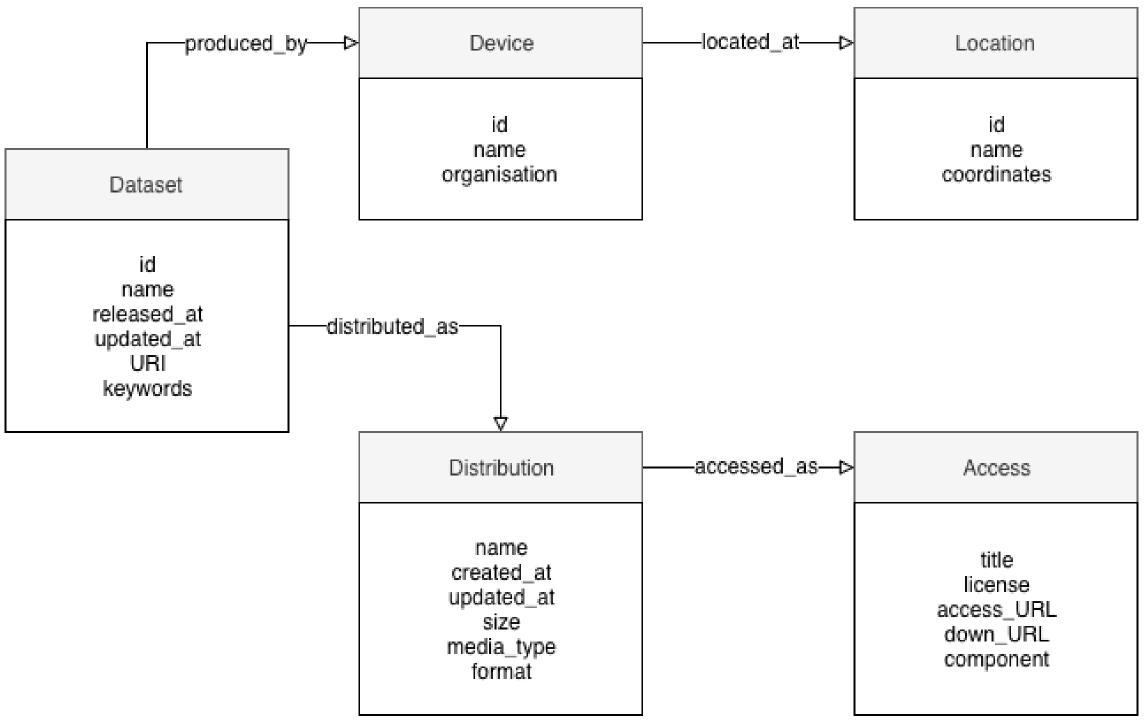

Figure 4: Data Ontology - Core Entities

Further 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* (as defined in the Components ontology) 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 at:

https://gitlab.com/project-dare/data-registry/blob/master/Data_Schema.owl

4 Summary and Next Steps

The present document summarizes the state of the DARE Integrated platform at the end of M12 of the project. The core principles for the integration, as reported in section 2, are expected to remain relatively stable throughout the duration of the project. The same holds for the overall technical approach and technical project management in DARE. That is, all assets of the platform (code, containerized components, ontologies, schemas, etc.) will be maintained as GitLab projects along with their documentation.
These assets will however be continuously updated and calibrated as the use cases of the project are executed and consequently extended and refined. It is probable that additional components, new modules and more sophisticated and specialized schemas will be required in the forthcoming periods. The Integrated Software Stack will be updated accordingly, and major changes will be reflected in reports similar to this.