HEALTHYCLOUD
Health Research & Innovation Cloud

D5.2 Analysis of existing orchestration mechanisms for distributed computational analyses including a general overview to facilitate new developments

Document Information

Contract Number	965345
Project Website	http://www.healthycloud.eu/
Contractual Deadline	M18, August 2022
Dissemination Level	PU
Nature	R
Author(s)	Rosa M Badia (BSC)
Contributor(s)	
Reviewer(s)	Marco Roos (LUMC), Harald Wagener (CHARITE)
Keywords	Workflow environments, orchestration, distributed computing

Notice: *The HealthyCloud project has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement №965345*

Change Log

Version	Author	Date	Description of Change
V0.1	Rosa M Badia	05/07/2022	Table of Content
V0.2	Rosa M Badia	19/07/2022	First version
V0.3	Rosa M Badia	26/08/2022	Version after review by Harald Wagener
V1.0	Rosa M Badia	29/08/2022	Version after reviews
			(Final Change Log entries reserved for releases to the EC)

Table of contents

Executive Summary

This deliverable is the output of task 5.2 *Understanding the challenges for distributed computational analysis across Europe.* The task activities started defining a set of aspects for which the project would like to investigate their status in existing workflow and orchestrator systems. These aspects relate to the distribution of data and computation, to the support to sensitive data, and to the support for data provenance and reproducibility. In addition to questions related to these topics, other generic questions were put all together. An initial classification of workflow environments and orchestrators was also done, and from there the task searched for information about the different systems. The information was provided in some cases by project partners, in other cases by system owners and in other cases fetched by partners from websites or other sources of information.

From this initial catalogue, the partners involved in this activity selected the most promising ones that were invited to present in two workshops. The workshops were online and organized quite informally, but enabled a lot of discussion and provided in-depth information on the topics more relevant to the project.

The information collected in the two phases of the task (catalogue elaboration and workshops) is summarized and analysed in this deliverable.

1. Introduction

The work has been developed on the framework of the *WP5 Designing a decentralized cloud for health data research*. This work package aims to deepen knowledge and establish scenarios on the technological challenges for enabling an information and communications technology (ICT) infrastructure that allows secure access to sensitive health research data across Europe. Such infrastructure will be decentralised by design covering a broad range of infrastructures including public and private cloud, edge, high-performance computing as well as on-premises computing.

Another important layer of complexity to take into account is the deployment of analytical software resources and the execution of workflows across the different computational instances taking into consideration all the legal and ethical limitations imposed by European regulations (GDPR) and existing national legislations.

More specifically, this deliverable is the output of task *5.2 Understanding the challenges for distributed computational analysis across Europe*, which aims at surveying the existing orchestration mechanisms, e.g workflows managers, that enable distributed data-centric analysis for health research. The task has performed a general revision of the different aspects in distributed health applications, such as data and computation distribution, management of sensitive data, and how are they approached in the orchestration mechanisms. An important aspect that has also been considered in the task is how to handle properly data provenance as data transformation is an essential part of analysis reproducibility.

Section 2 presents the methodology used in the preparation of this deliverable, section 3 gives an overview of existing workflow and orchestration systems, section 4 presents with more detail some of the environments that were presented in the workshops organized by task 5.2 and section 5 concludes the deliverable.

2. Methodology

With the goal of surveying the existing approximations (workflow environment and orchestrators) that can enable a distributed data-centric analysis for health applications, the following set of activities were performed:

1. Definition of the classification and information that it would be queried for each workflow environment or orchestrator
2. Identification of existing environments and systems and obtention of answers for the different fields
3. Identification of environments that were more promising for the project purposes
4. Organization of workshops with stakeholders to obtain more in-depth information

5. In-depth analysis of the results

The first activity consisted of defining a classification for the different environments. The following categories were defined: a) application oriented, b) infrastructure oriented, c) container environments and d) job managers. In addition, a set of questions to be answered for each environment was defined, that included more general questions such as website, license, user community, or maturity level, plus other more specific to the purposes of the project:

- Whether it has been used with health data and which data was used
- If it provides support for distributed data and a short summary of how this is achieved
- If it provides support for execution in multi-domain computing infrastructure
- If it provides support for sensitive data and how this is dealt
- Whether it provides support for reproducibility/provenance and with which mechanisms (i.e., containers, etc)

An online spreadsheet was set-up to include all information.

While the list of potential workflow environments and orchestrators is very long[1], the project partners identified environments that could be relevant for the project purposes.

The answers to the questions about the different systems were provided in some cases by the project partners and in other cases a questionnaire was sent to the tools owners (see Annex I) to collect the answers.

Once this step was finished, we did a selection of the workflow environments and orchestrators that we considered more relevant for the project goals. In this sense, the answers to the specific project-related questions were considered.

Two rounds of workshops were organized with invited speakers from the relevant selected systems. See Annex II and Annex III the agendas of the workshops. In each of the workshops, the speakers were asked to do a concise presentation of their systems, focusing on the following points:

- Overview of the tool
- How distributed data and computation is managed
- How sensitive data is managed

[1] https://github.com/common-workflow-language/common-workflow-language/wiki/Existing-Workflow-systems

- Overview of how reproducibility/provenance is handled
- Description of 1-2 success stories

In addition to the presentations, sessions of questions and answers were held after each presentation and a final discussion session.

We acknowledge Paolo di Tommaso (Seqera Labs), Ivo Buchhalter (DKFZ), Daniele Spiga (INFN), Daniele Lezzi (BSC), Karan Vahi (ISI, USC), Tamas Kiss (UOW), Iacopo Colonnelli (UNITO), and Jose M Fernandez (BSC) for contributing to the workshops with their presentations and fruitful discussion.

3. Overview of existing workflow and orchestrator environments

Distributed computing has been evolving in the recent years and growing in complexity. The old grid computing environments, composed of a set of computing servers distributed geographically were superseded by the cloud computing paradigm, which together with supercomputers and large clusters provide computing and data intensive systems for a wide category of scientific and industrial applications. However, the current scene for distributed computing does not only consider the large computing systems, but also IoT (Internet of Things) devices, like sensors or large instruments, and also devices on the edge that can be both sources of data and provide computation capacities. In this sense, the term computing continuum has been defined as the infrastructure that considers all these devices.

Applications have also evolved to leverage these complex infrastructures, becoming over time more complex and demanding on the software underlying that needs to support their execution. A special type of applications are the workflows, that combine different components providing diverse functionalities which may exchange large number of data items. Workflows require complex management systems to orchestrate their execution in distributed computing infrastructures.

As previously mentioned, workflow management systems and orchestrators in general can be classified as follows:

- **Application oriented**: environments that offer a developer interface for workflow applications. The type of interfaces can be graphical, textual or programmatic. Some environments focus in specific areas of application. Initially, we further decomposed this category in application specific areas such as health oriented, life sciences oriented or application agnostic. However, a single category was decided later since this was not so relevant to the purposes of the task. Examples of this category are NextFlow, PyCOMPSs, Pegasus or StreamFlow.
- **Infrastructure oriented**: this category includes environments that enable the orchestration of distributed computing systems, enabling to manage the whole lifecycle application: deployment, execution, data stage-in and stage-

out, and undeployment. In this sense, some Application oriented environments can be delegating some or all of these functionalities to an infrastructure-oriented orchestrator. Examples of this category are the Infrastructure Manager, Yorc, or Occopus.

- **Container environments**: Container environments are a special type of infrastructure-oriented systems that enable the management of applications through the use of containers. Per se, a container environment is not an orchestrator, but some support container orchestration, such as Micado or Kubernetes.
- **Job managers**: In this category we consider those systems that manage the batch execution of applications in large clusters or supercomputers through queues. Some of these systems offer primitive workflow management, enabling to define that a job should run only after the completion of a previous one, for example. Examples of job managers are Slurm, LSF or grid engine.

From the set of systems considered, the following include features that were more interesting for the project purposes (application oriented systems are shown in blue and infrastructure oriented orchestrators are shown in orange):

System name	Support for distributed data	Support for execution in multi-domain computing infrastructure	Support for sensitive data	Support for reproducibility/ provenance
One Touch Pipeline (OTP)	N	N	Y, internal user management that is propagated to the file system using unix groups	Y, versioned workflows and configurarions
PyCOMPSs	Y, load and data is distributed between the different worker nodes. Locality is exploited to reduce data transfers	Y	Y, if used with dataClay, which can manage different user profiles	Y, support for containers, tasks in containers and the whole application in a container. Provenance provided through RO-Crate
DODAS	Y, includes deployment of embedded caching layer close to the compute resources	Y	N, it relies on iso-certified providers.	Y, support for docker container both for batch and interactive execution

Galaxy	Y	Y	Yes. Norway (TSD) and the broad have Galaxy running in those settings.	Yes. Galaxy is hosting and creating all the containers of BioContainers. All provenance is captured in a relational database.
Parsl	Y, supports a variety of data staging mechanisms to move data between storage locations and usage locations where tasks run, including transparent movement of Python objects and file transfer using scp, ftp, http, and Globus	Y, in experimental usage, and with some issues in production usage; funcX (which integrates with Parsl) is more reliable for this use case	N	Y, supports containers, monitoring system captures information about workflow execution
Pegasus	Pegasus adds data management tasks into the workflow that are responsible for retrieving data required for the jobs in the workflow from various data sources. Supported endpoints are HTTP, FTP, GridFTP, Globus Online, SCP, StashCache, HPSS, S3, Google Cloud Storage, etc.	Yes, workflows are regularly run on Open Science Grid (OSG) and LIGO Data Grid composed of multi-domain clusters. Workflows can also run in hybrid edge-cloud computing infrastructures.		Yes, runtime provenance stored in database, and also monitoring information stored in backends. Reproducibility: users can specify containers that are required. Pegasus automatically deploys the containers to the computing nodes.

StreamFlow	Y, automatically manages data transfers between subsequent workflow steps, without the need of a unique shared data space.	Y, can deploy environments, transfer data and offload computation to HPC queue managers, Cloud VMs and container orchestrators also in the context of a single workflow execution.	Partial, supports secure data movements through encrypted channels and a detailed control of the information flow by letting users explicitly modelling the mapping between workflow steps and executing locations.	Y, supports container executions for portability of results. The explicit inclusion of a deployment/connection plan for the execution environment fosters reproducibility and reusability. Finally, provenance is enabled by storing details about workflow execution.
Infrastructure Manager	N, not natively but any distributed system can be configured in the infrastructure template.	Y	N, not natively but it can be configured in the infrastructure template.	Y, Using the IaC paradigm the same infrastructure can be reproduced.
Elastic Compute Cluster EC3	N, not natively but any distributed system can be configured in the infrastructure template.	Y	N, not natively but it can be configured in the infrastructure template.	Y, Using the IaC paradigm the same infrastructure can be reproduced.
Indigo-PaaS	Y, includes native support for Onedata. Any other distributed system can be configured in the TOSCA template.	Y	N, not natively but it can be configured in the TOSCA template.	Y, Using the IaC paradigm the same deployment can be reproduced.

A subset of these systems was selected for further discussion and analysis in the workshops. The systems presented in the workshops depended, also, in the availability of the system owners.

4. In-depth analysis of selected workflow and orchestrator environments

The section will compare the selected orchestrator environments especially with regard those features that are more relevant to the project and analyse the workshop results.

An overview of the systems presented in the workshops is given below. Presentations about Galaxy and SnakeMake were initially planned, but did not take place due to final unavailability of speakers.

NextFlow[2]

NexFlow implements a DSL on top of a workflow management system and enables to orchestrate and parallelise the execution. It scales to different infrastructures and leverages containerization. Tasks can be encapsulated into containers, with support for multiple container technologies (Docker, Singularity, ...) and support for multiple job/cloud schedulers and infrastructures (Google cloud, AWS, Grid engine, Slurm, Azure, Kubernetes).

An open-source version is released under license Apache v2. A commercial version is distributed as NextFlow tower[3].

Distributed data and computing is managed with the task-based paradigm. In NextFlow, tasks are self-contained work units with no side-effects. Communication across tasks is achieved through asynchronous messages. Scheduling of tasks is delegated to the execution platform (each task is submitted as a job to the job scheduler). Data is stored in a shared file system or Object storage.

Sensitive data is delegated to target cloud/infrastructure solution: AWS S3/KMS for server-side encryption with custom keys. A secret manager has recently been added that allows safe handling of passwords and credentials in the user pipeline.

Provenance and reproducibility is based on the use of a unique ID per task computed as a 128 bit hash, which is used to create the task working directory. The data generated by the task is stored in this directory. Reproducibility is supported by containerisation together with built-in support for version control and management based on the commonly used git software suite for version control.

One Touch Pipeline (OTP)[4]

One Touch Pipeline (OTP) is a data management platform for running bioinformatics pipelines in a high-throughput setting, and for organising the resulting data and metadata. OTP automates the complete digital process from import of raw sequence data, via alignment and identification of genomic events, to notifying project members their analyses are finished. OTP is not a workflow manager itself and provides an interface with workflow environments (WES). Current supported

[2] https://nextflow.io
[3] https://tower.nf
[4] https://gitlab.com/one-touch-pipeline/otp

workflow managers are Roddy (native support), Snakemake and NextFlow (last two through the WES interface). OTP only supports the execution in clusters.

OTP is project based: a project is set-up with different metadata. A project may have users, with different roles and rights and can give different rights for files to different users.

Project information is stored in a database, which is uses to generate statistics. Support for in sensitive data is provided through third parties: The environment OTP runs in has to be secure. Reproducibility is supported by enabling workflow re-run through the data stored in the database. OTP is offered under the MIT license.

Dynamic On Demand Analysis Services (DODAS)[5]

DODAS (Dynamic On Demand Analysis Service) is an open-source Platform-as-a-Service tool, which allows to deploy software applications over heterogeneous and hybrid clouds. DODAS is currently part of the EGI-ACE Service portfolio.

DODAS aims to be an experiment agnostic cloud enabler for scientists seeking to easily exploit distributed and heterogeneous clouds to process, manipulate or to generate simulated data. One of the major drivers of DODAS is to reduce as much as possible the learning curve, as well as the operational costs of managing community specific services, while running on Cloud.

DODAS supports distributed compute resources, with different flavors: batch-like, interactive and quasi-interactive. High-level federation of resources is done via HT-Condor orchestrated via Kubernetes, with support for hybrid resources (HTC-HPC and Cloud), and cloud orchestration through INDIGO-PaaS, which coordinates the provisioning of virtualized compute resources on both private and public Clouds.

Data management supports object storage (S3) via MinIO, and POSIX-like file based storage is also supported, connected to high-level application layers. A caching solution for CDN is implemented with Xrootd and MinIO.

Modern federated identity management for AAI is supported through INDIGO-IAM (i.e allows to federate EGI Check-in). Software distribution is done via CVMFS and containers (i.e. Docker, Singularity).

DODAS per se does not implement anything specific for sensitive data. Relies on Enhanced Privacy and Compliance Cloud (EPCC).

[5] https://dodas-ts.github.io/dodas-apps/

Data provenance is supported via MinIO metadata and user defined metadata enrichment. Reproducibility is supported through workflow automation based on triggers and events.

DODAS relies on services which software is under Apache 2.0 License.

PyCOMPSs[6]

PyCOMPSs is a task-based programming model that enables the parallel execution of workflows in distributed computing infrastructures. PyCOMPSs enables to have the application code independent of the infrastructure: same code can run with different back-ends: cloud, HPC, etc. It supports multiple cloud and container environments.

The data distribution supports both data stored on local file systems and shared file systems. The runtime performs the necessary data transfers (both objects in memory and files) between nodes, giving the illusion of a single large file system or memory.

PyCOMPSs offers support for persistent storage: objects in persistent memory are accessed as regular objects. Through dataClay supports different user profiles which can be used to implement support for sensitive data

There is ongoing work in the integration with Trusted Execution Environments through Scone (TUD). All communications from containers are encrypted.

Provenance is supported thanks to a recent development based on RO-CRATE. A logger registers unique accesses to files and directories, to automatically identify inputs and outputs of the workflow. A post-process extracts the information to generate the RO-Crate (using ro-crate-py library).

Reproducibility is under development, through the HPC Workflows as a Service (HPCWaaS) paradigm. HPCWaaS will enable workflows to be defined by developers and later instantiated and executed by users. The technology is based on containerization of all the components.

PyCOMPSs is provided under Apache v2 license.

Pegasus[7]

The Pegasus project encompasses a set of technologies that help workflow-based applications execute in a number of different environments including desktops,

[6] compss.bsc.es
[7] http://pegasus.isi.edu

campus clusters, grids, and clouds. It supports the separation between workflow description and workflow execution. In addition, it offers tools for task execution - monitoring, debugging, fault tolerance…

Pegasus involves three different components: the Pegasus planner which maps workflows to infrastructure, the DAGMan workflow engine that manages dependencies and reliability and the HTCondor scheduler/broker which is used as a broker to interface with different schedulers.

Pegasus is deployed with a workflow submit node and one or more compute sites. With regard data, it differentiates between input sites, that host input data, and data staging site, which coordinates the data movement for the workflow.

Is also offers the Pegasus-transfer tool, with support for multiple protocols for data transfer (http, scp, gridftp, …) which also does credential management.

Pegasus supports provenance by tracking at run-time information about the workflow execution. It gets information from the workflow logs tracking information about the jobs and stores this information in a relational database. It also supports real-time monitoring with the Pegasus dashboard.

Pegasus does not provide any specific support for sensitive data and it is highly dependent on the actual environment. Also, it does not support for encryption/decryption. It has support for secure protocols to transfer data (can use scp, s3, ssh, gridftp).

Pegasus provides comes an end-to-end integrity checking that ensures data does not get corrupted in transit. This tool performs integrity checksums on input files before job starts with support for sha256 checksums. Is able to detect failures amnd a job failure is triggered if checksums fail.

Reproducibility is natively supported from the workflow description. Support for containers is also provided: users can use containers with their executable preinstalled (each job runs in a container). Support for multiple container environments is provided.

Pegasus is provided under Apache v2 license.

Micado[8]

Micado is a cloud orchestrator aiming at giving support to reusable microservices that communicate between each other.

[8] https://micado-scale.readthedocs.io/

Micado aims at giving support when deploying applications and to dynamically adjust the execution. Provides automated application deployment based on TOSCA-based descriptions (topology and policies) and automated scaling based on highly customisable scaling policies. It supports scaling at container and virtual machine levels. It also comes with advanced security settings.

Microservices are deployed into the cloud based on the topology and (TOSCA description). After deployment the status of the application is monitored. Based on the monitoring the environment can be changed, taking into account the specified scaling policies. If running out of resources, new resources can be allocated.

It relies on different tools and environments. Cloud orchestrators: Terraform or Occopus; Containers: Kebernetes (Swarm also, but not used now); Monitoring: Prometheus. Micado components include a Submitter, a Policy Keeper and a machine learning Optimiser.

The distribution aspects in Micado are based on the microservice architecture. For computation it provides support for multi-cloud and edge/fog applications., including portability between multiple clouds. Data distribution supports solutions native to Kubernetes and Terraform. However, applications are responsible for moving the data and computation is driven by the microservices invocation. The support for Edge works the same as with the cloud.

Micado implements some advance security solutions, with user authentication and authorization and secret management in master node (via Hasicorp Vault). Also, it uses kubernetes secret in the worker node.

Sensitive data is not handled by default, but they have given support to a specific use case in hospitals, with security solutions to manage sensitive data and deployed with Micado.

In addition to the microservices execution, Micado can be configured to support batch job execution. It is implmented with the JQueuer manager that receives a json description. Micado supports the execution of a number of jobs in a number of workers that can be scaled.

Reproducibility is not a strong point, with no direct support. Although a rich set of monitoring information is collected, it is not currently stored. There is a potential to extend with provenance support, e.g. via ProvToolbox.

Micado is provided under Apache v2 license.

StreamFlow

StreamFlow provides runtime support for hybrid acyclic workflows on multicontainer environments. It can run in multiple heterogeneous resources. StreamFlow relies on open standard Common Workflow Language (CWL), based on

YAML, for the workflow description. The description of the execution environment is provided in a model file. StreamFlow provides support for container and multi-container environments. With regard data, StreamFlow does not require a shared data space among involved workers.

StreamFlow supports the executing of workflows in multiple execution environments, such as docker containers, ssh accessible nodes or cluster job managers. A workflow in StreamFlow is a set of steps. A step becomes fireable when all its predecessors have completed. The required software environment needs to be deployed (i.e., container or service) as well as data should have been transferred by the runtime before the step is executed. Data is transferred with ssh.

Sensitive data is implicitly supported in StremFlow by moving the computation to the node where the sensitive data is. When data transfers are unavoidable, StreamFlow ensures that data is passed through encrypted channels (SSH or HTTPS WebSockets). In the future, they plan to support trusted execution environments and data encryption at rest.

StreamFlow provides portability through container support (Docker and Singularity). The explicit description of the execution environment (the locations topology) fosters reproducibility and reusability. Provenance is provided through storing log information about scheduling, execution location, commands, and intermediate results. In the future, they plan to adopt a more standardised provenance format (e.g. an extended version of CWLProv that handles distributed workflows).

StreamFlow is provided under LGPLv3 license.

WfExS

WfExS is a high-level workflow execution service backend, through the setup on reproducible, secure execution environments, focusing in HPC environments. WfExS is more an orchestrator than a workflow engine, focusing on enabling the access to human sensitive data from analytical workflows. It has a strong focus on reproducible and replicable analysis, and RO-Crate is used to describe the different digital objects which were involved in a workflow execution.

It considers secure execution scenarios where due to legal requirements data needs to stay where it is and computation needs to be moved there. Alternatively, computational resources stay and data needs to be securely moved (encryption).

It is based on the use of permanent or public identifiers: workflow available in public repositories (i.e., WorkflowHub, Dockstore); containers are used by workflows; inputs and reference datasets.

WfExS keeps cached copies of everything unless it is sensitive data. It keeps a clear separation among the workflow execution scenario preparation (where all the preconditions are materialized), workflow execution (no external resource should be queried or fetched) and the export of results. It maintains a separate working directory for each execution with a copy of the inputs, workflow and containers is kept, as well as intermediate, outputs and metadata from the execution. In case of sensitive data, the whole working directory is encrypted with a FUSE encryption filesystem (gocryptfs, encfs).

To execute a workflow, WfExS fetches the description of the workflow (NextFlow and CWL are supported). Encrypted user keys are used to access the computing resources and delegates the execution to other workflow engines. There is challenge on keeping the working directory transparently encrypted in a distributed environment.

An important component in WfExS is the fetcher that obtains the contents required for an execution: workflows' description, containers, reference datasets, inputs... WfExS provides multiple fetchers' implementations based on http, ftp, S3 between others and is planning to support others.

Reproducibility is planned by re-executing workflows from previously generated RO-CRATEs.

WfExS is provided under Apache v2 license.

4.1. Workshop final discussion

In addition to the specific questions to each workflow or orchestration system presentation, the following situation was presented to all speakers to address it from the point of view of their frameworks:

Imagine there are multiple hospitals, each with their own clusters or computational nodes, geographically distributed. Also, hospitals have their own databases with data with privacy/sensitvity issues. Sometimes this data would not be possible to be moved for sensitivity issues, but there is a need to run workflows that involve multiple of these venues. How do you think your solution can help in this scenario?

We highlight some the more relevant answers below:

PyCOMPSs: part of the computation can be done in each hospital with the data they own. Then, a PyCOMPSs application can be orchestrating the different parts and exchange processed data between the nodes (intermediate results, which do not have sensitivity issues). Since PyCOMPSs can also support services orchestration, a micro-service can be deployed in each of the nodes and the COMPSs runtime will orchestrate the whole execution. The support for trusted environments is delegated to Scone.

DODAS: Using only a distributed environment can be not efficient all time. If we consider that sensitive data can be anonymized, some steps can be distributed and others run centralized in the cloud.

NextFlow: the strategy will be to move the computation where the data is. Some hybrid cases can be also considered. Multiple users can deploy in multiple nodes.

Micado: in this case, the application should provide the data protection aspects. A Micado instance can be deployed in each execution node and Micado workers would run in a secure environment, although data is not encrypted in Micado.

StreamFlow: StreamFlow will be aware that data cannot be moved from worker to worker. The master will do the exchange of the intermediate results that can be transferred.

Pegasus: this scenario can be quite straightforward supported in Pegasus. During the site selection phase, you can select sites based on the data in that node.

4.2. Workshops' conclusion and guidelines for the future

Analysing the status of current workflow and orchestration environments presented in the workshops we find out that the community is considering the aspects required for an HRIC infrastructure.

Basic support for distributed data and computing is provided by all studied systems. In fact, workflow environments have had a long tradition of execution in distributed environments originated from grid computing initiatives. However, this distribution of data and computing is considered in different ways by the different systems. For example, while PyCOMPSs can run a whole workflow as a single job in an HPC cluster, NextFlow or StreamFlow submit steps of the workflow as individual jobs. Also, support for cloud environments is available for most of the systems, as well as support for containerized executions.

Reproducibility and provenance are also well tackled by most of the solutions, by providing means to store details about executed workflows that can be used to reproduce the workflows or to analyze the data analysis process.

What is less common is the support for sensitive data. Most of them rely on third party solutions or are only able to provide solutions based on encryption, without support for more complex solutions that can consider which pieces of data are sensitive or not, and different user profiles. While WfExS focuses on the access of human sensitive data, we consider the field is an open topic to be considered by the community.

Therefore, HRIC applications can be based on existing mature workflow environment solutions by reinforcing aspects related to the support of sensitive data. The community have been working on these environments for a long time and it will be important to not reinvent the wheel when considering the frameworks to

support HRIC applications. In this sense, it will be good to involve the stakeholders of the workflow environments in the design of the HRIC application workflows.

For the forseeable future, there is no simple out of the box solution to provide workflow execution systems that can handle sensitive data easily. HRICs will need to implement appropriate technical infrastructure and technical and organizational measures to ensure the proper handling of sensitive data irrespective of the HRIC application they choose.

5. Conclusion

This deliverable presents the results of task 5.2 of the HealthyCloud project. The task has surveyed a large number of workflow environments and orchestrators, with the goal of analysing the characteristics that are more relevant to the project.

Some of the available environments fulfil most of the needs of HRIC applications and provide good starting points for their development.

Annex I – Orchestration environments questionnaire

HEALTHYCLOUD
Health Research & Innovation Cloud

Questions for D5.2 Analysis of existing orchestration mechanisms for distributed computational analyses

Workflow environment/orchestrator name:

Assessment made by:

Contact:

Date of assessment: dd/mm/yyyy

Answer briefly the following questions:

1. **URL of environment**
2. **Open Source (Y/N), License**
3. **Maturity level (TRL or similar)**
4. **Being used with health data (Y/N), which data has been used**

Answer with a bit of detail the following questions:

5. **Support for distributed data (Y/N), indicate short summary of how this is achieved**

Notice: *The HealthyCloud project has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement Nº965345*

6. **Support for execution in multi-domain computing infrastructure (Y/N)**

7. **Support for sensible data (Y/N), how this is dealt**

8. **Support for reproducibility/provenance (Y/N, mechanism, i.e., containers, etc)**

9. **Main initiatives where it has been used (give number and a short list in case of many)**

Annex II – Agenda first workshop

1st HealthyCloud workshop
Analysis of existing orchestration mechanisms for distributed computational analyses

Location:

https://rediris.zoom.us/j/3606864208?pwd=UFZWL0xueEw5MWhnM0Z6aFlBeW1ZZz09

Meeting ID: 360 686 4208

Passcode: 318244

Date: 27/05/2022

Time: 10:00 to 12:30 CEST

Agenda

10:00 - 10:15	Welcome and short introduction	Rosa M Badia (BSC)
10:15 - 10:30	Galaxy	Bjoern Groening
10:30 - 10:45	SnakeMake	Johannes Köster
10:45 - 11:00	NextFlow	Paolo Di Tommaso
11:00 - 11:15	OTP	Ivo Buchhalter
11:15 - 11:30	DODAS	Daniele Spiga
11:30 - 11:45	PyCOMPSs	Daniele Lezzi
11:45 - 12:15	Discussion	Rosa M Badia (moderator)
12:15 - 12:30	Workshop conclusion	Rosa M Badia (moderator)

Notice: *The HealthyCloud project has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement Nº965345*

Annex III – Agenda second workshop

2nd HealthyCloud workshop
Analysis of existing orchestration mechanisms for distributed computational analyses

Location:

https://rediris.zoom.us/j/3606864208?pwd=UFZWL0xueEw5MWhnM0Z6aFlBeW1ZZz09

Meeting ID: 360 686 4208

Passcode: 318244

Date: 06/07/2022

Time: 16:00 to 18:30 CEST

Agenda

16:00 - 16:15	Welcome and short introduction	Rosa M Badia (BSC)
16:15 - 16:35	Pegasus	Ewa Deelman
16:35 - 16:55	Micado	Tamas Kiss
16:55 - 17:15	StreamFlow	Iacoppo Colonnelli
17:15 - 17:35	Wfex	Jose M Fernandez
17:35 - 18:15	Discussion	Rosa M Badia (moderator)
18:15 - 18:30	Workshop conclusion	Rosa M Badia (moderator)

Notice: *The HealthyCloud project has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement Nº965345*

Acronyms and Abbreviations

Each term should be bulleted with a definition.

Below is an initial list that **should be adapted** to the given deliverable.

- CA – Consortium Agreement
- D – deliverable
- DoA – Description of Action (Annex 1 of the Grant Agreement)
- EB – Executive Board
- EC – European Commission
- GA – General Assembly / Grant Agreement
- HPC – High Performance Computing
- IPR – Intellectual Property Right
- KPI – Key Performance Indicator
- M – Month
- MS – Milestones
- PM – Person month / Project manager
- WP – Work Package
- WPL – Work Package Leader