D5.1 First Architecture design and Implementation Plan

Status: FINAL

Dissemination Level: public

Funded by the European Union

Abstract

Key Words	Computing, Storage, HTC, Cloud, Data, Orchestration, Policies

This deliverable provides a comprehensive description of the Digital Twin Engine Infrastructure architecture. It focuses on the overall design of the system that provides the pillars for both data and compute federations. This includes the requirement analysis of the needs of the scientific communities as well as the requirements coming from a set of highly heterogeneous resources providers. It will detail the different components of the architecture defining the related role and how they will interact with each other in order to implement the digital "continuum" between cloud-edge and multi cloud environments, HPC Centers and Quantum. Policies to access the resources and resources accounting will be also discussed.

Finally, it provides a timeline and roadmap toward the implementation of the Digital Twin Engine architecture including the vision for the early testbeds and pilot systems.

Document Description			
D5.1 First Architecture design and Implementation Plan			
Work Package number WP5			
Document type	Deliverable		
Document status	FINAL	**Version**	1
Dissemination Level	Public		
Copyright Status	This material by Parties of the interTwin Consortium is licensed under a Creative Commons Attribution 4.0 International License.		
Lead Partner	INFN		
Document link	https://documents.egi.eu/document/3945		
DOI	https://doi.org/10.5281/zenodo.8036983		
Author(s)	• Diego Ciangottini (INFN) • Paul Millar (DESY) • Liam Atherton (UKRI) • Marica Antonacci (INFN) • Daniele Spiga (INFN) • Andrea Manzi (EGI) • Renato Santana (EGI) • David Kelsey (UKRI) • Adrian Coventry (UKRI) • Shiraz Memon (JSC)		
Reviewers	• Pablo Orviz (CSIC) • Tom Clark (EODC)		
Moderated by:	Sjomara Specht (EGI)		

Approved by	Yurii Pidopryhora (MPG) on behalf of TCB

Revision History

Version	Date	Description	Contributors
V0.1	06/03/2023	ToC	Daniel Spiga (INFN)
V0.2	26/04/2023	First version Policy and accounting	Liam Atherto (UKRI), Adian Coventry (UKRI), Renato Santana (EGI)
V0.3	27/04/2023	First version of the Data Management section	Paul Millar (DESY)
V0.4	02/05/2023	First version architecture, requirements, and providers	Daniele Spiga (INFN)
V0.5	08/05/2023	First version of the Federated Compute section	Diego Ciangottini (INFN)
V0.6	10/05/2023	First version of the Orchestration section	Marica Antonacci (INFN)
V0.7	14/05/2023	Version ready for Internal review	Daniele Spiga (INFN) all the other authors
V0.8	02/06/2023	Version including reviewers comments	Pablo Orviz (CSIC) Tom Clark (EODC)
V0.9	03/06/2023	Version ready for TCB approval	Daniele Spiga (INFN)
V0.9a	09/06/2023	Version approved by TCB	Yurii Pidopryhora (MPG)
V1.0	12/06/2023	**Final**	

Terminology / Acronyms

Term/Acronym	Definition
DTE	The Digital Twin Engine developed by interTwin
FTS	The File Transfer Service. A software component maintained by a development team at CERN.

Rucio	A third generation data management software component, maintained by the Rucio development team.
gfal	Grid File Access Library; software that provides an abstraction for POSIX-like access.
CVMFS	The CernVM File System provides a scalable, reliable, and low-maintenance software distribution service.
AUP	Acceptable Use Policy
WISE	Wise Information Security for Collaborating e-Infrastructures
EOSC	European Open Science Cloud
AARC	Authentication and Authorization for Research Collaborations
PDK	Policy Development Kit
K8s	Kubernetes. Container Orchestration Technology
QoS	Quality of Service
NVMe	non-volatile memory express is a new storage access and transport protocol for flash and next-generation solid-state drives (SSDs)
scp	Secure copy protocol (SCP) is a means of securely transferring computer files between a local host and a remote host or between two remote hosts.
UFTP	UNICORE FTP is a file transfer tool similar to Unix'

Terminology / Acronyms: **https://confluence.egi.eu/display/EGIG**

Table of Contents

Table of Figures

Table of Tables

Executive summary

The twofold objectives of the deliverable D5.1 are to provide a comprehensive collection and analysis of the requirements collected from relevant stakeholders and to describe the first version of the architecture for the Digital Twins Engine infrastructure.

Requirements include both those from scientific communities that need to interact with WP5 services through the WP6 provided systems, as well as the requirements of resource providers that are an integral part of WP5 itself.

The document has been composed by experts working on the various pillars of the architecture with a deep knowledge of the adopted services. From the technical perspectives the document provides a general overview of the overall architecture and describes the vision toward the implementation of a continuum model. The latter includes all the challenges related to the compute, data (and storage) and policy specific matters.

1 Introduction

1.1 Scope

The primary scope of this deliverable is to provide an overview of the Digital Twin Engine architectural pillars and how they are harmonically integrated. On the one side, addressing the requirements coming from the portfolio of scientific communities that interact with WP5 through the WP6 services (including the related interfaces and APIs) and on the other side the technical needs of the interTwin heterogeneous sites. Those represent the actual computing capacity of the project.

The services and the systems that compose the architecture of the DTE infrastructure have the role to implement all the capabilities needed by the Digital Twins to effectively exploit computing resources. In summary WP5 is expected to:

- Provide software solutions to enable resources provisioning on a wide range of compute providers to implement a digital continuum.
- Support data access, data management and to support real time data processing, in a federated environment.
- Develop and document best-practice to achieve harmonised access in terms of authentication and authorisation, security, and operational policies together with Rules of Participation.
- Provide services and tools to enable the automated storage and compute resources orchestration enhancing the automatic and intelligent identification of the best provider either based on static and dynamic metrics.

The design of the DTE architecture will follow the specific requirements provided by the WP6. Additionally, requirements will come from the sites supporting the activities of interTwin.

1.2 Document Structure

This document describes the technical design of the architecture for the Digital Twin Engine infrastructure, and it is organised in three main sections corresponding to the description of the requirements, the overview of the architecture and the detailed description of the components that compose the architecture. A short focus on the providers and on policy access are also provided. The document is organised as follow:

- Section 2 describes the providers that take part in the project. The main objective is to show what type of resource is available and will be integrated through the services of WP5. The procedure to gather the feedback from the providers introduced is also described
- Section 2.3 summarises the requirements that come from the site and from WP6 as the principal interface to WP5 exposed services..

- **Section 3** describes the overall architecture and highlights the key features. The result of the scouting performed to identify the suitable solution used to build the infrastructure is presented as well.
- **Section 4** is about the fine grained description of all the software components developed and enhanced by WP5. A general description and detailed architectural overview of the components are also provided.
- **Section 5** is about policies to access the resources.

Finally, the last section draws the conclusions about the technical design of the architecture and summarises the analyses performed during the first eight months of the project. These findings were needed to implement the first design of the DTE Infrastructure.

2 Requirements

Before designing the architecture for the WP5 DTE infrastructure, a wide collection of requirements has been performed. Requirements have been gathered from all the stakeholders involved with the WP5 services integration. This was a mandatory step toward the success of the design process.

WP5 identified the two main actors to take into consideration since this early design phase, namely WP6 and Resources Providers. First, WP6 is the main responsible entity for developing and maintaining most of the components that directly interface WP5 such as Workload Management systems, Pipeline services, interfaces for interactive data analysis etc. Second, the participating sites represent the computing capacity that must be exploited by scientific communities through the WP5 services.

Although scientific communities are the drivers of the requirements definition process, at the current stage of the design, we do not foresee that scientific communities will be directly interfacing WP5 but channelling the requirements through WP6 except for the interactions with data management services. The latter will require a direct interaction with end-users. In any case, like the providers, as soon as the testbeds will be ready, the scientific communities are expected to co-design the overall system (the next phase of the project).

In order to gather requirements, we started with a collection of the site specification information. The decision was to prepare a survey as the most executive solution for us to build a comprehensive view of both the technical and policy related peculiarities.

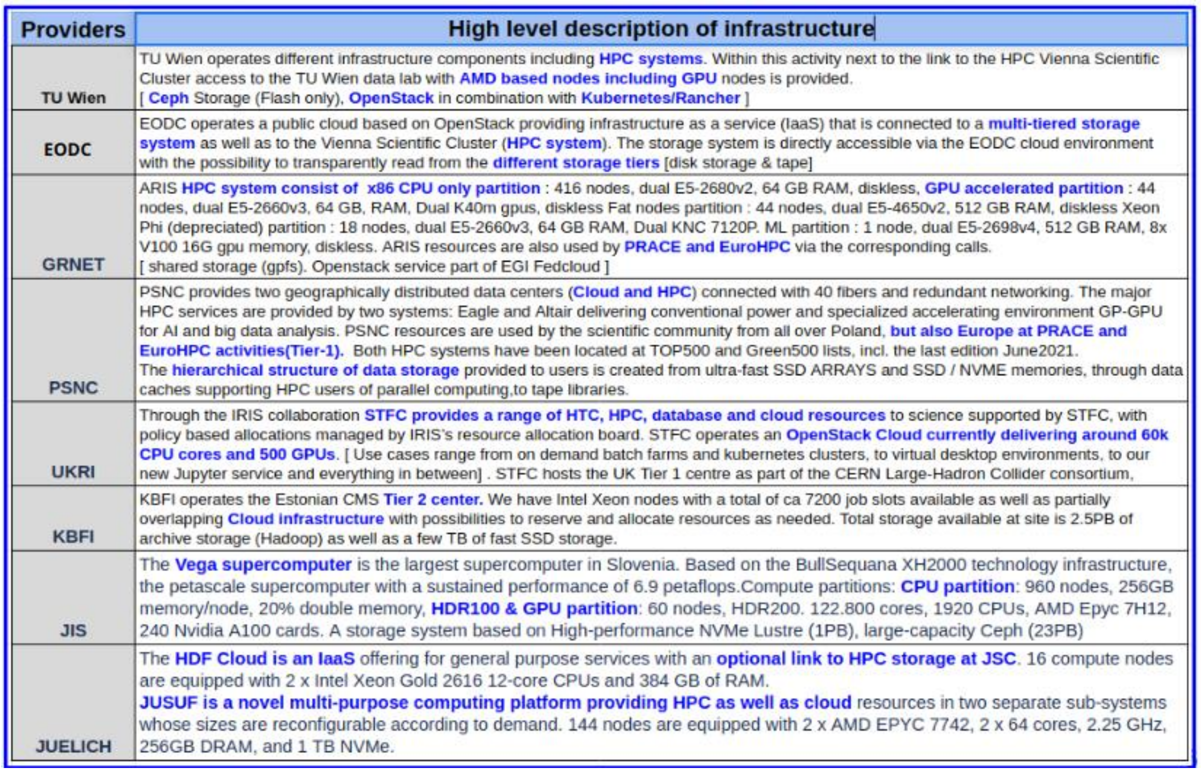
In this section the schema of the survey is detailed. Then the results of the analysis are presented. Finally, a summary of the requirements taken from WP6 is shown.

2.1 Resource Providers at interTwin

An interTwin resource provider is any organisation that contributes with computing resources supporting the scientific use-cases of the project. Providers are meant to support the science needs such as to train the digital twin models (AI/ML), to perform analysis on those models, and conduct any subsequent post-processing as well as data analysis. It is worth highlighting that to cope with these providers need also to grant, to some extent, data access, transfer, archiving capability that a Digital Twin might require.

As shown in table 1 there are 8 sites participating in the project, overall featuring a high level of heterogeneity in terms of size, technology, and type. They span from specialised HPC Centers to Cloud providers passing through High Throughput dedicated sites.

Table 1 - High-level specification of the interTwin site providers



Providers	High level description of infrastructure
TU Wien	TU Wien operates different infrastructure components including **HPC systems**. Within this activity next to the link to the HPC Vienna Scientific Cluster access to the TU Wien data lab with **AMD based nodes including GPU** nodes is provided. [**Ceph** Storage (Flash only), **OpenStack** in combination with **Kubernetes/Rancher**]
EODC	EODC operates a public cloud based on OpenStack providing infrastructure as a service (IaaS) that is connected to a **multi-tiered storage system** as well as to the Vienna Scientific Cluster (**HPC system**). The storage system is directly accessible via the EODC cloud environment with the possibility to transparently read from the **different storage tiers** [disk storage & tape]
GRNET	ARIS **HPC system consist of x86 CPU only partition** : 416 nodes, dual E5-2680v2, 64 GB RAM, diskless, **GPU accelerated partition** : 44 nodes, dual E5-2660v3, 64 GB, RAM, Dual K40m gpus, diskless Fat nodes partition : 44 nodes, dual E5-4650v2, 512 GB RAM, diskless Xeon Phi (depreciated) partition : 18 nodes, dual E5-2660v3, 64 GB RAM, Dual KNC 7120P. ML partition : 1 node, dual E5-2698v4, 512 GB RAM, 8x V100 16G gpu memory, diskless. ARIS resources are also used by **PRACE and EuroHPC** via the corresponding calls. [shared storage (gpfs). Openstack service part of EGI Fedcloud]
PSNC	PSNC provides two geographically distributed data centers (**Cloud and HPC**) connected with 40 fibers and redundant networking. The major HPC services are provided by two systems: Eagle and Altair delivering conventional power and specialized accelerating environment GP-GPU for AI and big data analysis. PSNC resources are used by the scientific community from all over Poland, **but also Europe at PRACE and EuroHPC activities(Tier-1).** Both HPC systems have been located at TOP500 and Green500 lists, incl. the last edition June2021. The **hierarchical structure of data storage** provided to users is created from ultra-fast SSD ARRAYS and SSD / NVME memories, through data caches supporting HPC users of parallel computing,to tape libraries.
UKRI	Through the IRIS collaboration **STFC provides a range of HTC, HPC, database and cloud resources** to science supported by STFC, with policy based allocations managed by IRIS's resource allocation board. STFC operates an **OpenStack Cloud currently delivering around 60k CPU cores and 500 GPUs**. [Use cases range from on demand batch farms and kubernetes clusters, to virtual desktop environments, to our new Jupyter service and everything in between] . STFC hosts the UK Tier 1 centre as part of the CERN Large-Hadron Collider consortium,
KBFI	KBFI operates the Estonian CMS **Tier 2 center.** We have Intel Xeon nodes with a total of ca 7200 job slots available as well as partially overlapping **Cloud infrastructure** with possibilities to reserve and allocate resources as needed. Total storage available at site is 2.5PB of archive storage (Hadoop) as well as a few TB of fast SSD storage.
JIS	The **Vega supercomputer** is the largest supercomputer in Slovenia. Based on the BullSequana XH2000 technology infrastructure, the petascale supercomputer with a sustained performance of 6.9 petaflops.Compute partitions: **CPU partition**: 960 nodes, 256GB memory/node, 20% double memory, **HDR100 & GPU partition**: 60 nodes, HDR200. 122.800 cores, 1920 CPUs, AMD Epyc 7H12, 240 Nvidia A100 cards. A storage system based on High-performance NVMe Lustre (1PB), large-capacity Ceph (23PB)
JUELICH	The **HDF Cloud is an IaaS** offering for general purpose services with an **optional link to HPC storage at JSC**. 16 compute nodes are equipped with 2 x Intel Xeon Gold 2616 12-core CPUs and 384 GB of RAM. **JUSUF is a novel multi-purpose computing platform providing HPC as well as cloud** resources in two separate sub-systems whose sizes are reconfigurable according to demand. 144 nodes are equipped with 2 x AMD EPYC 7742, 2 x 64 cores, 2.25 GHz, 256GB DRAM, and 1 TB NVMe.

The provider selection represents a perfect playground made of a handful of distinct backends (**Figure 1**) whose integration challenge is key for the project in order to develop and experiment software solutions to transparently provision computing capacity on a wide range of providers. In turn this will allow interTwin to enhance DTE modules with capability to access large amounts or resources accessing local and remote data.

Indeed, the ultimate objective is to deliver a generic and extensible architecture for the DTE infrastructure, providing stable interfaces to the storage systems and the distributed data sources and not an integration tailored to a single site configuration. Thus, having a set of heterogeneous providers, each one potentially offering different interfaces represents a unique opportunity.

Resource providers on their side must co-design the platform to ensure the development of effective solutions and to avoid naive design. Summarising sites are a key asset for the WP5 programme of work

2.2 Survey for providers

As anticipated in order to coherently collect technical information from resources providers, a survey has been prepared. A list of 19 questions, split into three main categories has been submitted to all the sites with the objective were to gather details about the current site configurations in the relevant areas in order to better understand which kind of solution the WP5 services needs to provide in order to make the integration effective. The areas of interests are listed below:

2.2.1 Compute resources and access interfaces:

The aim of this area of the survey was to identify the exact set of compute resources types, meaning high level cloud services, virtual machines, and containers, together with the types of supported interfaces, i.e., self-service API or Graphical interfaces. Moreover, we wanted to understand if sites offer managed batch systems at all, or simply virtual machines are provided, or more modern container orchestrators were deployed.

In addition to that, we wanted to understand the level of flexibility and commitment from our providers to allocate their resources for DTEs for scaling purposes. The availability of specialised hardware such as GPUs (if any) and the quantity accessible by scientific communities and under which “policy” the site can make them available to the interTwin project. Another aspect that we wanted to understand from the survey was the possibility for a provider to support global file systems such as CVMFS, needed for user software distribution and, possibly, restrictions for project-specific repositories. Finally, we wanted to verify whether the usage of containers to execute end-user payloads is supported or not.

2.2.2 Data and Storage

In this case the main objective was to understand in detail what storage capacity (or quota) each site can offer, although it is rather clear that the overall system is expected to change dynamically over the time it is important to know which kind of topology can be implemented for the first data lake system of interTwin. Similarly, we planned to identify the storage quality of service (Storage QoS) that each site can offer, if more than one. This information helps to understand whether user communities can exploit different storage types at a given site; for example, high-performance vs scratch vs archival. This is rather important once a scientific community designs a computing model for a Digital Twin.

In particular, about the QoS we expect several use cases that might have the need of High-speed NVMe storage, or High-speed distributed disk accessible on the various nodes/ virtual machines. Information about storage and its quality is expected to be relevant for resource orchestration as well.

Another objective of the survey has been to try to identify the internal organisation of the storage system and how this might impact on the data ingestion, i.e., if data can be uploaded to this system and is immediately available across all of a given user's virtual environment. Also, this has been linked to the data transfer support. We wanted to survey the already available solutions for the user data transfer, a key also to design approaches and capabilities for the user produced data handling (user output) that was already provided, Last but not least the compute nodes access to storage has been a topic of the survey. In this respect we aimed to clarify if some shared/mounted file system is available or if the user processes are expected to manage data-access themselves

2.2.3 Policies

Policy wise we tried to cover several aspects starting with the options that exist for installing new software. interTwin represents a major step forward, by building a distributed analysis environment. It may prove helpful, if we find limitations with existing services, to explore new approaches and software for exposing the existing resources. Examples are: storage services, caches, compute edge services etc.

Since data has a key role for the communities of interTwin we wanted to understand what obligations exist for using storage resources and if it is expected that the researcher removes data within that period, or risks having their files deleted.

Moreover, we wanted to identify how providers grant access to compute and data resources in terms of access policies (and procedures) as well as to try to highlight users' acceptance of terms and conditions for resource usage.

2.3 Requirements from providers

As introduced the sites present a very high level of heterogeneity. There are both cloud, HPC and HTC providers (**Figure 1**). Some are enabling a mix of them by means that both Cloud solution and HPC/HTC are supported by a single provider.

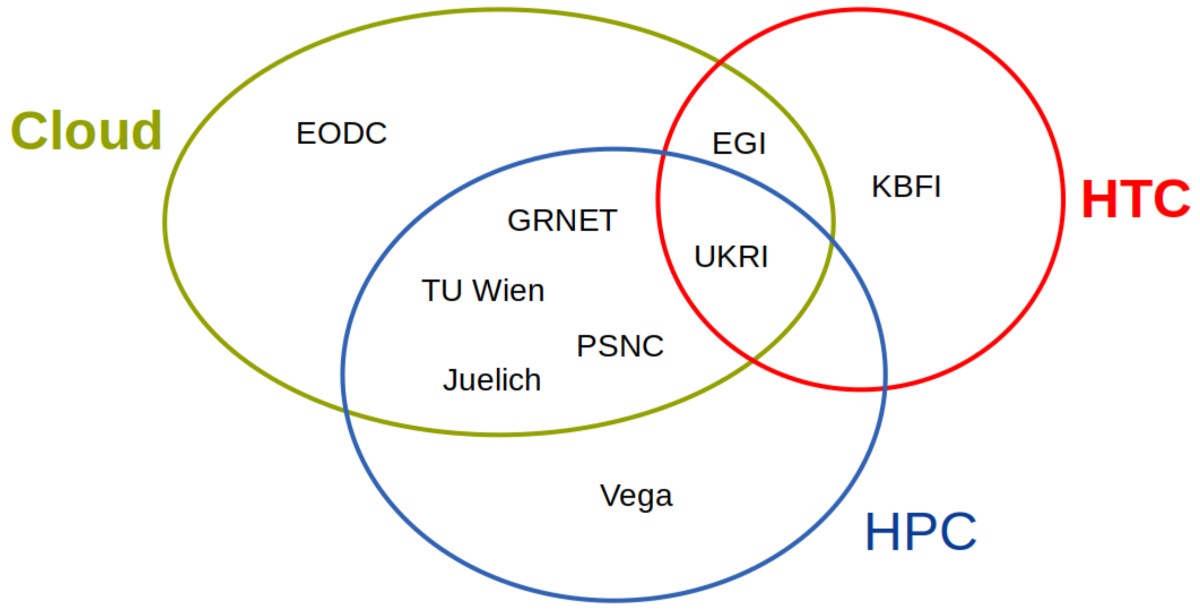


Figure 1 - Schematic representation of the heterogeneous resources providers in interTwin

One example is Juelich that offers a Cloud service that supports some internal integrations with HPC. The latter is not accessible from remote. One of the HPC centres is also supporting WLCG as an opportunistic provider for the experiment. Also, in terms of hardware the offer differs from site to site. Both CPUs and GPUs are available although not all centres have both of them. All this gives already an example of how the WP5 will need to enable the possibility for a service (i.e., a scientific community) to define which kind of hardware is required and then to re-route the workflow accordingly to the request but without any direct action taken by the end-user. Many more features are expected to be considered to enable such routing or compute resource orchestration. From the information collected thanks to the survey described in **Section 2.2** we extracted the first set of high-level requirements that WP5 services should satisfy in order to grant an effective integration.

From the compute specific part, a list of major topics is summarised below:

1. All the sites offering Cloud services are adopting Openstack[1] as middleware.
2. The actual resources are exposed differently from site to site. The main three options are
   - Centrally managed batch system implemented either via Slurm[2] or HTCondor[3]
   - Site managed Virtual Machine
   - Self-service Virtual Machine via publicly exposed interfaces/APIs
3. A subset of the sites configured to offer a centrally managed batch system provide support to Computing Element[4] as a gateway for remote access. All the others provide access via login nodes with locally managed accounts where users actually perform access using ssh.
   - Technology wise, sites supporting a computing element actually adopted ArcCE[5] as implementation.
   - Yet another approach is supported in case of HPC which actually is to enable HPC access via local cloud service. In such a case the HPC is only exposed internally.
4. Not all the sites foresee the possibility to deploy specific services at the edge. Some are already doing it but upon negotiation, this is to highlight that it is not possible to expect a standardised pattern
   - Technical ad hoc evaluation is required if this kind of service is expected to be installed.

1 https://www.openstack.org/

2 https://slurm.schedmd.com/documentation.html

3 https://htcondor.org/

4 https://cds.cern.ch/record/840543/files/lhcc-2005-024.pdf

5 https://www.nordugrid.org/arc/ce/

5. Network wise the landscape is highly heterogeneous. The most important aspect is that not all the providers set up their internal worker nodes with outbound connectivity. A common scenario at HPC centres.
    - This means that once a process lands and runs on such a resource, by default, it cannot reach an external service such as a community specific database or any other service (like a Workload management hosted on a cloud provider).
6. Concerning the software distribution WP5 needs to ensure that any resource integrated in the federation will access the same version of the software expected by the runtime of the user. However not all the providers seem to adopt an approach that is directly suitable in this respect. the two principal options are:
    - Locally managed installation i.e., on shared file systems
    - Through read only global file system solutions such as CVMFS.

    All cloud providers allow for CVMFS installation on self-provisioned Virtual Machine. In such a case the related management is fully delegated to the end user.
7. Another aspect to consider is the support to run virtualization, particularly lightweight containerisation. The HPC sites are mainly oriented toward Apptainer[6] as a technological solution to execute containers. Another option mentioned is udocker[7]. Containers will play a crucial role as a solution to distribute software libraries as well as to manage user level runtime environments. Once again no specific constraints come from the cloud provider.

Similarly, to the compute resource exploitation also in terms of data and storage related systems a high level of heterogeneity has been verified. In particular:

1. Storage interfaces at edge to enable remote data to read/write. Many storage systems (that provide network FS) are accessible from outside. However, in this case there is a huge fork between a rather complete setup where dedicated transfer nodes (i.e., via ARC data delivery[8]) is available, together with dCache[9] nodes and Xrootd[10] service for remote access to user posix filesystem up to providers that do not allow at all for remote storage access

[6] https://apptainer.org/

[7] https://indigo-dc.github.io/udocker/

[8] https://www.nordugrid.org/arc/arc6/tech/data/dds.html

[9] https://www.dcache.org/

[10] https://xrootd.slac.stanford.edu

   a. Additional solutions supported for data ingestion and data movement are: SCP, rsync[11], UFTP, Globus[12], Rclone[13]

2. Quality of services. Some sites provide High-speed storage (NVMe) provided as block storage towards virtual infrastructure. High-speed distributed disk is a distributed shared storage accessible on all virtual machines. Data can be uploaded to this system and is immediately available across all of a given users virtual environment. Organisation of data within a user's designated filespace is considered their own responsibility.
3. Internal storage organisation at each site. Many different storage solutions deployed. Seemingly all are available on the worker node. Compute nodes do not always have local storage. In those cases, storage is exposed via shared file systems. In this scenario, in order to process data, users are supposed to copy it on a shared filesystem which will be available to all compute nodes.
4. Storage capacity: This varies from TB to PB depending on specific setup

These features are considered to design the solutions to federated storage and to support data movement

2.4 Requirements from WP6

Various deliverables are dealing with requirements analysis at different levels. The D3.1 includes requirement analysis for the definition of the blueprint architecture, while D6.1 and the 2 deliverables for WP7, D7.1 and D7.2 consider User communities requirements for their respective WP. In this section we will include some requirements towards WP5 that have been extracted from D6.1 deliverables about the components to be developed there.

2.4.1 Advanced Workflow Composition (T6.1)

The Advanced Workflow composition task includes both the tool for the definition of the Workflows together with the data acquisition and event-triggered workflows as described in section 3 of the D6.1 deliverable.

The data acquisition component can exploit the availability of storages with SSE interface (dCache is explicitly mentioned as technology to at least have a first PoC, as it already provides this interface). In addition, provisioning on demand of Kubernetes clusters will be needed in order to deploy Knative and the OSCAR framework.

2.4.2 Quality Assurance (T6.2)

No particular requirements arise from the quality assurance task at this stage of the project.

[11] https://wiki.archlinux.org/title/rsync

[12] https://www.globus.org/data-transfer

[13] https://rclone.org/

2.4.3 Data Fusion (T6.3)

Data Fusion at this stage of the design phase is planning to rely mainly on openEO processes to run data fusion tasks, therefore it needs the availability of openEO backends on the infrastructure.

2.4.4 Big Data Analytics (T6.4)

As described in section 6 of the deliverable D6.1, the goal of the Big Data Analytics deployment layer is to create a set of topology templates and recipes for general-purpose data analytic environments to be deployed on demand on top of the cloud resources.

The cloud topology templates will be created using the OASIS TOSCA Simple YAML specification and will describe all the virtual resources and the software components required to deploy the final application. Furthermore, they will provide the user with a set of input parameters enabling them to customise the application configuration.

The main requirement is the availability of the PaaS Orchestrator that will be in charge of processing the TOSCA templates and creating all the required cloud resources.

2.4.5 Advanced AI workflows (T6.5)

The artificial intelligence (AI) subsystem in the proposed digital twin engine (DTE) is intended for data-driven digital twin (DT) models, and is mainly devoted to two macro-operations: training and deployment of machine learning (ML) model

Requirements at IaaS level:

- Nodes with multiple GPUs for distributed ML
- Storage space for Models Registry (~10s GBs, to begin with)

Requirements at PaaS level:

- Kubernetes-like API abstraction layer to manage container deployments.
- S3 object storage services, or the services to enable to deploy an S3 storage service (e.g., MinIO) from a container image
- RDBMS, or the services to enable to deploy an RDBMS (e.g., PostgreSQL) from a container image
- Interested towards Apache Airflow / Argo Workflows (Kubeflow)
- Kafka for real-time predictions and twinning (DT core property)

3 WP5 Architecture

Starting from the analysis of the requirements gathered and described above we defined the first version of the DTE infrastructure architecture. Three high level principles have been identified:

- A clear definition of the interfaces, both for the interactions with resources and toward the WP6 services.
- High flexibility either to accommodate diverse needs coming from very diverse scientific communities.
- Extendibility toward the integration of a very different set of resources.

The fact that computing capacity is provided by a set of sites with huge differences in their internal implementation requires WP5 to develop and provide a toolset including solutions to cope with all the highlighted peculiarities. In other words, WP5 does not foresee a single solution that fits all but the other way around: to develop a composable set of tools. This is considered a strategic approach as a path forward.

Regarding the computing federation and integration representing one of the objectives of the work package, two main concepts have been identified:

- Resources orchestration
- Workload and/or payload offloading

The two are supposed to be combined together and complement each other features wise. The resource orchestration has the main responsibility to keep track of the federation and thus to identify at a given point in time which is the "best provider" to support the deployment of a service defined by WP6. The deployment of a service primarily targets a cloud provider because it offers all the handles (namely APIs) to automate the provisioning and, mostly, the configuration of possibly complex topologies of clusters where the aforementioned services will actually execute. The offloading mechanism, described in detail below (**Section.3.1**), instead is meant to be responsible to allow a seamless extension of any services toward a remote provider. The main assumption of the offloading is that just a unit of work (i.e., a job, a JupyterLab, a function in a serverless system as well as a containerized application etc) will be distributed. Clearly, being a container, the granularity managed by the offloading is much smaller than the one handled by the Resource orchestration. The decision to adopt a container-based approach is fully compliant with requirements and discussions made with WP6.

Thanks to the offloading the high-level service will be able to fully exploit specialised hardware to satisfy very specific workflows such as high performances instead of high throughput. Moreover, the offloading can be considered a suitable solution to accomplish with opportunistic resource usage, for the sake of optimization, which allows to exploit any idle slots anytime, anywhere (i.e., a remote batch system, a single "*fat node*" or a dynamic cloud generated cluster)

Obviously there are few main prerequisites to all this. The main one is that scientific data required by Digital Twins are accessible by the service as well as by the offloaded process. This requires a system to deal with:

- Data handling
- Data management
- Storage federation

These became key in the overall architecture of the interTwin system and in particular in the DTE Infrastructure. In particular it is essential that WP5 provides the following capabilities:

- Transparent data movement
- Federated diverse and dispersed storages and data archival possibly considering disting storage technologies
- Abstraction layers to allow an easy interaction with a complex storage topology
- Tracking and querying dataset locality

Finally, it is worth mentioning that another prerequisite is to ensure that any provider of the federation can actually access the very same version of the code, libraries, and dependencies irrespective of where it lands, through the offloading mechanism.

So, concluding, the aforementioned system can actually work under the assumption that runtime software can be accessed anytime anywhere. Data availability is expected to be granted by the DataLake model and related Data Management services.

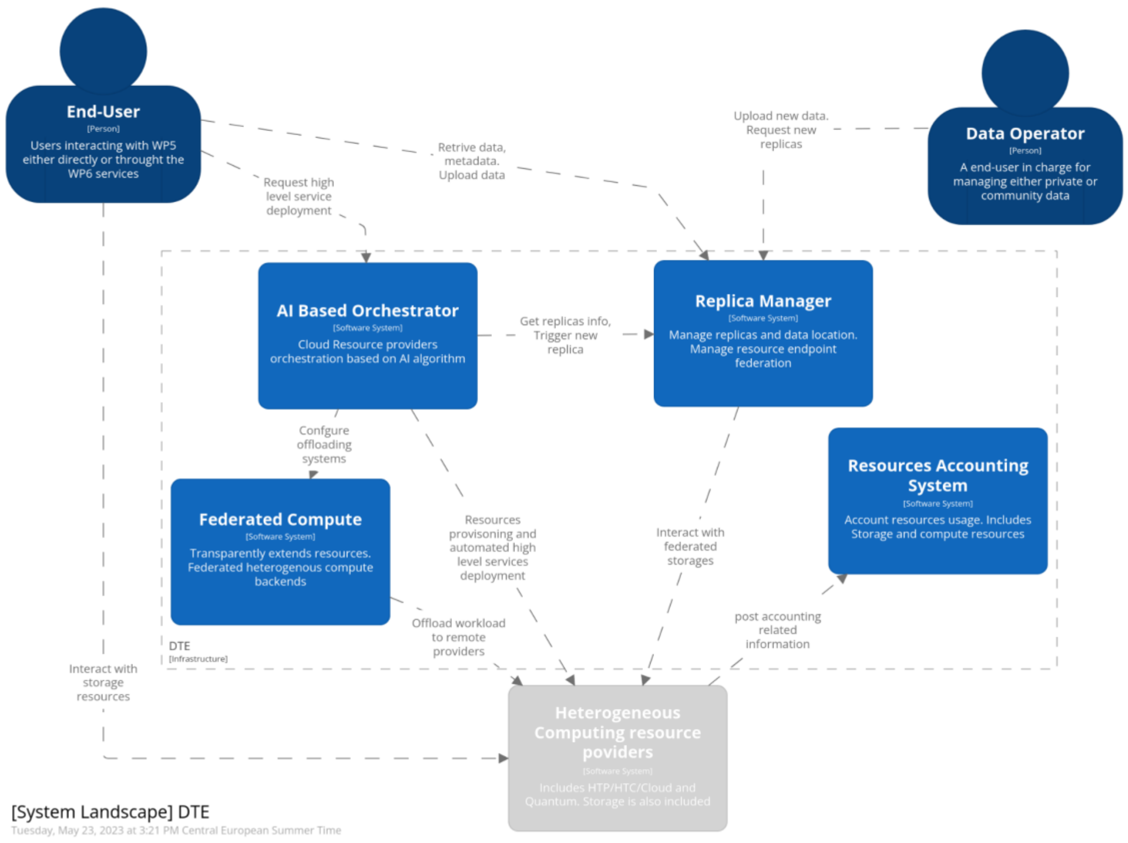

End-User
[Person]
Users interacting with WP5 either directly or throught the WP6 services
Retrive data, metadata. Upload data
Request high level service deployment
Upload new data. Request new replicas
Data Operator
[Person]
A end-user in charge for managing either private or community data
AI Based Orchestrator
[Software System]
Cloud Resource providers orchestration based on AI algorithm
Get replicas info, Trigger new replica
Replica Manager
[Software System]
Manage replicas and data location. Manage resource endpoint federation
Confgure offloading systems
Federated Compute
[Software System]
Transparently extends resources. Federated heterogenous compute backends
Resources provisoning and automated high level services deployment
Interact with federated storages
Resources Accounting System
[Software System]
Account resources usage. Includes Storage and compute resources
Offload workload to remote providers
post accounting related information
DTE
[Infrastructure]
Interact with storage resources
Heterogeneous Computing resource poviders
[Software System]
Includes HTP/HTC/Cloud and Quantum. Storage is also included
[System Landscape] DTE
Tuesday, May 23, 2023 at 3:21 PM Central European Summer Time

Figure 2 - System landscape diagram (in the C4 model) of the DTE Infrastructure. This is a high-level view that highlights the user interactions with WP5.

3.1 General overview of the architecture

Figure 2 represents the context diagram in the C4 model of the overall architecture for the DTE infrastructure that follows what has been summarised in the previous paragraph. The Interaction among the major services is shown. The AI Based Orchestration and the Replica Manager are the main elements and are responsible for deploying services defined by WP6 and moving data between providers respectively. In addition, the relationship with resource providers and external services is represented. In details:

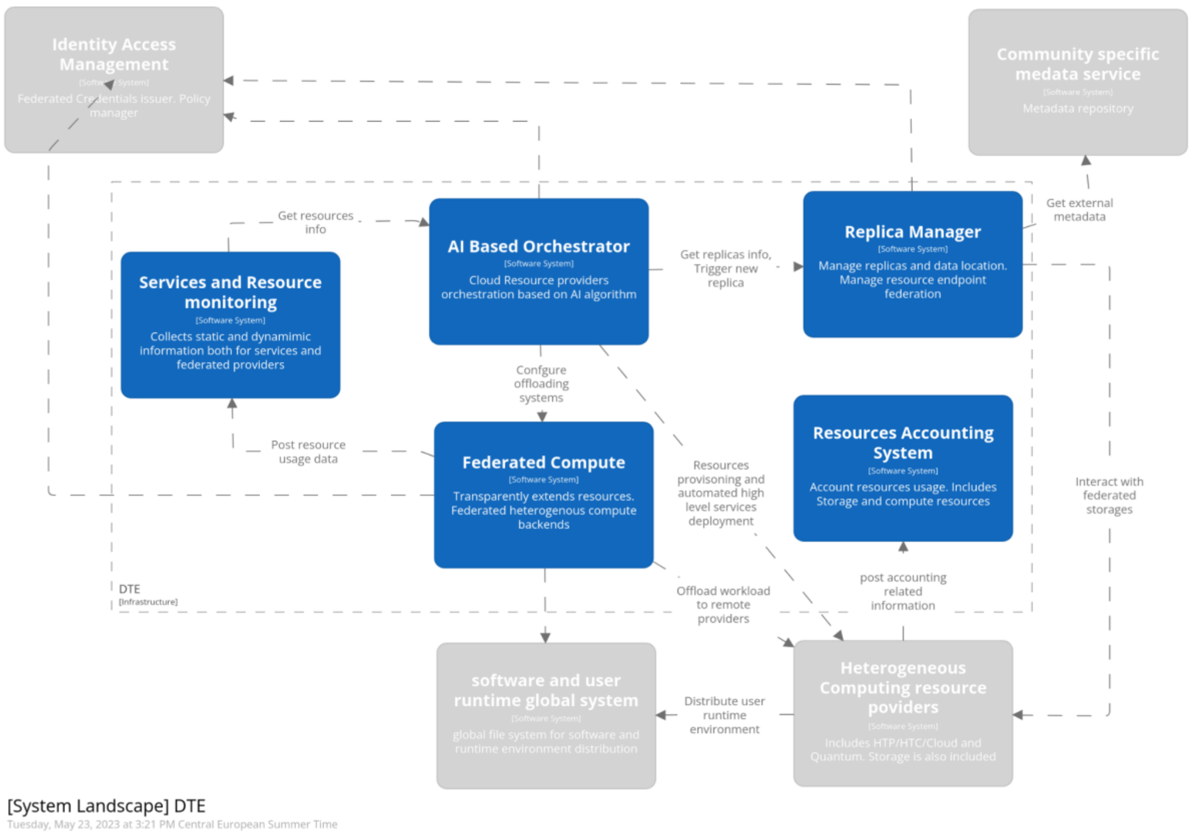
- Services deployed on cloud by the orchestrator are equipped with the offloader that actually allow to seamlessly extend any service toward any type of provider (HPC, HTC, Cloud..). The offloader is the key element to **implement the federated compute.**
- The Resource orchestration is meant to be able to interact with Replica Manager both to ask for data location, in order to possibly consider of it to decide where to deploy a service, and to request to replicate a given dataset.
- Resource info is key to enable the intelligent provider selection for the service deployment.
- All the providers are expected to see the very same repository, implemented by external services, for the software distribution.

- The Replica Manager will be also able to integrate external metadata catalogue in order to possibly use community specific information to take decisions about replicas of data
- Regarding the Authentication and Authorization external identity access management solutions will be integrated. It is foreseen to support federated identities, group based access control
- Services that are expected to enable the offloading mechanism will expose interTwin API (**Section. 4.1.2**). They are responsible for enabling the execution of the offloaded payload, in the form of a container application, into the remote resource. The interTwin API will also allow to track the status of the offloaded payloads and to stop the execution.
- Similarly, to the previous bullet, storage edge service as a component that is deployed "close to" a facility's existing storage capacity is also foreseen. This service will enhance the storage capabilities of a site, if needed, so that it may participate within the federated data infrastructure.

It is important to highlight once more that we envision a high level of flexibility for the infrastructure where components and capabilities can be combined together in order to accommodate specific needs. It is with this vision that we foresee that the latter two components, namely the interTwin APIs for the offloading and the storage edge service, could be combined, lego block, as a single element, with multiple interfaces (microservice approach) to be deployed at the edge (i.e., of a HPC).

Figure 3 shows the high-level interactions between all the aforementioned elements of the architecture and the external services.

The detailed description of the components of the DTE infrastructure are described in detail in the next section (**section. 4**)



Identity Access Management
[Software System]
Federated Credentials issuer. Policy manager
Community specific medata service
[Software System]
Metadata repository
Get resources info
AI Based Orchestrator
[Software System]
Cloud Resource providers orchestration based on AI algorithm
Get replicas info, Trigger new replica
Replica Manager
[Software System]
Manage replicas and data location. Manage resource endpoint federation
Get external metadata
Services and Resource monitoring
[Software System]
Collects static and dynamimic information both for services and federated providers
Confgure offloading systems
Post resource usage data
Federated Compute
[Software System]
Transparently extends resources. Federated heterogenous compute backends
Resources provisoning and automated high level services deployment
Resources Accounting System
[Software System]
Account resources usage. Includes Storage and compute resources
Interact with federated storages
DTE
[Infrastructure]
Offload workload to remote providers
post accounting related information
software and user runtime global system
[Software System]
global file system for software and runtime environment distribution
Distribute user runtime environment
Heterogeneous Computing resource poviders
[Software System]
Includes HTP/HTC/Cloud and Quantum. Storage is also included
[System Landscape] DTE
Tuesday, May 23, 2023 at 3:21 PM Central European Summer Time

Figure 3 - System landscape diagram (in the C4 model) of the DTE Infrastructure. This is a high-level view that highlights the interactions with externals.

3.2 Adoption of existing technologies

The technological objective of interTwin is to develop and implement a prototype of an interdisciplinary Digital Twin Engine offering the capability to integrate with Digital Twins developed in different scientific domains. Not only it is expected to be domain agnostic, but it must be an open source platform based on open standards.

A quite comprehensive scouting process was then made by WP5 in order to identify any suitable software or technological solution to adopt within interTwin and possibly to be enhanced, already available and well established. We started analysing the outcome of several EU Funded projects where of course we find consistency with the "building blocks of the DTE infrastructure of interTwin.

Namely we analysed the results and the blueprints of C-Scale, ESCAPE, EGI-ACE whose blueprints are described in detail in D3.1[**R1**]. and the Knock from The Computer Architecture and VLSI Systems (CARV)[14] where we identified interesting software and existing solutions to be updated. This spans from software that support data access and management in a federated environment, facility orchestration by means of CPU and

[14] https://www.ics.forth.gr/carv/

Storage, resource usage accounting, authentication, and authorization as well as systems to build resources overlay.

Table 2 - Summary table with the results of the scouting performed to identify all the suitable open source components for the design of the architecture. All the services in the table have TRL > 6

Features	Tools
Transfers data into or out of storage services	FTS
Data Management and Storage Orchestration. Replica management	RUCIO System
Provisioning of on-demand high-level cloud-based services and cloud orchestration	INDIGO PaaS Orchestrator
Openid-Connect systems for Authentication and Authorization	Check-IN, INDIGO-IAM
Services and solutions for both users tailored and centrally managed software distribution	CVMFS, unpacked, As well as containers: Singularity, uDocker, Docker
Heterogeneous resource overlay and payload distribution.	HTCondor, Dask
Service for containers orchestration and advanced cloud-native service setup	Kubernetes Virtual Kublet
Abstraction layer to isolate application code from underlying storage technology;	gfal RUCIO Client
Resources usage accounting	APEL
Services providing http/webdav access to resources shared on a filesystem.	Xrootd, storm-webdav, dCache

4 Components

4.1 Federated Compute

The concept of transparently offloading scientific community services to any suitable computing providers has been introduced in the context of task 5.1. The objective is to implement a "continuum of resources" between the centres taking part in the project.

As per requirement analysis, we need to deal with several distinct resource provisioning models, distinct architectures, and quality of services etc. Nevertheless, we do expect the platform to hide such complexity in a way that the user doesn't have to care.

The implementation process needs to consider two categories of sites: 1) sites that provide cloud interface APIs 2) sites offering managed batch systems with minimal requirements. In other words:

- Batch systems are hosted at both HPC and HTC sites and are designed to handle scheduling of discrete jobs and tasks across pools of local resources.
- Cloud providers are more general purposes and are perfectly suitable to deploy high level services, possibly with a rather complex underlying topology and possibly based on cloud-native solutions.

The main challenge is to federate all those heterogeneous and disparate providers, we envision enabling the federation of high-level services (exposed to the users) capable of transparently offloading the inner payloads everywhere. If the offloading mechanism is transparently handled in terms of deployment and configuration, the end user can possibly take advantage of any type of computing capacity, while keeping the same experience as any regular cloud application. In other words:

"The aim is to enable high level services deployed on a cloud provider to transparently execute containerized payload on a remote batch system such as a SLURM on a HPC system. "

4.1.1 General Description and functionalities

We start from the assumption that cloud providers are used to deploy scientific services and consider those services being orchestrated through the Kubernetes system[15]. In addition to that, we envision the usage of clouds as the main target for the actual automated and on-demand service deployment. When it comes to computing capacity, however, we foresee any service capable of acquiring all the needed computing capacity from a resources federation. The model we identified is based on the Virtual Kubelet (VK)[16] technology.

15 https://kubernetes.io/

16 https://github.com/virtual-kubelet/virtual-kubelet

Virtual Kubelet features a pluggable architecture integrated with Kubernetes primitives which make it fully compatible with any workflow based on that platform. The key feature is to masquerade as a Kubernetes Kubelet[17] **which enables connecting Kubernetes to other APIs**. In fact, we find that by adopting an interTwin layer of APIs to be deployed at the edge of any resource provider, we can transparently extend a K8s system running on a cloud system to any remote resource coming from the providers, being either cloud or batch based.

All this translates into the following: the VK integration allows any cloud-native service to take advantage of large batch systems, possibly based on specialised hardware from within Kubernetes, and more importantly, any framework that directly interfaces with Kubernetes API (**Figure 4**).

From a user's perspective this would translate into: "make it easy and transparent the access and the use of GPU based workloads", some examples:

- Creating Kubernetes Jobs which train or execute ML models using GPUs.
- Accessing dedicated nodes on a possibly remote HPC system to perform Interactive processing.
- Distribute preprocessing workflow for data organisation and feature engineering phase.
- Scale out serverless workflow at the pre-exascale.

Translated into the k8s jargon this can be summarised in the following executive message:

"Any pods can be offloaded towards any remote resource without the need for the user to adopt custom configurations or frameworks"

[17] https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/

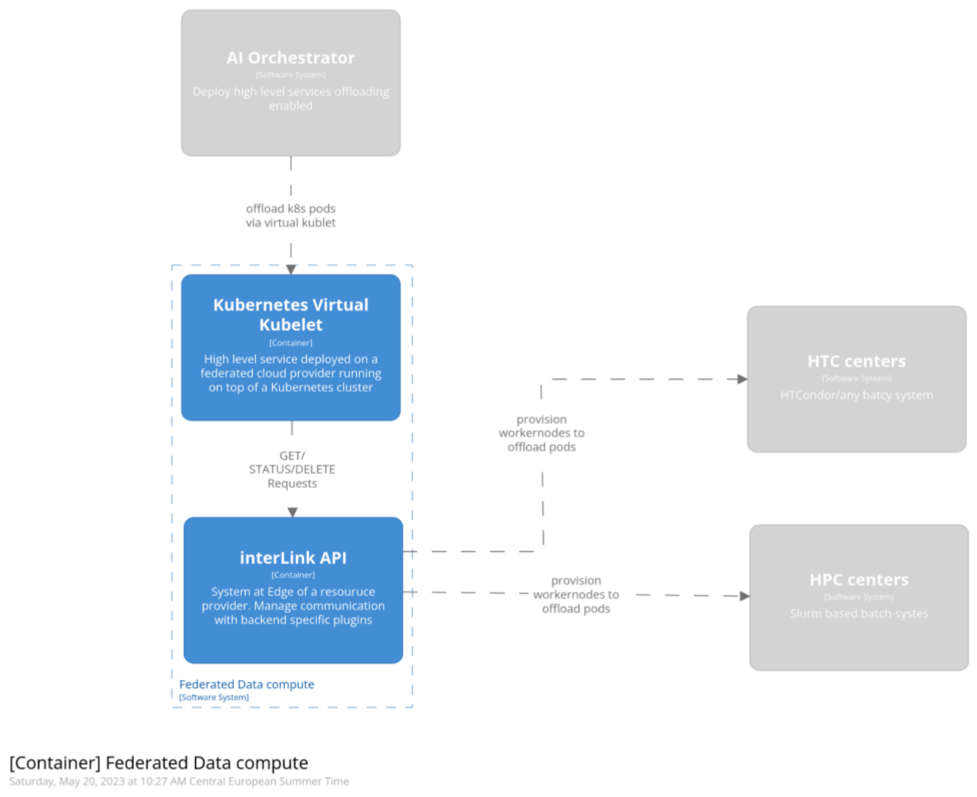

AI Orchestrator
[Software System]
Deploy high level services offloading enabled
offload k8s pods via virtual kublet
Kubernetes Virtual Kubelet
[Container]
High level service deployed on a federated cloud provider running on top of a Kubernetes cluster
GET/ STATUS/DELETE Requests
interLink API
[Container]
System at Edge of a resource provider. Manage communication with backend specific plugins
Federated Data compute
[Software System]
provision workernodes to offload pods
HTC centers
[Software System]
HTCondor/any batcy system
provision workernodes to offload pods
HPC centers
[Software System]
Slurm based batch-systes
[Container] Federated Data compute
Saturday, May 20, 2023 at 10:27 AM Central European Summer Time

Figure 4 - Container diagram (in the C4 model) of the offloading system

4.1.2 Technology stack: the interTwin API

The technical goal is to enable the lifecycle management of a container (eventually running on any engine either Apptainer, Docker et al.) via the Kubernetes API, enabled by Virtual Kubelet technology. An important aspect of the technical implementation is the capability of accommodating any type of backend, via a modular and plugin-base approach. These layers are represented in the following schema.

We named “interLink” the layer API to be developed by the interTwin project. interLink will guarantee a unique interface and thus a standard way for a cloud deployed service to communicate to any external system for the actual payload offloading.

- **Virtual Kubelet:** at this level three main methods will be implemented in order to communicate with the interLink layer. As such there will be the following functions: createRequest, deleteRequest and statusRequest that will use HTTP POST method, DELETE and GET respectively. The interaction with interLink will happen through a REST API.
- **interLink:** This will be the layer managing the communication with the provider plugins. The main responsibilities will be to expose the REST APIs and to provide a plugins-based system to interact with the actual backend.

- **Sidecar** is the name we refer to each plug-in talking with the InterLink layer. Each Sidecar is independent and separately talks with the InterLink layer translating the request for executing container the actual job or set of actions needed to manage the execution on the provider. This is a key feature in order to build a flexible model. In fact, a plugin represents the only piece of the system where the backend specific configuration will be implemented. Thus, if a site has specific needs, this will represent a suitable place where custom modules can be implemented without affecting the overall architecture and implementation. Concretely, as anticipated, if a site runs a HTCondor batch system, the plugin will translate the HTTP requests into HTCondor commands. Similarly, will happen for Slurm etc. Last but not least, if we will need to add a credential mapping, the plugins of the interLink system will be the place where to include it. We plan to begin with Mock module implementation, to return dummy answers. This will represent the test system for the APIs and the aforementioned Virtual Kubelet. Then we will work on a plugin dedicated to Docker, to manage the container lifecycle of a pod through a shell calling the Docker CLI commands. At the time of writing, we chose not to use Docker API to extend modularity and porting to other managers, since the more generic approach is better suited for being reused when going to scenarios where we need to interact with batch systems like Slurm. The latter will be the primary objective that will allow us to start testing the actual offloading mechanism. The described flow is represented in the diagram in **Figure 5**.

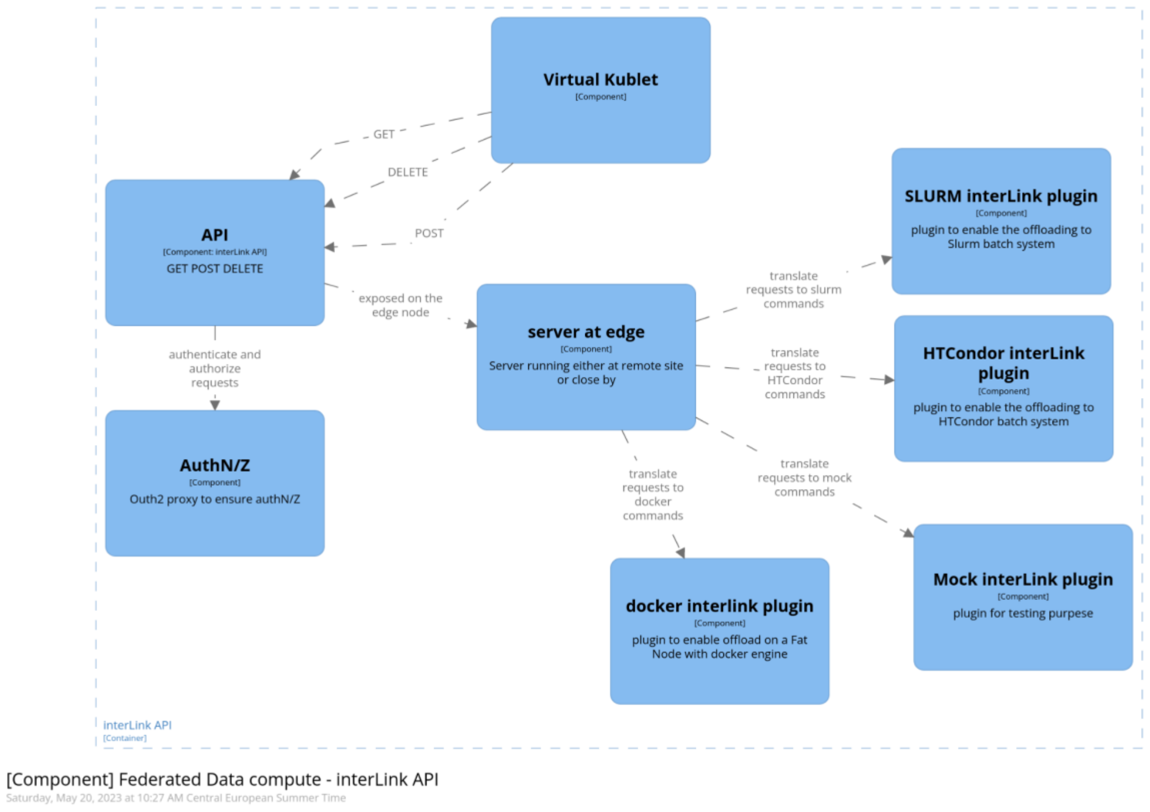

Virtual Kublet
[Component]
GET
DELETE
POST
API
[Component: interLink API]
GET POST DELETE
authenticate and authorize requests
AuthN/Z
[Component]
Outh2 proxy to ensure authN/Z
exposed on the edge node
server at edge
[Component]
Server running either at remote site or close by
translate requests to slurm commands
SLURM interLink plugin
[Component]
plugin to enable the offloading to Slurm batch system
translate requests to HTCondor commands
HTCondor interLink plugin
[Component]
plugin to enable the offloading to HTCondor batch system
translate requests to docker commands
docker interlink plugin
[Component]
plugin to enable offload on a Fat Node with docker engine
translate requests to mock commands
Mock interLink plugin
[Component]
plugin for testing purpese
interLink API
[Container]
[Component] Federated Data compute - interLink API
Saturday, May 20, 2023 at 10:27 AM Central European Summer Time

Figure 5 - Component diagram (in the C4 model) of the offloading system

4.1.3 Interaction with other components

The offloading mechanism interacts primarily with INDIGO PaaS Orchestration. In order to effectively realise what has been designed and described in the current paragraph, we envision that the INDIGO PaaS Orchestrator will be enhanced with the capability to automatically configure the offloader mechanism. This means that every high-level service automatically deployed on a given cloud provider, will be enabled to actually spawn containers on remote sites. The knowledge about sites and relevant information will be tracked by the information system of the INDIGO PaaS Orchestrator. This will be technically described in **section 4.3**.

In addition, the offloading model to be effective and actually exploitable by the scientific communities has few prerequisites that must be fulfilled in order to be functional and effective. There two major aspects:

- Software such as libraries and dependencies must be distributed everywhere and be correctly
- Data must be accessible through the federation of data and storage.

Finally, a third component is the AAI. Authentication of the user and the authorization to instantiate an offloaded process on a remote resource.

4.2 Federated data infrastructure.

In general, the federated data infrastructure has two main responsibilities. First, it must provide the data needed by the training algorithms to build the Digital Twins. Second, it must be able to deploy the trained Digital Twin along with the additional data needed to support any post-processing steps and the final exploitation of the trained DT, in order to gain the greatest insights.

In very general terms, the ability to store and retrieve data is necessary to satisfy this requirement. This data is assumed to be stored as files (in some file system) or data objects (in some object store). This assumption excludes data stored in specialised services, such as a database or domain-specific query services; although such services may be provisioned over underlying data objects.

However, simply storing data is not sufficient. The storage fabric into which data is stored must fulfil the performance and reliability expectations of the researchers in order to be useful. These requirements may be different for different classes of files and may change over time. Such variation may allow for cost optimisation; for example, by storing cold data on magnetic tape, or storing easily reproducible files on low redundant storage.

The interTwin service providers (offering HPC, HTC and Cloud resources) all operate storage fabrics that satisfy the baseline requirements of the experiments. However, collectively they do so through a wide variety of technologies, which expose the functionality through different interfaces and APIs. Some harmonisations may be necessary in order to allow core interTwin services to interact with multiple technologies without having to support multiple storage APIs. Such an abstraction will also make it easier for core interTwin modules to adopt new storage technologies.

In addition, we anticipate that DT training and subsequent exploitation activity will likely take place on different computing infrastructures. Indeed, the exploitation phase could involve complex workflows that are themselves located over different facilities.

In order to satisfy these activities, a framework is needed that can identify potentially large volumes of data in a manageable and reliable fashion, while allowing it to be moved seamlessly between different storage facilities based on the demands of the researchers.

It is worth noting that, in some cases, the input data is provisioned by the computing facility itself; managing this data is not considered part of the interTwin activity. The Copernicus Sentinel satellite data provides an example. Such data is assumed to be available through existing projects such as the C-SCALE project.

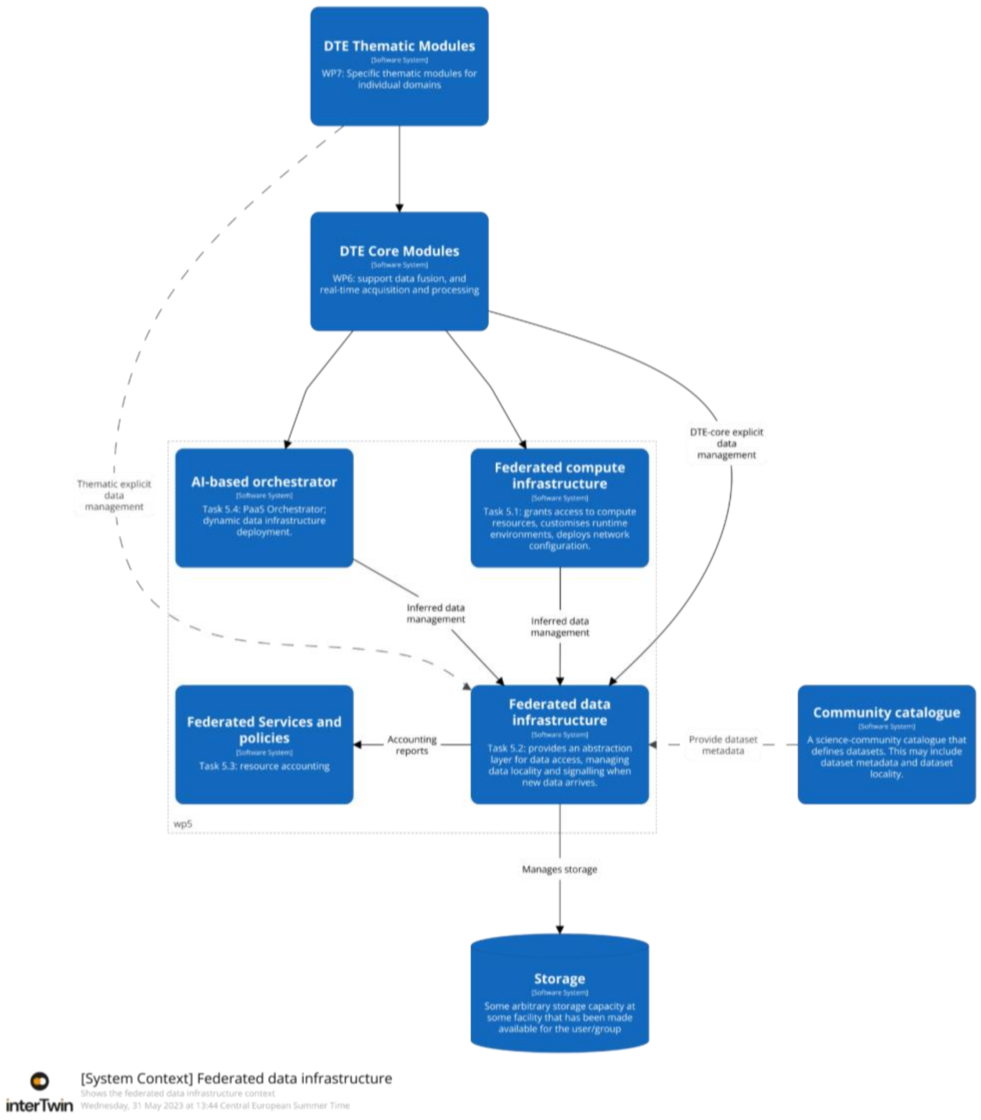

DTE Thematic Modules
[Software System]
WP7: Specific thematic modules for individual domains
DTE Core Modules
[Software System]
WP6: support data fusion, and real-time acquisition and processing
Thematic explicit data management
DTE-core explicit data management
AI-based orchestrator
[Software System]
Task 5.4: PaaS Orchestrator; dynamic data infrastructure deployment.
Federated compute infrastructure
[Software System]
Task 5.1: grants access to compute resources, customises runtime environments, deploys network configuration.
Inferred data management
Inferred data management
Federated Services and policies
[Software System]
Task 5.3: resource accounting
Accounting reports
Federated data infrastructure
[Software System]
Task 5.2: provides an abstraction layer for data access, managing data locality and signalling when new data arrives.
Provide dataset metadata
Community catalogue
[Software System]
A science-community catalogue that defines datasets. This may include dataset metadata and dataset locality.
wp5
Manages storage
Storage
[Software System]
Some arbitrary storage capacity at some facility that has been made available for the user/group
interTwin
[System Context] Federated data infrastructure
Shows the federated data infrastructure context
Wednesday, 31 May 2023 at 13:44 Central European Summer Time

Figure 6 - System Landscape diagram for federated data infrastructure.

The diagram in **Figure 6** shows the system landscape for the DT Engine Infrastructure (WP5). The focus is on federated data infrastructure; therefore, many of the interactions between components not related to data are excluded for simplicity. Although not shown, the researcher will interact directly with the DTE Thematic Modules and possibly with the DTE Core Module. In principle, these thematic modules may interact with the federated data infrastructure; however, the bulk of the data activity will be triggered by DTE Core modules: either from the thematic modules or directly by the researcher. We do not anticipate the researcher interacting directly with the federated data infrastructure.

Although not shown in the diagram, we anticipate that data is written into storage by some ingestor agent. This process provides new data: either for retraining purposes (updating the digital twin based on updated information) or for exploitation activity.

The computing-related tasks within WP5 are also shown interacting with federated data infrastructure. This interaction is related to preparing data locality and/or storage QoS to satisfy the computational demands.

Finally, communities may already have dataset catalogues with which they manage their data. We anticipate such catalogues providing metadata, most prominently information on files and their locality, to the federated data infrastructure, to allow seamless integration with communities existing data practices.

4.2.1 General description and Functionalities

The diagram in **Figure 7** provides an overview of the different code functional elements that combine to make the federated data management layer.

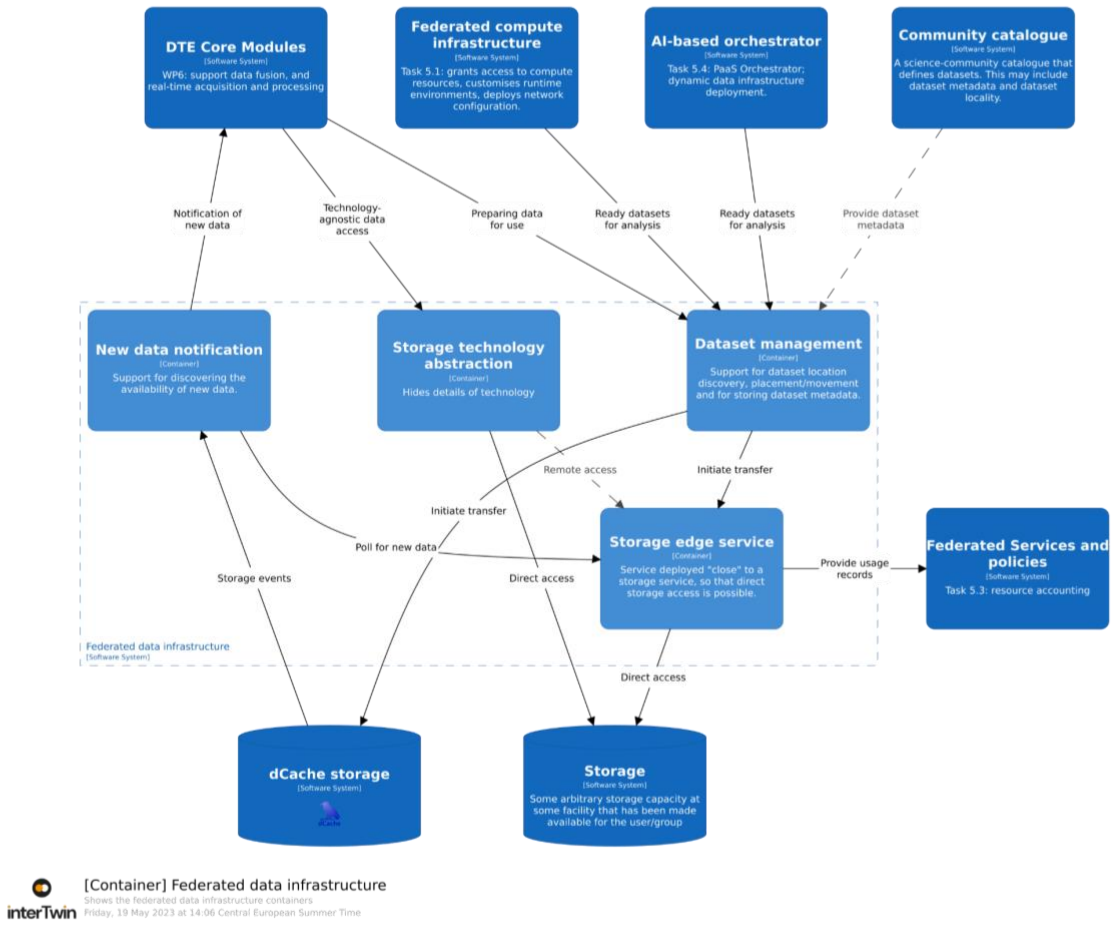

DTE Core Modules
[Software System]
WP6: support data fusion, and real-time acquisition and processing
Federated compute infrastructure
[Software System]
Task 5.1: grants access to compute resources, customises runtime environments, deploys network configuration.
AI-based orchestrator
[Software System]
Task 5.4: PaaS Orchestrator; dynamic data infrastructure deployment.
Community catalogue
[Software System]
A science-community catalogue that defines datasets. This may include dataset metadata and dataset locality.
Notification of new data
Technology-agnostic data access
Preparing data for use
Ready datasets for analysis
Ready datasets for analysis
Provide dataset metadata
New data notification
[Container]
Support for discovering the availability of new data.
Storage technology abstraction
[Container]
Hides details of technology
Dataset management
[Container]
Support for dataset location discovery, placement/movement and for storing dataset metadata.
Remote access
Initiate transfer
Initiate transfer
Poll for new data
Storage edge service
[Container]
Service deployed "close" to a storage service, so that direct storage access is possible.
Provide usage records
Federated Services and policies
[Software System]
Task 5.3: resource accounting
Storage events
Direct access
Federated data infrastructure
[Software System]
Direct access
dCache storage
[Software System]
Storage
[Software System]
Some arbitrary storage capacity at some facility that has been made available for the user/group
interTwin
[Container] Federated data infrastructure
Shows the federated data infrastructure containers
Friday, 19 May 2023 at 14:06 Central European Summer Time

Figure 7 - Container diagram showing the four main containers providing the federated data infrastructure

Each of the four containers in **Figure 7** have a specific responsibility.

- **New data notification** is responsible for accepting requests to monitor a certain directory for new files and to send a notification promptly when new data arrives in that directory.
- **Storage technology abstraction** is responsible for providing a simple interface for interacting with storage that hides details of the underlying storage technology.
- **Dataset management** is responsible for the locality of the files within a dataset, creating additional copies of data to satisfy demand and to remove excessive copies when under space pressure.
- **Storage edge service** is a component that is deployed "close to" a facility's existing storage capacity and enhances its capabilities so that it may participate within the federated data infrastructure.

For the most part, the interactions shown in **Figure 7** are a direct result of the interactions shown in **Figure 6**, with **Figure 7** providing some more detail.

The new data notification container will use the storage edge service to discover when new data is available. This provides a generic solution to this problem; however, some storage technologies (such as dCache) already support new data notification. For storage built on such technologies, the new data notification will interact directly with the service.

The storage technology abstraction provides a POSIX-like API for accessing storage, exploiting a plugin structure. The software using this abstract will either use a plugin that communicates directly with the storage service (e.g., via the POSIX API), or that will interact over the network, using the storage edge service to allow remote access.

In general, the dataset management container will use the storage edge service to support the transfer of data, or removal of data under space pressure. Storage built from technologies that already support the underlying protocols (e.g., EOS, StoRM, dCache) direct communication will be used, instead.

Finally, storage accounting information will be provided by the storage edge service, except where the storage's underlying technology already supports producing the desired accounting records.

4.2.1.1 Data availability and caching

At this time, we anticipate two general classes of data: interTwin-managed data and external data.

External data is any information that is needed by the science use-cases where interTwin is not responsible for that data's availability. Instead, that data is provisioned through some other means, either at an infrastructure level (e.g., C-SCALE) or provided by the scientific community itself.

By contrast, interTwin-managed data is the responsibility of the federated data infrastructure. Following the ESCAPE DataLake model[18], such data is grouped into datasets, which form the most fine-grain level of management.

The ESCAPE DataLake model includes support for caching by only deleting files when a storage location is under space pressure, from storing additional datasets. By selecting the appropriate "high water" mark, an arbitrary percent of the available capacity may be given over to caching.

While an explicit file-level caching service could be deployed, using such a cache would likely result in poor HPC performance. This is because, on a cache miss, the resulting data transfers would take place while the job is running. Such delays would potentially stall HPC nodes until the transfer has completed. In contrast, Rucio is able to provide stronger guarantees on data available by transferring the data prior to starting the HPC job.

Therefore, we currently do not anticipate deploying a dedicated file-level caching service. Of course, this decision is subject to revision based on the experience we will achieve with the early DT integration through the first pilot system that we will develop.

The following sections describe each of the containers, providing details of the technologies involved and highlighting what is the anticipated development work.

4.2.2 New data notification

The elements of the new data notification container are shown in **Figure 8**.

[18] https://projectescape.eu/services/data-infrastructure-open-science-dios

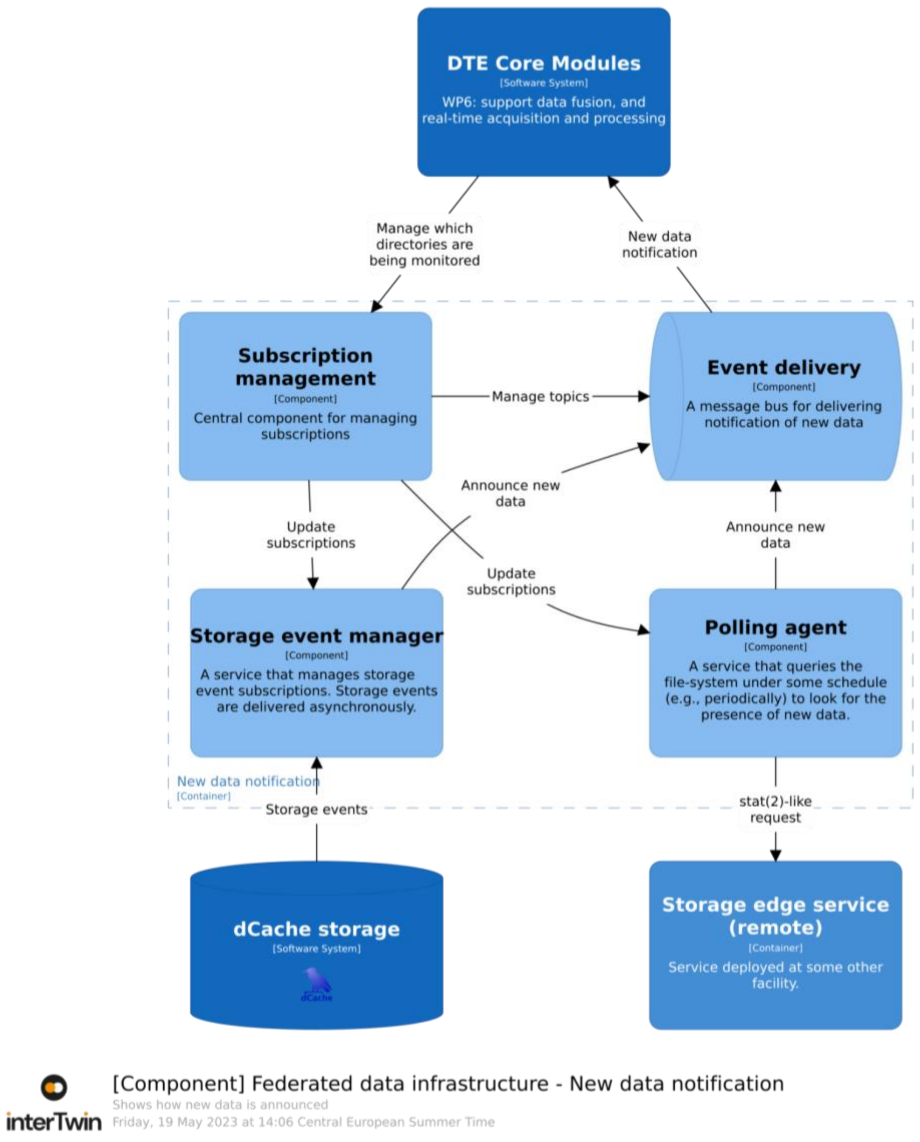

DTE Core Modules
[Software System]
WP6: support data fusion, and real-time acquisition and processing
Manage which directories are being monitored
New data notification
Subscription management
[Component]
Central component for managing subscriptions
Manage topics
Event delivery
[Component]
A message bus for delivering notification of new data
Announce new data
Update subscriptions
Announce new data
Update subscriptions
Storage event manager
[Component]
A service that manages storage event subscriptions. Storage events are delivered asynchronously.
Polling agent
[Component]
A service that queries the file-system under some schedule (e.g., periodically) to look for the presence of new data.
New data notification
[Container]
Storage events
stat(2)-like request
dCache storage
[Software System]
Storage edge service (remote)
[Container]
Service deployed at some other facility.
[Component] Federated data infrastructure - New data notification
Shows how new data is announced
Friday, 19 May 2023 at 14:06 Central European Summer Time
interTwin

Figure 8 - Diagram showing the components making up the new data notification container.

In **Figure 8**, the interactions are shown with DTE Core modules that should be notified on the arrival of new data. In principle, this interaction might also be needed by DTE Thematic Modules, as indicated in Figure 3. In the interests of reducing the complexity of this diagram, such interactions have been omitted, but could be imaged by substituting "DTE Thematic Modules" for the "DTE Core Modules" shown.

The **Subscription Management** component provides an API through which the different modules may register their interest in the arrival of data. Such interest provides the name of a topic to which the client may subscribe and receive new data events.

The events are delivered through the **Event delivery** component. This handles the reliable and timely delivery of new-data notification to interested parties. If a client is disconnected then this component will store any new-data events the client missed and provide them once the client reconnects.

The **Storage Event Manager** component is a centrally run component that manages the event subscription facility already present with certain technologies (e.g., dCache). The advantage of this approach is that it does not require polling, providing rapid notification of new data without continuously querying the underlying storage. If the facility has provisioned storage using technologies that do not support remote notification of new data then the polling agent provides a similar rule to the Storage Event Manager.

dCache storage is a storage service built from the open source dCache software. This is an example of a storage technology that already supports notification, allowing a client (in this case, the Storage Event Manager) to learn of new data without polling.

The **Polling agent** is a centrally run component that manages the polling of some underlying storage that does not support new data notification. It uses the Storage Edge Service to detect when new data is available. The initial version will have the Polling agent querying the Storage Edge Service periodically, to learn if new data is available. Depending on demand and available resources, an enhancement will be investigated where the Storage Edge Service will poll for new data, providing a new data notification.

The Event Delivery component will be based on a standard open-source event delivery, such as Apache Kafka. The other components will be developed by interTwin.

4.2.3 Storage technology abstraction

The components of the Storage Technology Abstraction container are shown in **Figure 9**.

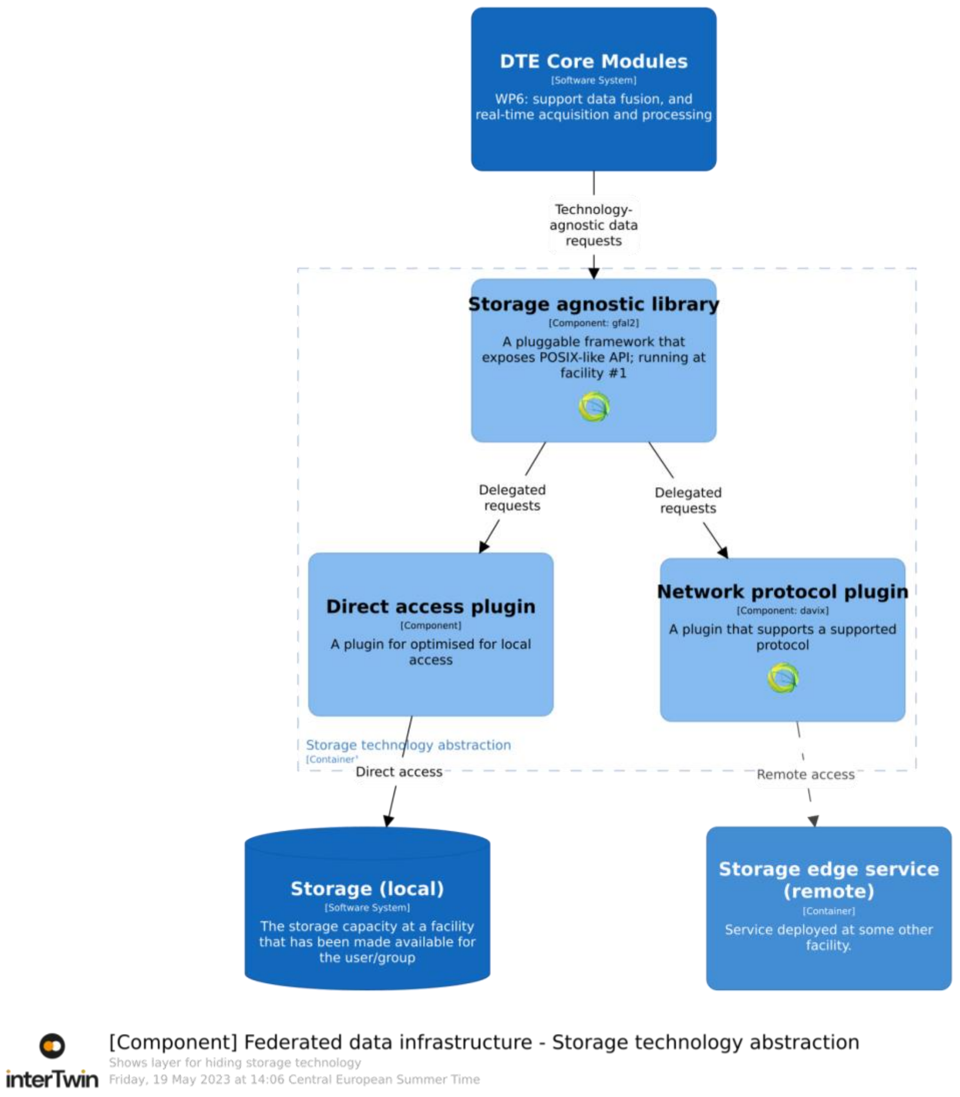

DTE Core Modules
[Software System]
WP6: support data fusion, and real-time acquisition and processing
Technology-agnostic data requests
Storage agnostic library
[Component: gfal2]
A pluggable framework that exposes POSIX-like API; running at facility #1
Delegated requests
Delegated requests
Direct access plugin
[Component]
A plugin for optimised for local access
Network protocol plugin
[Component: davix]
A plugin that supports a supported protocol
Storage technology abstraction
[Container]
Direct access
Remote access
Storage (local)
[Software System]
The storage capacity at a facility that has been made available for the user/group
Storage edge service (remote)
[Container]
Service deployed at some other facility.
[Component] Federated data infrastructure - Storage technology abstraction
Shows layer for hiding storage technology
Friday, 19 May 2023 at 14:06 Central European Summer Time
interTwin

Figure 9 - Diagram showing the components making up the storage technology abstraction container.

The **storage agnostic library** provides a POSIX-like interface through which applications can access data independently of the underlying technology. It does this by providing an API that software can link against.

Figure 9 shows the DTE Core Modules taking advantage of the storage agnostic library API. In principle, this interaction might also be needed by DTE Thematic Modules, as indicated in **Figure 6**. In the interests of reducing the complexity of this diagram, such interactions have been omitted, but could be imaged by substituting “DTE Thematic Modules” for the “DTE Core Modules” shown.

The library will support various plugins, each supporting a different storage technologies or network protocol. **Figure 9** shows two such plugins.

The **Direct Access Plugin** will provide access to the underlying filesystem. In effect, provides a simple pass-through, allowing POSIX access to an underlying POSIX filesystem.

The **Network Protocol plugin** allows remote access to data, with the Storage Edge Service providing support for remote access.

This plugin structure allows for software to take advantage of new storage technologies without requiring any changes. For example, a new plugin may be developed to provide compatibility with Destination Earth's storage technology.

The storage agnostic library is based on gfal2, an open-source solution from CERN. An initial set of plugins will be based on those supplied by gfal2. Some development effort may be needed to support interTwin use-cases.

4.2.4 Dataset Management

The components of the Dataset Management container are shown in **Figure 10**.

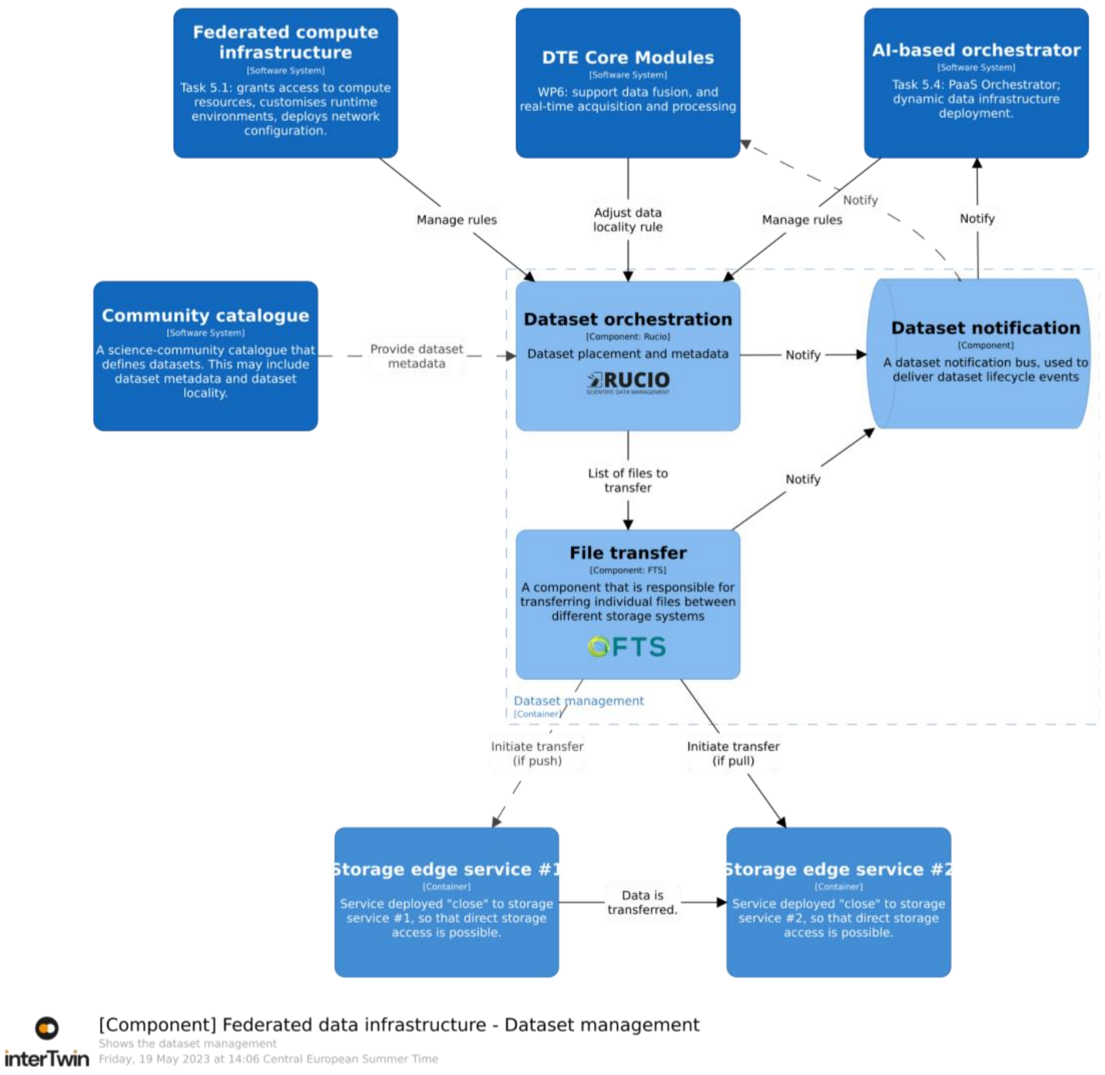

Federated compute infrastructure
[Software System]
Task 5.1: grants access to compute resources, customises runtime environments, deploys network configuration.
DTE Core Modules
[Software System]
WP6: support data fusion, and real-time acquisition and processing
AI-based orchestrator
[Software System]
Task 5.4: PaaS Orchestrator; dynamic data infrastructure deployment.
Manage rules
Adjust data locality rule
Notify
Manage rules
Notify
Community catalogue
[Software System]
A science-community catalogue that defines datasets. This may include dataset metadata and dataset locality.
Provide dataset metadata
Dataset orchestration
[Component: Rucio]
Dataset placement and metadata
RUCIO
Notify
Dataset notification
[Component]
A dataset notification bus, used to deliver dataset lifecycle events
List of files to transfer
Notify
File transfer
[Component: FTS]
A component that is responsible for transferring individual files between different storage systems
FTS
Dataset management
[Container]
Initiate transfer (if push)
Initiate transfer (if pull)
Storage edge service #1
[Container]
Service deployed "close" to storage service #1, so that direct storage access is possible.
Data is transferred.
Storage edge service #2
[Container]
Service deployed "close" to storage service #2, so that direct storage access is possible.
[Component] Federated data infrastructure - Dataset management
Shows the dataset management
Friday, 19 May 2023 at 14:06 Central European Summer Time
interTwin

Figure 10 - Diagram showing the components making up the dataset management container.

In **Figure 10**, the interactions are shown with DTE Core modules that adjust dataset locality by adding or removing locality rules. In principle, this interaction might also be needed by DTE Thematic Modules, as indicated in **Figure 6**. In the interests of reducing the complexity of this diagram, such interactions have been omitted, but could be imaged by substituting "DTE Thematic Modules" for the "DTE Core Modules" shown.

The dataset management container is based on the ESCAPE Data Lake stack. This is a set of components that were established during the EU-funded ESCAPE project as a scalable, cross-domain solution for managing dataset locality.

The **Dataset Orchestration** is a single, community-centric component. It is responsible for establishing the optimal data locality, based on the multiple declarative rules supplied by the clients. It is aware of where data is actually located. It then reconciles these two by transferring data to satisfy the demand. It can also delete data to satisfy spare capacity requirements.

The Dataset Orchestrator supports direct dataset management by accepting rules from DTE Core Modules (or DTE Thematic Modules). It also supports implicit dataset management, where Federated computing infrastructure or the AI-based orchestrator adjust the dataset locality rules as implicit dataset management: data replication to support DT training or exploitation activity.

Some communities already have a dataset concept, which they maintain in their own Community Catalogue. This is something that the Dataset Orchestration may need to consult to understand the dataset placement rules from the community.

The **File Transfer** component is responsible for transferring data effectively between endpoints, where the Dataset Orchestration has identified that an additional replica is needed.

The File Transfer component manages transfers at scale; however, the individual transfers are achieved by services running at the site. Some storage technologies support the underpinning protocols natively; for storage built on such technologies, the File Transfer component will talk directly with the storage. Otherwise, the Storage Edge Service will enable the data transfer.

Both the Dataset Orchestration and File Transfer components feed notification of their activity to a **Dataset Notification** component. This component is responsible for providing low-latency information about dataset availability, as this changes due to the application of new rules. This allows for rapid orchestration of services to satisfy demand once the required datasets have the desired locality.

The Dataset Orchestration component is based on the open-source Rucio[19] project and the File Transfer component is based on the open-source FTS software both are from CERN. Rucio will need to be extended to support Community Catalogues. Further customisations are likely to support interTwin-specific use-cases.

4.2.5 Storage edge service

The components of the Storage edge service container are shown in **Figure 11**.

[19] https://rucio.cern.ch/

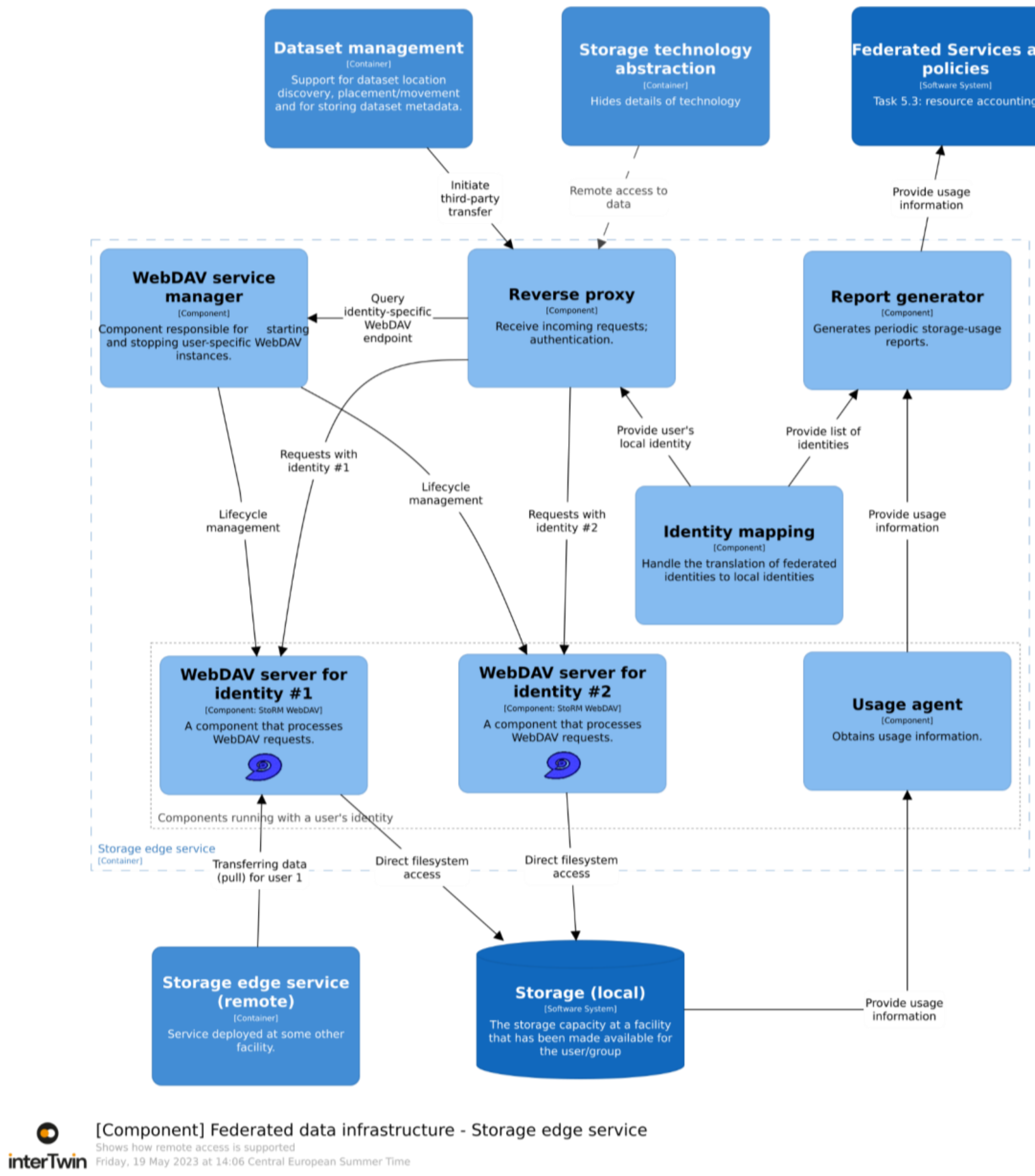

Dataset management
[Container]
Support for dataset location discovery, placement/movement and for storing dataset metadata.
Storage technology abstraction
[Container]
Hides details of technology
Federated Services and policies
[Software System]
Task 5.3: resource accounting
Initiate third-party transfer
Remote access to data
Provide usage information
WebDAV service manager
[Component]
Component responsible for starting and stopping user-specific WebDAV instances.
Query identity-specific WebDAV endpoint
Reverse proxy
[Component]
Receive incoming requests; authentication.
Report generator
[Component]
Generates periodic storage-usage reports.
Provide user's local identity
Provide list of identities
Requests with identity #1
Lifecycle management
Lifecycle management
Requests with identity #2
Identity mapping
[Component]
Handle the translation of federated identities to local identities
Provide usage information
WebDAV server for identity #1
[Component: StoRM WebDAV]
A component that processes WebDAV requests.
WebDAV server for identity #2
[Component: StoRM WebDAV]
A component that processes WebDAV requests.
Usage agent
[Component]
Obtains usage information.
Components running with a user's identity
Storage edge service
[Container]
Transferring data (pull) for user 1
Direct filesystem access
Direct filesystem access
Storage edge service (remote)
[Container]
Service deployed at some other facility.
Storage (local)
[Software System]
The storage capacity at a facility that has been made available for the user/group
Provide usage information
interTwin
[Component] Federated data infrastructure - Storage edge service
Shows how remote access is supported
Friday, 19 May 2023 at 14:06 Central European Summer Time

Figure 11 - Diagram showing the components making up the storage edge service.

Unlike the other containers in the federated data infrastructure the storage edge service is deployed locally to the storage; for example, this container would be deployed “close to” HPC facilities. Ideally, the storage edge service would be deployed within the facility's local service fabric and would be maintained by that facility as part of the service portfolio they offer to their users. The “close to” caveat covers the possibility that the edge service is not run by the facility itself, but on resources located somewhere with sufficient network connection.

The storage edge service’s main goal is to mediate access to the facility’s storage capacity, allowing other federated data infrastructure containers to function. The facility may already run services that provide the necessary functionality, in which case the existing services may be used directly by the other federated data infrastructure containers.

Requests from the Dataset Management container or the Storage Technology Abstraction are accepted by the **Reverse Proxy** component. Although this component has overall responsibility, it delegates most responsibility to other components.

The **Identity Mapping** component is responsible for identifying the corresponding local user for an incoming request. The Reverse Proxy uses this to understand the identity of the incoming request in terms of the local users.

The **WebDAV Service Manager** component is responsible for the lifecycle of an identity-specific WebDAV endpoint. It will start the service if one is not already started, provide the port number of the running service, allowing the Reverse Proxy service to forward an incoming request.

The **Identity-specific WebDAV Server** component runs as a specific user. This allows it to have access to the local Storage, while honouring the permission model on that filesystem. Similarly, any files written by such a WebDAV server will have the correct ownership.

The **Report Generating** component is responsible for generating storage usage records and delivering them to Federated Services and Policies system, according to the desired reporting schedule.

In many cases, the Report Generating component will be unable to obtain the required data running as a generic user. Therefore, it is anticipated that a **User Agent** runs with the identity of the local user, to provide the necessary information to generate the storage usage records.

The Reverse Proxy component will be based on a standard open-source reverse proxy, such as NGINX. Similarly, the WebDAV server will be based on the open-source StoRM WebDAV software. Some customisations of these components may be needed. The other components (WebDAV service manager, Identity Mapping, Report Generator and Usage Agent) will largely be written within interTwin, reusing existing components whenever possible.

4.2.6 Interaction with other components

Most components will interact with Rucio for federated data management aspects, or directly with some storage fabric to read data from a file. Such interactions include querying the locality of data, specifying that data should be copied elsewhere.

Workflow engines may trigger data transfer; however, they would then need to wait until the data is available before starting the analysis workflow.

4.3 Intelligent providers orchestration

4.3.1 General description and Functionalities

Resource orchestration is a critical element in the DTE infrastructure architecture. Its main function is to automate the deployment and setup of user-defined services, as well as make decisions regarding provider selection. The orchestration system is expected to be capable of making intelligent identification of the best Cloud provider by utilising a variety of features based on both static and dynamic metrics. The static metrics can be defined in advance, while dynamic information must be collected at runtime and continuously updated.

Furthermore, the orchestration system will be designed to enable and support the workload offloading mechanism that was previously described in **Section. 4.1** by meaning that on-demand deployed services will be automatically configured to offload payloads. For the system to successfully implement the workload offloading mechanism, it is essential to include and rely on a detailed information system that collects and publishes information about the different heterogeneous providers in the federation. This information system will allow access to key features and capabilities of the providers, such as computing capacity, specialised hardware, and data location. With this information, the orchestration system will be able to spawn services that are ready for making workload offloading, without requiring any further configuration by the user. The deployed service will be set up with all the necessary configurations for offloading, thus making the process completely transparent for the user. Resource orchestration will be exposed directly to the WP6 managed services. Being based on the INDIGO PaaS Orchestration system, the interaction will be based on TOSCA language and APIs. On the other side, the interaction of such a layer will be with Cloud resources providers. There are few requirements for a site to fulfil for an effective integration. Those span from the support of an OAuth2 authentication mechanism, configuration of the service account to grant access to the federated projects, and information to configure with PaaS operators such as name of the site, name of the project configured for a specific Virtual Organization, quotas for each federated project endpoints.

4.3.2 Technology stack

The following components constitute the technology stack for the orchestration system:

- **INDIGO PaaS Orchestrator**: built with Java technologies and on the open-source Workflow Manager "Flowable", the Orchestrator allows federating heterogeneous resource providers and orchestrating the deployment of TOSCA templates, selecting the best provider according to criteria like data location, SLA, and monitoring information. It provides a set of APIs to create, monitor, and manage the deployments.
- **PaaS Dashboard**: a Flask application with a SQL database that enables users to interact easily with the services of the PaaS, particularly the Orchestrator, to create TOSCA-based deployments. The dashboard provides a user-friendly interface for managing and monitoring deployments.
- **Infrastructure Manager**: a Python tool designed to simplify the access and usability of IaaS clouds for scientific applications. It automates the selection, deployment, configuration, monitoring, and update of virtual appliances. It supports APIs from various cloud platforms, provides a contextualization system for installing and configuring user-required applications, and offers multiple access methods including a web-based GUI, an XML-RPC API, a REST API, and a command-line application.

- The **Federated Service Catalogue** is an *upcoming component* that will gather and publish information about the federated providers, their services, and resource availability. Automatic and dynamic registration of new providers/services will be supported via REST API.
- **Monitoring and Metering System**: this *new component* will collect metrics from the federated providers in order to monitor their status and provide the Orchestrator with useful information for selecting the optimal deployment path. This component has a distinct role with respect to the resource accounting system described in **Section. 4.4**.
- **Cloud Provider Ranker**: a Java application built on top of the open-source Rule Engine "Drools" that computes a rank for each federated provider considering parameters like the resources quota provided to the users groups, and the monitoring information (services availability, performance metrics). This component will be enhanced in order to improve the scheduling mechanism of the Orchestrator.

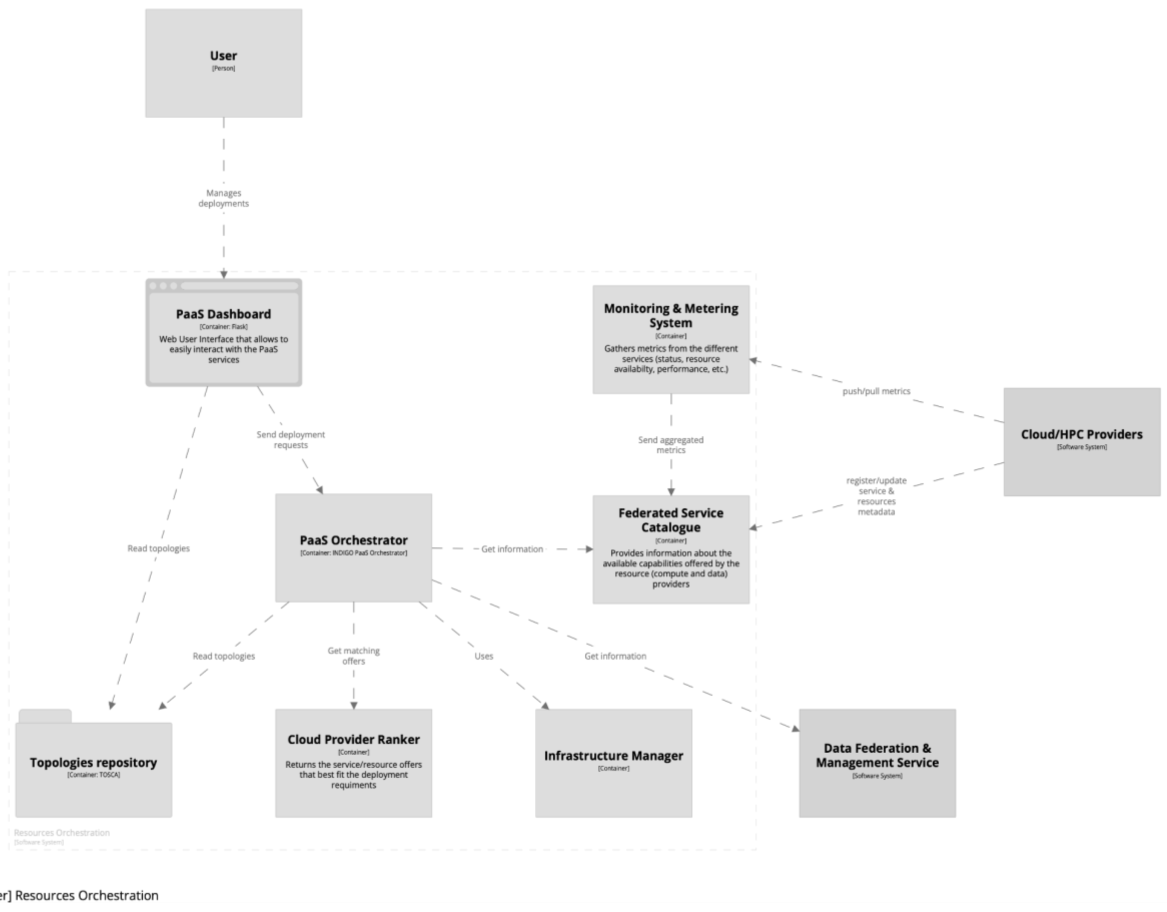

User
[Person]
Manages deployments
PaaS Dashboard
[Container: Flask]
Web User Interface that allows to easily interact with the PaaS services
Monitoring & Metering System
[Container]
Gathers metrics from the different services (status, resource availabilty, performance, etc.)
push/pull metrics
Cloud/HPC Providers
[Software System]
Send deployment requests
Send aggregated metrics
register/update service & resources metadata
Read topologies
PaaS Orchestrator
[Container: INDIGO PaaS Orchestrator]
Get information
Federated Service Catalogue
[Container]
Provides information about the available capabilities offered by the resource (compute and data) providers
Read topologies
Get matching offers
Uses
Get information
Topologies repository
[Container: TOSCA]
Cloud Provider Ranker
[Container]
Returns the service/resource offers that best fit the deployment requiments
Infrastructure Manager
[Container]
Data Federation & Management Service
[Software System]
Resources Orchestration
[Software System]
[Container] Resources Orchestration

Figure 12 - Diagram showing the components making up the PaaS Orchestration services.

4.3.3 Interaction with other components

The Resource Orchestration system will work with the Cloud interacting directly via the cloud-exposed API while with HPC Providers the interaction will not be direct but through the dedicated offloading mechanism. Moreover, the Resource Orchestration will communicate with the Data Management and Federation System to optimise the deployments performance and resource allocation. In particular, the Federated Service Catalog will receive information about the provider services and resources through specific tools that will be developed. Monitoring metrics will be included as well through the interaction with the Monitoring & Metering System that will act as a gateway for collecting the metrics coming from the different providers (and eventually from external monitoring systems). This information will be consumed by the Orchestrator together with the information provided by the Data Management and Federation Service (data availability and location, replicas, etc.) in order to identify the optimal path for the deployment. Moreover, in the "offloading-enabled" scenario, the Orchestrator will leverage the information from the Federated Service Catalog to prepare the needed service configuration. Regarding the authentication and authorization, the resource orchestration will interact with external JWT issuers.

4.4 Accounting

4.4.1 General Description and functionalities

To ensure resources in a heterogeneous distributed environment are allocated and used fairly, a centralised resource accounting system is needed to aggregate usage records from across the infrastructure.

APEL[20] is a compute resource usage accounting tool that collects and processes usage data from resource providers participating in distributed infrastructures. Typically, accounting data is collected from different probes run at the resource provider level and then sent to a central repository, where it is processed to generate statistical summaries that are available through visualisations. APEL encompasses the open source client and server software, as well as the service that's centred around a central accounting repository, all developed and run by UKRI STFC[21]. APEL currently supports a number of types of accounting including for grid batch jobs, cloud virtual machines, and storage space[22]. It also supports a range of different systems within those types such as different batch systems or cloud platforms.

[20] https://github.com/apel/apel

[21] https://www.ukri.org/councils/stfc/

[22] https://docs.egi.eu/providers/high-throughput-compute/storage-accounting/

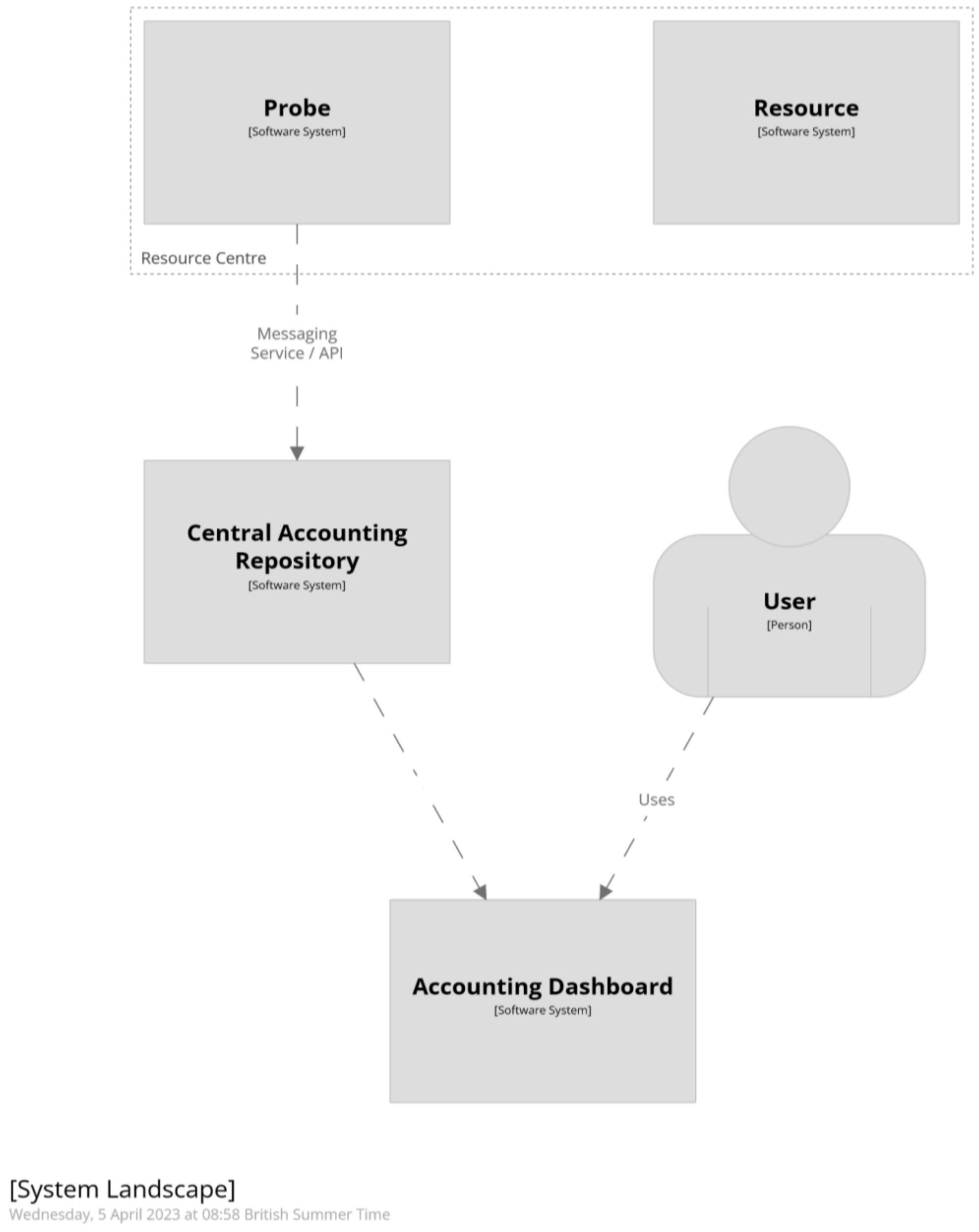


Figure 13 - Diagram showing high-level overview of the APEL accounting workflow

4.4.2 Technology stack

APEL uses MySQL relational database schemas to store accounting records, currently implemented for MariaDB[23], with a Grafana dashboard to display the data. Records from resource providers are sent via the ARGO Messaging Service (AMS)[24] using the APEL SSM[25] messaging component.

[23] https://mariadb.org/en/

[24] https://argoeu.github.io/argo-messaging/

[25] https://github.com/apel/ssm

As mentioned in the previous section, APEL already supports a number of resource platforms, but developments will be needed to support the additional platforms that interTwin will use.

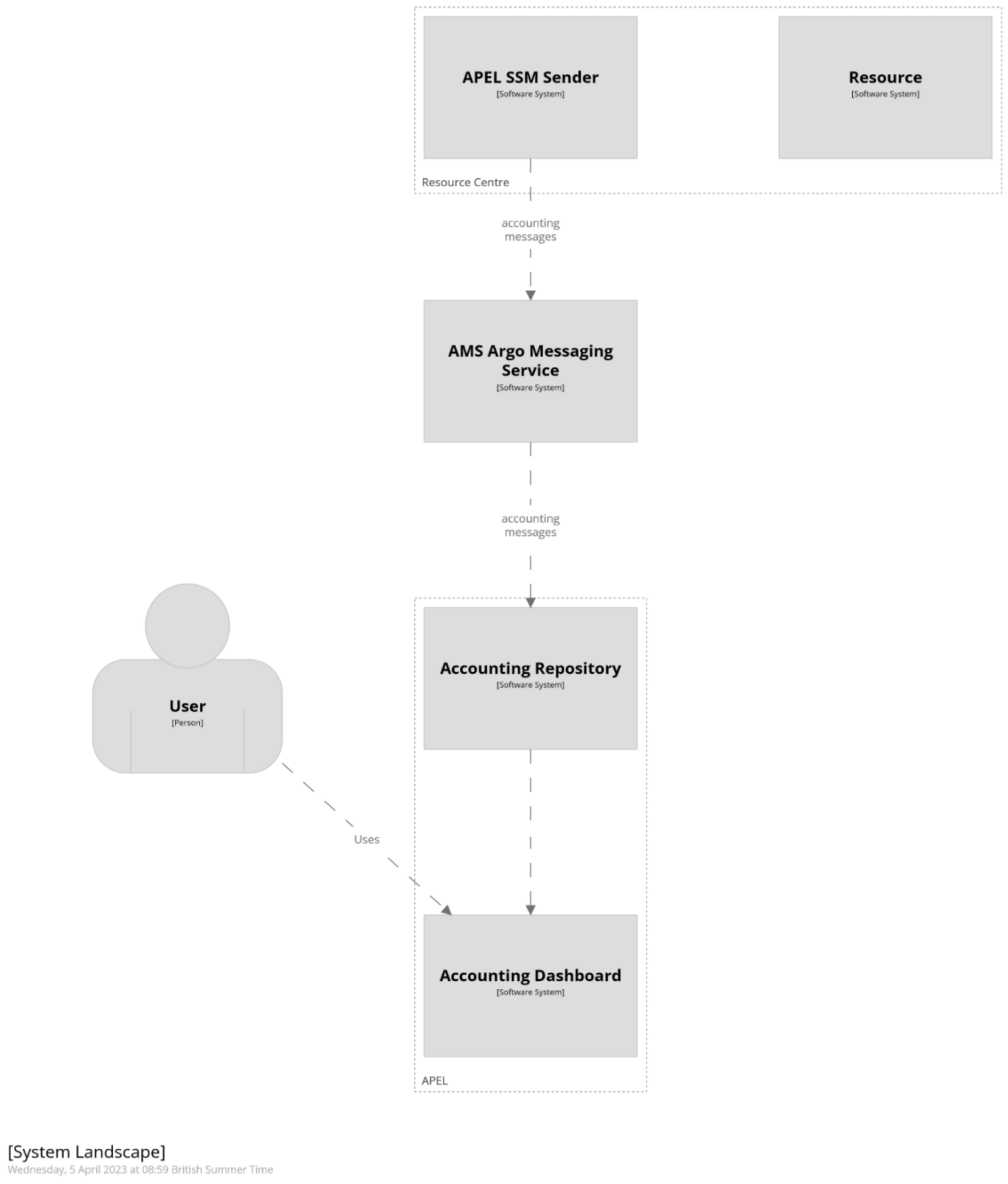


Figure 14 - Diagram showing the service components of the APEL accounting system.

Developments required:

- Probes for the interTwin HPC environment
    - Scripts and tooling that can interact with the interTwin HPC and container (Kubernetes) environments, to extract accounting data and write it into an APEL-compatible format

- Developing an interTwin specific accounting dashboard to explore the data

Deployment steps:

- Creating and configuring messaging queues in AMS, for resource providers to send accounting records via. This includes agreeing on AAI methods
- Deploying a dedicated central database using APEL schemas
- Deploying a Grafana instance with the interTwin accounting dashboard to display the data from the central database

4.4.3 Interaction with other components

APEL interacts with resource providers via the resource usage records that get forwarded to the Accounting Repository through AMS. The records need to be generated to an APEL specific schema and they are then bundled into messages and sent using the APEL SSM messaging component.

For OpenStack cloud systems, resource providers can use the cASO[26] accounting reporter to extract usage records from OpenStack for sending to an APEL repository.

For grid batch systems, the APEL client software comes with a variety of parsers for different batch systems or alternatively resource providers can use the in-built support for APEL format records that certain batch systems come with.

[26] https://caso.readthedocs.io/en/stable/

5 Policies for resource access

5.1 Background

Secure and sustainable operation of any technical implementation of interoperable DTs can only be achieved if based on a trusted platform. The technical challenges posed by interTwin's core objectives push the requirements for such a platform beyond existing distributed computing models to encompass diverse research communities accessing highly heterogeneous resources seamlessly through orchestration. Enabling such interoperability in a secure manner across this diversity and heterogeneity requires not only technical solutions but also a security policy framework establishing transitive trust between all participants: communities must trust that their data is safe, and results are reliable; resource providers must protect their resources from misuse and account for use.

The Authentication and Authorisation for Research Collaborations (AARC)[27] projects published the AARC Blueprint Architecture, implementing federated access management solutions for international research collaborations, together with a Policy Development Kit (PDK) of nine template policy documents to regulate and facilitate the trust requirements outlined above. The AARC outputs have been used by a number of large-scale research communities[28]. In addition, the PDK provides a basis for the implementation of an operational incident response capability (Sirtfi[29]) and the trustworthy operation of infrastructure services (Snctfi framework[30]) in a distributed, federated environment, both derived from requirements arising from The SCI Trust Framework[31], endorsed by a number of research infrastructures.

Whilst the deployment of Cloud and HTC compute and storage resources within a distributed federated environment has been widely adopted in the research domain, access to HPC remains typically based on local account management. Access to such resources with the scale and ease foreseen within the interTwin objectives, whilst retaining appropriate security and accountability is one of the policy challenges for the project. Linked to account management is the problem of presenting multiple, diverse terms of use agreements (AUPs etc.) from each potential resource at which a user's work (compute or storage of data) to users who do not have a direct relationship with the resource owner. The WISE AUP, providing a common baseline set of rules governing users' behaviour, discussed below, is one solution proposed for this problem.

[27] AARC (aarc-community.org)

[28] AARC in Action – AARC (aarc-community.org)

[29] Sirtfi – AARC (aarc-community.org)

[30] Snctfi – AARC (aarc-community.org)

[31] WG: Security for Collaborating Infrastructures (SCI-WG) – WISE Community (wise-community.org)

Similarly, establishing a common security baseline for the operation of resources, with provisions including incident response, traceability and personal data handling provides minimum expectations and requirements of the behaviour of those offering services to users and communities. Here, we consider the EOSC Security Operational Baseline 2022[32,33], accepted as part of the EOSC Interoperability Framework[34], to be a very useful evolution of the AARC PDK, and that this could provide the basis for starting consultation with the resource providers and other stakeholders regarding interoperable policies.

To help establish interTwin providers' alignment, if any, with the WISE AUP and EOSC IF Security Operational Baseline, and to begin to highlight problems in the adoption of such baseline policies in the interTwin DT environment, three questions were inserted into the WP5 questionnaire, described in section 2, to resource providers: "How do you grant access to compute and data resources?", "What are the requirements for users' acceptance of terms and conditions for resource usage?" and "What is your alignment with the EOSC Security Operational Baseline?". Answers to these questions provide an overview of the problem space for further policy development and engagement with participants. The answers provided in the responses to the questionnaire have been examined and no resource provider has identified any problem with starting development of the interTwin policies on templates based on the WISE AUP and the EOSC Security Operational Baseline so that is indeed where we plan to start.

Whilst the policy-driven requirements on resource and service providers may be driven by interoperability across other resources and services, establishing trust in and across a broad range of user communities, of varying scale and organisation will be an ongoing challenge. The development of the DT policies for resource access will also have to take the providers existing policies and national and EU legal requirements (e.g., NIS2 and GDPR) into account.

5.2 Provider and community onboarding

"Onboarding" means registering and publishing resources to a portal or other access to an IT infrastructure. As an example of how this is done elsewhere, the EOSC Onboarding Team - EPOT[35] - follows operational procedures[36] on the EOSC Portal to facilitate the addition of new Resource Providers and Resources and abides by the EOSC Inclusion criteria[37]. It also ensures that resources continue to adhere to these criteria.

[32] EOSC Security Operational Baseline - EOSC Future Private Space - Wiki EOSC Future

[33] EOSC Security Operational Baseline 2022 | Zenodo

[34] EOSC Interoperability Framework | EOSC Portal (eosc-portal.eu)

[35] https://confluence.egi.eu/display/EOSCOB/EOSC+Portal+Onboarding+Team+-+EPOT

[36] https://wiki.eoscfuture.eu/display/EOSCOB/EPOT+Procedures

[37] https://eosc-portal.eu/providers-documentation/eosc-provider-portal-inclusion-criteria

5.3 Harmonised access policies for service providers

We will start from the AARC policy development toolkit to develop appropriate security policies and rules of engagement, namely Acceptable Use Policies (AUP), security operations, privacy, data, escalation, and assurance. Security policies will need to apply to sites and resource providers and will be developed to be interoperable with collaborating infrastructures (such as EOSC) and its participants and will be compatible with the WISE Security for Collaborating Infrastructures trust framework. We will consult all resource providers and other stakeholders and especially consider new types of providers, such as HPC, who have not previously used AARC policy templates.

5.4 Harmonised access policies for communities

As in the previous section we will also develop access policies for users and research communities. The development will require engagement with some communities and any other relevant stakeholders in such development.

An Acceptable Use Policy and Conditions of Use ("AUP"), for example, defines the rules and conditions that govern access and use (including transmission, processing, and storage of data) of the resources and services ("Services") as granted by community/agency/infrastructure for specific purposes.[38]

There are 10 base clauses listed in the WISE baseline AUP[39], from where we will start development. These are immutable and are important for the creation of any AUP. They can be added to with additional clauses. The base 10 clauses are:

1. You shall only use the Services in a manner consistent with the purposes and limitations described above; you shall show consideration towards other users including by not causing harm to the Services; you have an obligation to collaborate in the resolution of issues arising from your use of the Services.
2. You shall only use the Services for lawful purposes and not breach, attempt to breach, nor circumvent administrative or security controls.
3. You shall respect intellectual property and confidentiality agreements.
4. You shall protect your access credentials (e.g., passwords, private keys, or multi-factor tokens); no intentional sharing is permitted.
5. You shall keep your registered information correct and up to date.

[38] EOSC AUP and use template

[39] WISE Baseline AUP

6. You shall promptly report known or suspected security breaches, credential compromise, or misuse to the security contact stated below; and report any compromised credentials to the relevant issuing authorities.
7. Reliance on the Services shall only be to the extent specified by any applicable service level agreements listed below. Use without such agreements is at your own risk.
8. Your personal data will be processed in accordance with the privacy statements referenced below.
9. Your use of the Services may be restricted or suspended, for administrative, operational, or security reasons, without prior notice and without compensation.
10. If you violate these rules, you may be liable for the consequences, which may include your account being suspended and a report being made to your home organisation or to law enforcement.

6 Conclusions

The architecture of the Digital Twin Engine Infrastructure has been designed and the relationship between main components have been analysed throughout the first 8 months of the project. All the needed components and interaction to effectively build a continuum model have been defined combining the two main concepts described by this document namely Cloud Orchestration and compute offloading. The two allow to build the Compute Federation. Regarding the data, a key for the project, the Data Lake model as developed by ESCAPE has been considered the foundation. Finally, the main challenge for the accounting and policies access have been identified.

The relevant stakeholders involved in the WP5 contributed identifying the technical requirements that are key to the design of the DTE infrastructure. Particularly requirements from resource providers as well as from WP6 have been collected and thoroughly analysed. More in detail requirements concerning the interfaces to exploit the compute capacity have been considered. In addition, requirements concerning data access and storage capability were considered and finally access policies have been discussed. The outcome has been that we need to deal with a highly heterogeneous set of resources and that the heterogeneity applies to several aspects spanning from interface to access and exploit resources up to the actual hardware and system architectures passing from very distinct approaches to the quality of the services and access policies.

From the community perspective the main focus has been on requirements from modules belonging to WP6. These are supposed to be the primary consumers of the WP5 infrastructure. Here we received useful feedback for the initial phase of the architectural definition although we expect more details will come during the upcoming phase of the project when we plan to start pilots and testbeds.

Before designing the architecture, a comprehensive scouting of the tools and services already available has been carried on and successfully identified several services and software's to be enhanced to meet the objectives of the project. Namely we analysed the results and the blueprints of C-Scale, ESCAPE, EGI-ACE and other open source initiatives and we maximise the services to adopt while building the interTwin toolkit.

After describing the overview of the architecture, all the main components described deep into the details the solutions as well as the specific services. All of them highlighted where they build on and what is expected to be enhanced within interTwin.

As a future activity we identified the prototyping step as the most urgent. The vision in this respect is that testbeds will implement the playground that will actually enable the fruitful process of the co-design. In other words, the testbeds are meant, at least in the initial part of the project implementation, the place where all the relevant stakeholders can actively contribute to the process of: testing - gathering feedback - processing requests and implementing the needed features.

7 References

Reference	
No	**Description / Link**
R1	interTwin Deliverable D3.1 “Blueprint architecture, functional specifications and requirements analysis first version” https://documents.egi.eu/document/3930