

interTwin

D5.5 DTE infrastructure development and integration report

Status: Under EC Review
Dissemination Level: public



Funded by the
European Union

Disclaimer: Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them



Abstract

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

This deliverable provides a comprehensive description of the Digital Twin Engine Infrastructure developed during the course of the project. The focus is to report on the services and components implemented to build the federations (data and compute). A report on strategy adopted to integrate heterogeneous providers is also part of the present document aiming at providing a complete description of how WP5 implements the digital "continuum". The latter has been proved between cloud-edge and multi cloud environments, HPC Centers and Quantum. Access policy and resources accounting are also included.

Finally, it provides an overview of the integration with modules developed by other WPs. With this we aim to show how the DTE infrastructure has been adopted by the scientific communities belonging to the project.



Document description			
D5.5 - DTE infrastructure development and integration report			
Work Package number 5			
Document type	Deliverable		
Document status	UNDER EC REVIEW	Version	1.0
Dissemination Level	Public		
Copyright Status	 <p>This material by Parties of the interTwin Consortium is licensed under a Creative Commons Attribution 4.0 International License.</p>		
Lead Partner	INFN		
Document link	https://documents.egi.eu/document/3948		
DOI	https://zenodo.org/records/16567077		
Author(s)	<ul style="list-style-type: none"> • Diego Ciangottini (INFN) • Paul Millar (DESY)  0000-0002-3957-1279 • Liam Atherton (STFC UKRI) • Adrian Coveney (STFC UKRI) • Giacinto Donvito (INFN) • Daniele Spiga (INFN) • Andrea Manzi (EGI Foundation) • Zdenek Sustr (CESNET) • Dijana Vrbanc (DESY) 		
Reviewers	<ul style="list-style-type: none"> • Andrea Manzi (EGI Foundation) • Sandro Fiore (UNITN) 		
Moderated by:	<ul style="list-style-type: none"> • Andrea Anzanello (EGI Foundation) 		
Approved by	Germán Moltó (UPV) on behalf of TCB		

Revision History			
Version	Date	Description	Contributors
V0.1	12/05/2025	Created template	Andrea Manzi(EGI Foundation)
V0.2	30/05/2025	initial text	Daniele Spiga (INFN)
V0.3	8/06/2025	Intro/scope and section 1 completed	Daniele Spiga (INFN)



V0.4	30/06/2025	Section 3	Adrian Coveney (STFC)
V0.5	2/07/2025	Section 6	Giacinto Donvito (INFN)
V0.6	07/07/2025	Section 4	Paul Millar (DESY)
V0.7	08/07/2025	Harmonization and conclusions	Daniele Spiga (INFN)
V0.8	19/07/2025	Version including reviewers comments	Sandro Fiore (UNITN) Andrea Manzi (EGI Foundation)
V0.9	25/07/2025	Version ready for TCB approval	Daniele Spiga (INFN)
V0.91	27/07/2025	Revision by TCB	Germán Moltó (UPV)
V0.92	28/07/2025	Version ready for QA	Daniele Spiga (INFN)
V1.0		Final	

Terminology / Acronyms

Term/Acronym	Definition
CESGA	Centro de Supercomputación de Galicia (Galicia Supercomputing Center)
CVMFS	The CernVM File System provides a scalable, reliable, and low-maintenance software distribution service.
DID	Data Identifier
DTE	The Digital Twin Engine developed by interTwin
DZA	Deutsches Zentrum für Astrophysik (German centre for astrophysics)
EOSC	European Open Science Cloud
FPGA	Field Programmable Gate Arrays
FTS	The File Transfer Service. A software component maintained by a development team at CERN.
HPC	High-Performance Computing
HTC	High-Throughput Computing
ILDG	International Lattice Data Grid
K8s	Kubernetes. Container Orchestration Technology
KBFI	Keemilise ja Bioloogilise Füüsika Instituut (National Institute of Chemical Physics and Biophysics)
ML/AI	Machine learning / artificial intelligence
OIDC	OpenID Connect
POSIX	Portable Operating System Interface
RSE	Rucio Storage Element
Rucio	A third generation data management software component, maintained by the Rucio development team.
STAC	SpatioTemporal Asset Catalogs
VO	Virtual Organization
VPN	Virtual Private Network,

Terminology / Acronyms: <https://confluence.egi.eu/display/EGIG>



Table of Contents

1	Introduction	8
1.1	Scope	8
1.2	Document Structure	8
2	Overview of WP5 Modules	9
2.1	DTE infrastructure integration: strategy and overview	10
3	Task 5.1: Federated compute infrastructure (HTC, HPC, Cloud, Quantum)	13
3.1	interLink	13
3.2	Functionalities developed since the last release	14
3.3	Integrations with other DTE components	14
3.4	Integration with Sites	15
3.5	Pilots and testing activities	16
4	Task 5.2 Federated data infrastructure	20
4.1.1	Teapot	20
4.1.2	ALISE	21
4.1.3	FTS3	22
4.1.4	Rucio	23
4.1.5	Rucio JupyterLab extension	24
4.1.6	Onedata S3	24
4.2	Functionalities developed since the last release	25
4.2.1	Teapot	26
4.2.2	ALISE	26
4.2.3	Rucio	26
4.3	Integrations with other DTE components	27
4.4	Integration with Sites	28
4.4.1	Storage integration	29
4.4.2	Certificate Authorities	30
4.5	Pilots and testing activities	31
4.5.1	Development DataLake	31
4.5.2	Virgo DataLake	31
4.5.3	LatticeQCD DataLake	32
4.5.4	STAC Integration Pilot	32
5	Task 5.3 Federation Services and Policies	34
5.1	APEL Accounting	34
5.2	Kubernetes Usage Accounting Probe	35
5.3	Functionalities developed since the last release	35
5.4	Integrations with sites and other DTE components	36
5.5	Pilots and testing activities	37
6	Task 5.4 AI-based orchestrator	38
6.1	AI-based orchestrator	38
6.2	Functionalities developed since the last release	39
6.3	Integrations with other DTE components	41
6.4	Integration with Sites	41
6.5	Pilots and testing activities	41
7	Conclusions	42
8	References	43



List of Figures

Figure 1 The figure shows the interTwin topology integrated using the WP5 toolbox	12
Figure 2 The picture shows a high-level view of the itwinAI workflow and its integration with interLink	16
Figure 3 The JupyterHub instance of interTwin where the available integrations are made available to the users.	18
Figure 4 The plot shows the number of concurrent GPUs at Vega have been used by Ray running at UKRI, through the interLink based offloading.	19
Figure 5 Overview of the DataLake components on the map of Europe.....	29
Figure 6 interTwin Accounting Dashboard, with development and production data	37

List of Tables

Table 1 The table provides a compact view of the modules developed and enhanced by WP5....	10
--	----



Executive summary

The deliverable D5.5 has two main objectives. On the one hand, it aims to provide a comprehensive report on the Digital Twins Engine infrastructure development carried out during the project, and on the other hand, it thoroughly describes how the WP5-developed services have been successfully integrated.

Regarding the developed infrastructure the latest updates will be described including both new functionalities as well as enhancements and consolidations carried out to effectively deliver a continuum implementation. The latter involves computing, data and storage.

Concerning the integration activities, the twofold goal of the D5.5 is to describe the results achieved with sites (the providers) with the software modules developed by interTwin, principally with core modules and thematic modules developed by WP6/WP7 respectively. It's worth highlighting that the integration with resource providers represents a peculiarity of the work package activities. Thanks to their technological heterogeneity they had a strategic role in the overall workplan implementation, offering different interfaces and implementing distinct policies they represented a unique opportunity to build a real playground where to discuss first and implement then, operationally suitable and maintainable solutions.

The document has been composed by experts, with a deep knowledge of the adopted services and used technologies, that contributed both to the initial design [R1] and to the implementation of the pillars of the DTE infrastructure.

1 Introduction

1.1 Scope

The primary scope of this deliverable is to provide an overview of the development activities carried out to successfully build the Digital Twin Engine infrastructure. This will include how the various modules and subsystems have been harmonically integrated in order to build the actual middleware used by interTwin to enable scientific communities belonging to the project to exploit the continuum between heterogeneous providers (Cloud, HPC and HTC).

The toolkit developed by WP5 can be grouped in 4 distinct domains each one coping with specific aim:

- To enable the compute continuum seamlessly integrating Cloud and world-class HPC and HTC;
- To abstract storage infrastructure complexity enabling a DataLake model offering data management capabilities even on world-class HPCs;
- To federate and orchestrate Clouds to deploy high-level services and instantiate end-user services from WP6 on the most suitable cloud infrastructure;
- To provide effective resource usage accounting Including specialized HW and resources provisioned via container orchestrators.

In addition, the document will also describe the results of the services integration process, an asset of WP5. The vision has been to implement a “continuous feedback model” as a strategy to codesign the DTE infrastructure architecture. In order to effectively implement such a model we proposed to build testbeds that, on the one side, act as playgrounds for scientific communities and, on the other side, make it possible to directly interact with providers. This was of fundamental importance to gather feedback from systems admins, a key to prove the long-term sustainability of the technological choices made by interTwin. From the end user perspective the testbed is the place where to create pilots, the place where to integrate the interTwin thematic and core modules with WP5 connectors.

1.2 Document Structure

This document describes two main areas, the technical development of the architecture for the Digital Twin Engine infrastructure, and the process that each module as well as the WP5 as a whole followed to integrate services, modules and computing resources.

It is organised in five main sections starting with a general overview of the modules including a high-level description of the integration strategy. Then it is followed by four sections, one per each task, where every single component is presented in terms of functionalities, integration with sites and with external (to the WP5) modules.

Finally, the last section draws the conclusions about the architecture developed and summarises the overall experiences achieved operating the DTE infrastructure through the testbeds.

2 Overview of WP5 Modules

WP5 developed a series of composable modules in order to create a toolbox providing all the fundamental elements to satisfy a wide range of use cases. The underlying idea is that no single solution fits all the possible use cases. Therefore, the strategy was to prepare a set of modules that can be integrated with each other depending on the specific needs of each scientific community. The ultimate goal is to enable transparent access to heterogeneous providers of resources. More in detail, heterogeneous in this context means the mixture of Cloud, HPC and HTC. Quantum is also part of the puzzle and it is successfully integrated through the WP5 toolbox even if at a different level of maturity.

The result obtained is shown in **Table 1**. The toolbox is composed of seven major modules. Three of them (interLink, Teapot and ALISE) are brand new developments carried out by WP5 while others are inherited from previous EU-funded projects such as C-SCALE¹, ESCAPE², EGI-ACE³ and enhanced in order to cope with the specific needs of the interTwin communities.

The strategy to build some of the modules on top of existing solutions has been strategic as it has given a lot of boosts to the activities and has allowed to invest a sizable amount of effort to adapt tools to the interTwin requirements, and thus to evolve the software in order to support additional use cases.

It is important to highlight that several other components, satellites of the main modules, have been developed and integrated. In particular few notable examples are:

- CVMFS⁴ has been integrated and partially used to manage software distribution (including container image distribution) in the heterogeneous set of providers;
- Grafana and Grafana Tempo for specific service monitoring;
- Rucio JupyterLab⁵ extension that integrates with Rucio - Scientific Data Management to allow users to access some of Rucio's capabilities directly from the JupyterLab interface.

¹ <https://c-scale.eu/>

² <https://projectescape.eu/>

³ <https://www.egi.eu/project/egi-ace/>








⁴ <https://cernvm.cern.ch/fs/>

⁵ <https://vre-hub.github.io/docs/extensions/rucio-jupyterlab/>



D5.5 DTE infrastructure development and integration report

Table 1 The table provides a compact view of the modules developed and enhanced by WP5, including a short description, the task responsible for developing and operating the module and finally highlighting the key feature each one brings to the DTE infrastructure

Module name	Short Description	Task	Key Feature in the context of DTE infrastructure
 interLink	An open-source service to support heterogeneous providers federation	5.1	Transparent and dynamic extension of K8s clusters over HPC providers
 Teapot	This application provides a WebDAV that supports multi-tenancy	5.2	Plug & Play solution to federate HPC provided storage within the DataLake
 RUCIO	Rucio provides services to manage large volumes of data spread across facilities	5.2	To orchestrate heterogeneous storage providers to manage any type of scientific data
 FTS3	The File Transfer Service (FTS) is responsible for moving sets of files from one site to another	5.2	To integrate with Rucio to perform third party copies within storage of the DataLake
 ALISE	An open-source service for linking user accounts	5.2	To enable HPC resource access using federated identity
 APEL Accounting	APEL is an accounting tool that collects accounting data from sites	5.3	To gather, collect and manage K8s provisioned resources
 AI Based Orchestrator	An open-source service to instantiate resources on Cloud Management Frameworks	5.4	To automate high level services deployment selecting the most suitable cloud provider

2.1 DTE infrastructure integration: strategy and overview

The interTwin resource providers are organisations that primarily contribute with computing resources supporting the scientific use cases of the project. In other words,



the providers need to grant not only access to the computing capacity but, to some extent, need to support data access, transfer, archiving capability that a Digital Twin might require.

Since the access to compute and data should happen through the DTE Infrastructure, we considered it key that the resource providers also participate in the definition and the validation of the DTE Infrastructure architecture. The idea to develop dedicated integrations evolved in the so-called Testbeds, created to allow all the sites to participate in the definition and mostly to validate various modules. This process allowed us not only to deliver a generic and extensible architecture for the DTE Infrastructure, providing stable interfaces to the storage systems and the distributed data sources, avoiding an integration tailored to a single site configuration, but also to validate with site owners the sustainability, from a facility perspective, of the technical choices.

WP5 includes 8 sites exposing cloud interfaces, HPC standard access and HTC interfaces meant as grid interfaces in details:

- Cloud: UKRI, GRNET, EODC;
- HPC: PSNC, VEGA, Juelich, Vilnius;
- HTC: KBFJ.

In addition, CESGA (Centro de Supercomputación de Galicia), even if not part of the consortium but in-kind contributing, is being integrated in the interTwin ecosystem with the main purpose to test WP5 toolbox to exploit a Quantum system. The main goal of this integration is to test the interTwin developed technology, in particular the compute integration of the DTE Infrastructure. At the time of writing there is no use case selected in our portfolio of scientific communities for the exploitation of such a machine. From a technical perspective the Quantum machine will be accessible through slurm batch system via a dedicated bastion node. However it's worth mentioning that we foresee a future activity, beyond the time scale of the project, to further evolve the current integration.

Finally, to compose the whole picture, there are additional providers that took part in the final integration such as DESY, Cyfronet and INFN-CNAF, also in this case computing capacity was not part of the project, but sites participated providing an in-kind contribution.

The importance and the key role of the testbeds in the development of WP5 program of work has been extensively discussed not only in the previous sections but also in previous deliverables where technical design was thoroughly described and motivated. We started building several testbeds, where initially we worked in silos mode, taking compute and data activities separated to then started integrating all the components as much as possible.

In particular, the vision is to use cloud providers as hosts for the high-level services deployment, being the hubs or gateways for the end users. This means that modules developed in WP6 are mainly expected to run over cloud, as in most cases all of them follow a cloud-native approach. Deployment of services is intended to be orchestrated and automated via INDIGO PaaS Orchestrator and Infrastructure Manager (IM) (further details are in [Section 5](#)).

Thanks to interLink, developed in Task 5.1, any service running on any cloud on top of a kubernetes (K8s) cluster is enabled to transparently offload payload execution (i.e. pods in K8s jargon) to HPC and HTC resource providers. This grants a transparent and dynamic



access to a sizable amount of computing capacity to scale out users' computations. Not only scale, as a matter of fact, it represents an ideal model to get access to specialised hardware that does not necessarily exist within the local cloud provider, where the high-level service is hosted. This represents how we extend cloud-native applications toward classical HPC machines such as those in the EuroHPC Joint Undertaking and how we implemented the compute continuum in interTwin (see [Section 2](#)).

An essential aspect to make the described flow effective is transparent access to data. To this end the services developed in Task 5.2 are meant to enable the interTwin DataLake. The latter has the objective to enable the possibility to transparently and automatically transfer any data available in any repository to all the sites belonging to the DataLake. In principle, this applies also to external sources of data, provided a data injection step is performed. Also, the system is enabled to support any type of data (see [Section 3](#)).

Finally, the resource accounting has been extended to K8s based systems and this has proved to enable metrics gathering in such a heterogeneous environment. A pictorial high-level view of the obtained result is shown in [Figure 1](#), where the actual interTwin topology is represented.

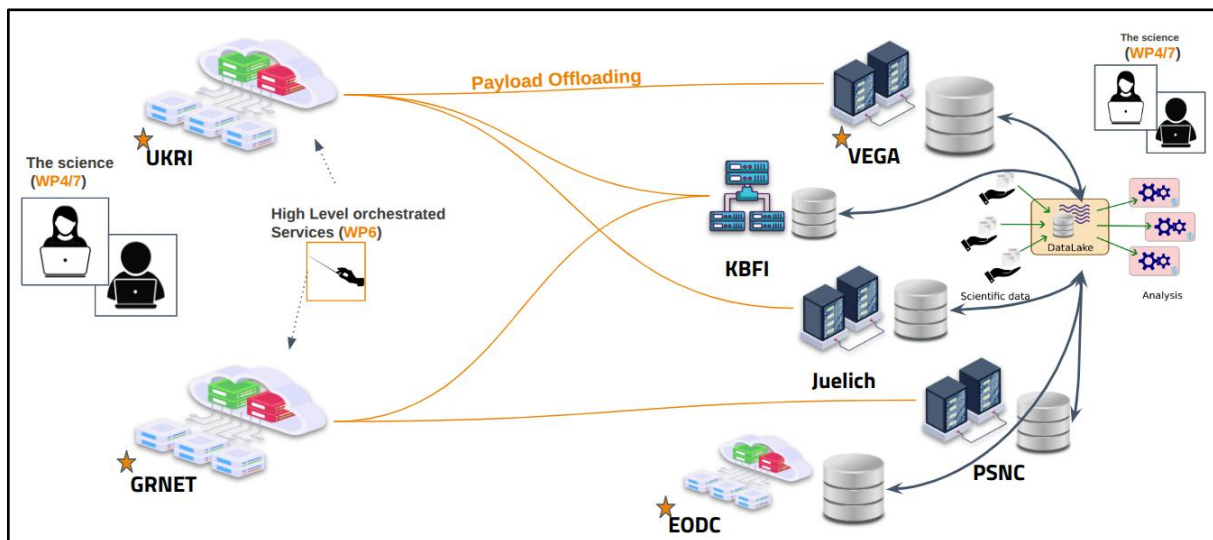



Figure 1 The figure shows the interTwin topology integrated using the WP5 toolbox. Starting from the left the cloud systems represent the entrypoint for the user to deploy high-level services. Deployed services can transparently scale out on HPC/HTC (orange connectors). Finally, on the right the data service layer represents the entrypoint to the user for managing the scientific input/output. The picture includes only sites supported by the project and not the externals (early adopters).

3 Task 5.1: Federated compute infrastructure (HTC, HPC, Cloud, Quantum)

The main goal of T5.1 is to provide software solutions to enable resource provisioning on a wide range of technologically heterogeneous compute providers. It delivers an architecture based on edge services to enable federation and to provide all the knobs to deal with customised runtime environments to accommodate the needs of scientific communities.

3.1 interLink

Component name and logo	<div>interLink</div> 
Page on interTwin website	https://www.intertwin.eu/article/infrastructure-component-interlink
Description	Federate heterogeneous providers compute resources enabling a continuum between distinct providers.
Value proposition	A unified set of APIs to integrate resources provided by providers using different technologies and architectures. Transparent and dynamic extension of K8s clusters over any provider or computing capacity, including EuroHPC providers, possibly geographically distributed.
Users of the Component	Any application/service using K8s as resource provisioning system
User Documentation	https://interlink-hq.github.io/interLink/
Technical Documentation	https://interlink-hq.github.io/interLink/docs/Developers
Responsible	INFN
License	Apache 2.0
Source code	https://github.com/interlink-hq/interLink
Language	Go, Python



3.2 Functionalities developed since the last release

Functionality-wise, the system was almost complete at the time of the second release. Since then, no major features have been added. However, early evaluations and feasibility studies for future evolution have been successfully carried out. In particular:

- using the offloading via interLink against non-x86_64 architecture. In particular, the evaluations are done targeting FPGA in view of future extensions toward low-power architectures;
- extending the overall system with the possibility to deploy VPN open source protocol at the user level in order to enable pod to “virtual” pod network connection with the main purpose of enabling process management communication between services and processes running through offloading. This is a key for future evolution in supporting ML/AI frameworks and to exploit the full HPC capabilities (MPI-based processing);
- integrating ALISE-based accounting mapping.

More effective monitoring capabilities have been added in order to allow an effective service operation.

Rather than that, during the last period of the project the focus has been on the implementation of extensive scale tests (see [Section 3.5](#)).

3.3 Integrations with other DTE components

WP5 modules can be seen as connectors for DTE modules (both core and thematic ones) to the underlying computing capacity (the resources providers). Several integration activities have been performed in order to put together components developed mainly in WP6 [\[R2\]](#) with interLink.

The most important ones are the integration with the following frameworks and modules:

- itwinAI [\[R3\]](#): a Python toolkit that simplifies AI workflows for scientific digital twins, offering distributed training, HPC optimization, and integrated ML logging;
- OSCAR [\[R4-R5\]](#): is an open-source platform designed to support the event-driven serverless computing model for handling data-processing intensive computations on elastic Kubernetes clusters;
- JupyterHub⁶: JupyterHub is a multi-user gateway that launches and manages isolated Jupyter notebook servers so each person gets their web-based workspace;
- KubeRay⁷: cluster: Python-native API that unifies data processing, distributed training, hyper-parameter tuning and inference.

All of them have been functional to allow some of the thematic modules to exploit heterogeneous resources. In addition, it is important to highlight that all of them were successfully capable of demonstrating how the proposed offloading model is powerful and promising for future evolutions.

Finally, additional integrations happened during the past months even if not in the context of interTwin but within projects highly synergic, notably:

⁶ <https://jupyter.org/hub>

⁷ <https://github.com/ray-project/kuberay>



- the KubeDask integration to support the Italian National Research Centre for High Performance Computing, Big Data and Quantum Computing (ICSC) use cases for high-rate data analysis in the context of the High Energy Physics domain;
- Argo workflows pipelines were set to offload part of the tasks to the HPC centers serving LISA interferometer, in the context of the design of their computing model;
- GenAI application development has been demonstrated working with the integration of the HelixML framework and the offloading on VEGA HPC supercomputer, leading to the publication of a success story of the project⁸.

3.4 Integration with Sites

Following the strategy detailed in **Section 1** interLink has been integrated with the whole set of sites identified as the most suitable target providers to enable the scale-out capability through the offloading mechanism.

The model, in a nutshell, is that K8s cluster runs on clouds and interlink makes it possible to extend resources with almost zero extra requirements for the end users. In the context of interTwin we experienced several patterns and topologies. In particular, we rely on cloud at UK Research and Innovation (UKRI) and INFN in-kind cloud resources provided via CNAF and at GRNET. While the latter has been functional to speed up the initial bootstrap phase of the activities, the first is what we consider the hub of interTwin for such a compute continuum exploitation. It is important to note that nothing prevents you from using any other provider, given that interLink is completely site agnostic. According to all the interLink architecture the K8s clusters deployed in the Cloud sites host a virtual kubelet. On the other side of the architecture, there are interLink APIs and plugins deployed close to the target HPC providers.

Several targets have been successfully integrated. Namely:

- **IZUM**: provides access to VEGA hosted at Maribor in Slovenia, the first of eight peta and pre-exa-scale EuroHPC. With 960 CPU nodes (overall 1920 CPUs AMD Epyc 7H12 – 122000 cores) and 60 GPU nodes (overall 240 GPUs Nvidia A100) had a primary role in the testing, validation and fine-tuning of the interLink proposed model (see **Section 3.5**). To integrate VEGA we used the interLink in-edge deployment mode. In this scenario, the Virtual Kubelet communicates with remote services deployed on a dedicated edge node, exposing authenticated interLink APIs and its associated plugin. This setup is ideal for scenarios where services at the edge of the HPC machine are utilized for controlled communication between the Kubernetes cluster and the remote resources. In this integration, the SLURM plugin is used.
- **Juelich**: JSC provides seamlessly integrated cloud and HPC resources through UNICORE middleware using interLink. The flexibility of the interLink API layer, allowed to transform pod creation requests into HPC jobs for the powerful JUWELS system using a dedicated plugin. The integration leverages the already available UI at the site for managing container execution on the HPC cluster.

⁸ <https://www.intertwin.eu/case-study/success-story-running-genai-on-supercomputers-thanks-to-interlink>



- **PSNC:** provides access to a world-class e-Infrastructure for the scientific community, a specific research and development environment. In terms of plugin this integration uses the SLURM one. With respect to VEGA this deployment follows another path which is named tunneled. In this scenario the Virtual Kubelet connects to a remote interLink API server and its plugin through a secure tunnel. This setup ensures secure communication between the Kubernetes cluster and the remote resources, making it suitable for environments with strict security requirements or to host services on a multi-user host like a login node.
- **KBFI:** The cluster consists of around 8000 compute cores and a distributed storage facility with 3.8 PB of raw disk capacity. All machines are connected to a 10Gbit local network. The primary user of the cluster is the Compact Muon Solenoid (CMS) collaboration through the Worldwide LHC Computing Grid (WLCG) infrastructure. The peculiarity of such integration was to follow a quite different pattern where interLink interacts with compute resources via ARC-CE Compute model. Another plugin is used, the ARC-CE plugin, and this proves yet again the flexibility of the system.

As previously discussed, such heterogeneity of technologies provided by these involved sites was a key value for the project and successfully contributed first to develop a generic, agnostic model and then to actually prove it with real integrations. Also, different policies for resource access and network management represented a very important aspect in order to test different deployment models supported by interLink.

Finally, to be noted that synergies with ICSC allowed for the integration with CINECA Leonardo both booster and general-purpose partition.

3.5 Pilots and testing activities

interLink has been used in several pilot activities of the interTwin project. In particular, all the pilots that exploited the integrations mentioned in [Section 3.3](#).

In particular its integration with itwinAI enabled the transparent access to HPC: by simply modifying YAML configuration files, both training and Hyperparameter Optimization (HPO) can run transparently on local or remote infrastructure via interLink as shown in [Figure 2](#).

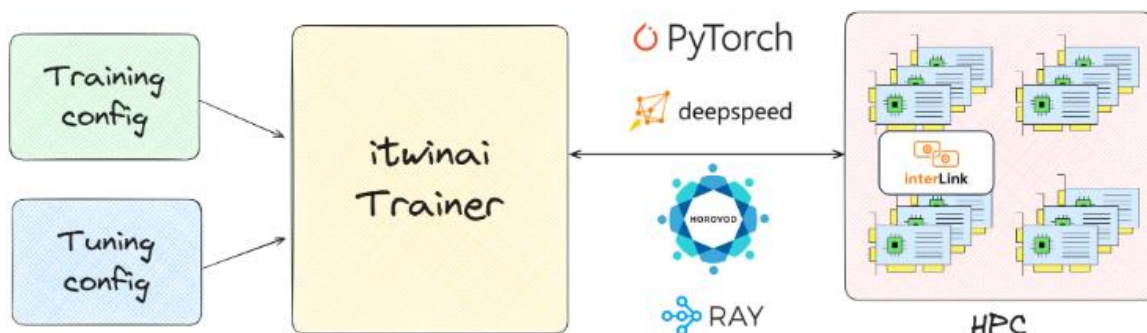


Figure 2 The picture shows a high-level view of the itwinAI workflow and its integration with interLink

In more detail, there are two YAML files, one for the training run, one for the HPO search space, that represents the inputs of the itwinAI Trainer. The itwinAI Trainer blackbox auto-

selects the ML backend (i.e PyTorch⁹, DeepSpeed¹⁰, Horovod¹¹, Ray, ...) and then a pod is executed as a job launched on the HPC cluster exploiting GPUs via interLink.

Another area where interLink significantly contributed to the pilots is the one that relates to the JupyterHub. In this case, interLink enabled the possibility to execute interactive processing on HPCs. A central JupyterHub running on a K8s cluster at UKRI has been enhanced via interLink to support multiple choices to launch notebooks. In detail, as shown in **Figure 3**. The single hub has been equipped with the capability to support three main patterns:

- Notebook spawned on local standard Kubernetes provisioned node;
- Notebook spawned, through interLink, on remote VEGA HPC system with one or more Nvidia A100 GPUs;
- Notebook spawned, through interLink on the PSNC HPC system with one or more Nvidia H100 GPUs;
- Notebook spawned, through interLink on the KBFI HTC system.

All this represents a concrete working example of how we successfully managed, thanks to interLink, to make transparent to the end user the access to the 3 different provisioning models: Cloud, HPC and HTC. As explained below this integration pattern is framework agnostic and Jupyter has been selected as a major requirement for the DT Application developers.

⁹ <https://pytorch.org/>

¹⁰ <https://www.deepspeed.ai/>

¹¹ <https://horovod.ai/>



Server Options

UKRI Cloud

itwinai environment on UKRI Cloud

CPU and Memory:

GPU:

Image:

Other image:

Offload to Vega HPC - dev VO

Offload Jupyterlab session on a GPU node at Vega supercomputer

CPU and Memory:

Partition and GPUs:

Image:

Other image:

Offload to PSNC HPC - dev VO

Offload Jupyterlab session on a node at PSNC supercomputer

CPU and Memory:

Partition and GPUs:

Image:

Other image:

Offload to KBFI

Offload Jupyterlab session on a KBFI

CPU and Memory:

Image:

Other image:

Start

Figure 3 The JupyterHub instance of interTwin where the available integrations are made available to the users. The four described options are present and offered to the users.

Finally, the Ray integration was recently added and used mainly for the first round of scale test in addition to perform evaluations for future activities. Despite none of the supported communities within interTwin are directly using Ray, we used such integration on Vega HPC, where we showed that anyone using a Kubernetes-native framework like Ray can spin up Ray clusters on an HPC system with only minimal tweaks to the RayKube Cluster configuration.

A first round of tests successfully implemented with Ray running at UKRI and offloading toward VEGA as shown in **Figure 4** we can manage multiple GPUs in parallel and thus how we can use interLink to scale out satisfying specific needs.

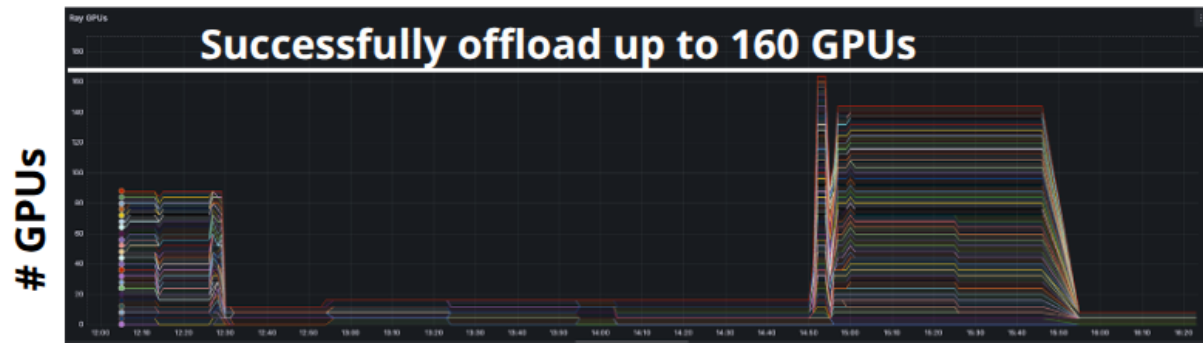



Figure 4 The plot shows the number of concurrent GPUs at Vega have been used by Ray running at UKRI, through the interLink based offloading.

4 Task 5.2 Federated data infrastructure

Task 5.2 has the main objective to support data requirements of Digital Twins providing data (files) needed to build digital twins, as needed by the scientific communities. In addition, it's responsible for dealing with orchestrated data transfer both for input and produced output data (files).

4.1.1 Teapot

Component name and logo	Teapot 
Page on interTwin website	https://www.intertwin.eu/article/infrastructure-component-Teapot
Description	<p>Teapot is an easy-to-install edge service that a facility may deploy to provide remote access to its storage, in order to facilitate data ingress and egress. It provides the reference implementation of the Teapot architecture: a scalable and technology-agnostic approach to support data ingress and egress.</p> <p>FTS (File Transfer Service) uses Teapot to copy files from some external source into the facility's file storage, or to copy data from the facility's storage to some external destination. Rucio uses Teapot to manage replicas at the facility: creating new replicas with FTS or deleting cached data when the storage is under space pressure.</p>
Value proposition	Unlike existing storage solutions that are compatible with the interTwin DataLake, Teapot does not require exclusive control of the underlying storage. Instead, it supports existing access methods. This allows facility admins to deploy Teapot on top of existing storage solutions without disrupting established modes of access.
Users (i.e., site admins) of the Component	Deployed: CESGA, KBFI In progress: DZA, University of Vilnius
User Documentation	https://intertwin-eu.github.io/Teapot/ https://github.com/intertwin-eu/Teapot/blob/main/CONFIGURATION.md
Technical Documentation	https://intertwin-eu.github.io/Teapot/developer/

Responsible	Deutsches Elektronen-Synchrotron (DESY).
Licence	Apache v2
Source code	https://github.com/interTwin-eu/Teapot
Language	Python, Java

4.1.2 ALISE

Component name and logo	ALISE
Page on interTwin website	https://www.intertwin.eu/article/infrastructure-component-alise
Description	Account Linking Service (ALISE) is a tool for linking a user's federated identity with their facility account. ALISE provides an automated procedure for users of a facility to register their federated identity.
Value proposition	<p>Most facilities have some account identity and access management (IAM) system. Among other things, this component is responsible for handling supported authentication, with typical features allowing passwords to be changed, handling forgotten passwords, and registering SSH public keys.</p> <p>Currently, most facilities have no support for OIDC (token-based) authentication. Therefore, their IAM solutions typically do not allow a user to register their OIDC identity. ALISE is an easy-to-deploy standalone service. By allowing users to register their OIDC identity, an ALISE instance allows sites to deploy other services that require OIDC/token-based authentication, and for those other services to identify users by their federated identity.</p> <p>The process to register a user's OIDC identity is needed only once per user. It requires no admin intervention.</p>
Users (i.e., site admins) of the Component	PSNC, KBFI, Vega, CESGA
User Documentation	N/A
Technical Documentation	https://github.com/m-team-kit/alise/blob/master/README.md



Responsible	Karlsruhe Institute of Technology (KIT).
Licence	MIT
Source code	https://github.com/m-team-kit/alise/
Language	Python

4.1.3 FTS3

Component name	FTS3
Description	FTS3 is the software underlying a service responsible for globally distributing the majority of the LHC data across the WLCG infrastructure. It is a low-level data movement service, responsible for reliable bulk transfer of files from one site to another while allowing participating sites to control the network resource usage.
Value proposition	Transferring large volumes of data between facilities requires a component that manages those transfers: monitoring their progress, cancelling transfers that have stalled or that take too long, and retrying failed transfers (where appropriate). Deploying FTS provides a common service for handling such transfers, allowing higher-level data management; e.g., bandwidth shaping links between facilities.
Users of the Component	FTS is a core component that most users do not access directly. Instead, FTS is used by any DT user or manager who is (directly or otherwise) managing data locality. FTS is used when a DT user or manager needs some data to be located at some facility and that data is not currently available.
User Documentation	https://fts3-docs.web.cern.ch/fts3-docs/docs/cli.html
Technical Documentation	https://fts3-docs.web.cern.ch/fts3-docs/docs/docs/developers.html
Responsible	CERN
Licence	Apache 2.0
Source code	https://gitlab.cern.ch/fts/fts3



4.1.4 Rucio

Component name	Rucio
Description	<p>Built on more than a decade of experience, Rucio serves the data needs of modern scientific experiments.</p> <p>Large amounts of data, countless numbers of files, heterogeneous storage systems, globally distributed data centres, monitoring, and analytics. All coming together in a modular solution to fit your needs.</p> <p>Rucio provides a service that manages data locality. It provides a scalable solution for managing the dynamic locality of files in a heterogeneous, federated storage DataLake.</p> <p>Rucio will create new replicas when it identifies that data locality requirements are not being satisfied. It uses an FTS3 service to create those replicas by issuing requests and monitoring their progress.</p>
Value proposition	<p>When deployed, the Rucio software provides a service that allows a group of researchers to manage non-trivial amounts of data. For any given file, dataset, collection of files, or container (a collection of files and datasets), it provides information on where that data is currently available.</p> <p>It also supports dynamic, time-limited data placement, with data being made available for some period (e.g., to support some computational workflow).</p> <p>It optimises the use of available storage by operating a cache, assuming that data that was previously used is more likely to be needed in the future.</p> <p>Desired data locality is expressed in terms of declarative rules. These rules may be applied both to existing datasets and anticipated, future data.</p>
Users of the Component	In principle, all DT users that use the DataLake concept to manage their data are using Rucio. Depending on the use cases, DT users may interact directly with Rucio, or they may use Rucio via some intermediate service.
User Documentation	https://rucio.cern.ch/documentation/
Technical Documentation	https://rucio.cern.ch/documentation/
Responsible	CERN
Licence	Apache 2.0



Source code	https://github.com/rucio/rucio
-------------	---

4.1.5 Rucio JupyterLab extension

Component name	Rucio Jupyterlab extension
Description	The Rucio extension is a jupyterlab add-on that acts as an interactive user client, intended to ease data accessibility and data discoverability. It allows using certain Rucio capabilities like interactive data browsing and triggering data replicas to the storage volume attached to the jupyter server.
Value proposition	The extension acts as an interface to the federated data infrastructure, aiming to simplify the interaction with the Rucio instance. Once a Jupyter session is spawned, the extension will be available on the left sidebar. The configuration to interact with the interTwin Rucio instance is executed in the background during the spawning of the session, allowing DT users to interact immediately with the Datalake.
Users of the Component	Any DT user spawning a Jupyter session on any remote resource compatible configured on the federated compute layer. The Rucio extension is compatible with the interTwin federated compute software stack (interLink), and therefore it can be installed on any jupyter-based container.
User Documentation	https://rucio.cern.ch/documentation/started/release-policy/#jupyterlab-extension
Technical Documentation	https://github.com/rucio/jupyterlab-extension/blob/master/CONFIGURATION.md
Responsible	CERN
Licence	Apache 2.0
Source code	https://github.com/rucio/jupyterlab-extension

4.1.6 Onedata S3

Component name	Onedata S3
----------------	------------

Description	Onedata is a high-performance, distributed data management system designed for global infrastructures. It provides seamless access to heterogeneous storage resources and supports diverse use cases ranging from personal data management to large-scale scientific computations. Leveraging a fully distributed architecture, Onedata facilitates the creation of hybrid cloud environments that integrate private and public cloud resources.
Value proposition	<p>When deployed, the system enables users to collaborate, share, and publish data while supporting high-performance computations on distributed datasets via various interfaces, including POSIX-compliant native mounts, pyfs (Python filesystem) plugins, REST/CDMI APIs, and an S3 protocol (currently in beta).</p> <p>It supports on-the-fly replication, data pre-staging, data indexing, data caching and time-dependent cache cleanup. By supporting multiple types of storage backends, such as POSIX, S3, Ceph and OpenStack Swift, Onedata can serve as a unified virtual file system for multi-cloud environments.</p> <p>The rich API enables integration with existing systems and creation of complex data pipelines serving distributed workloads.</p>
Users of the Component	In the context of the interTwin Datalake, all users who need access to datasets available via Onedata, can access them via the Onedata S3 which is in turn used by Rucio.
User Documentation	https://onedata.org/#/home/documentation
Technical Documentation	https://onedata.org/#/home/documentation
Responsible	ACK Cyfronet AGH
Licence	Apache 2.0
Source code	https://github.com/onedata

4.2 Functionalities developed since the last release

Since the last release, the development activities have focused mainly on Teapot and ALISE. There have also been some minor contributions to Rucio.

4.2.1 Teapot

The development activity within Teapot has been driven by the process of its adoption at various facilities. The code development activity has focused on increasing the variety of supported identity mapping schemes.

With the recent development, Teapot now supports three methods for identity mapping:

- **File-based mapping.** This is where the mapping between a user's federated and site-local identities are stored in a simple file.
- **ALISE-based mapping.** This is where Teapot uses an ALISE service instance to obtain the user's site-local identity.
- **VO-based mapping.** This is where all members of a Virtual Organization (VO) are mapped to a single local identity. VO membership is represented as a top-level group, assuming the AARC Guidelines AARC-G002 / AARC-G069.

These identity mapping schemes are now available as component-specific releases.

4.2.2 ALISE

Since the last interTwin release, ALISE has been successfully deployed at PSNC. This process highlighted concerns from the deployment team, which were addressed by enhancing the ALISE codebase. The main concern was related to how requests to map an identity are authorised. Such requests have long required the requesting service to have an API key, which requires an out-of-band process: the service should not attempt to acquire the API key itself. Previously, any user with a valid federated identity could obtain an ALISE API key. The PSNC admins felt this policy was too liberal and requested that ALISE implement a more restrictive policy. After some discussion, a suitable policy was agreed upon: only users who are members of a specific group (for example, identifying them as site-local admins) are authorised to request an ALISE API key. As before, once generated, the ALISE API key must be manually copied into the service, to authorise that service's use of ALISE.

4.2.3 Rucio

A number of bugs and feature requests have been reported to the Rucio development team. These are problems identified from interTwin project's decision to pivot away from using the external `gfal`¹² library for data uploads toward the Rucio Python-native client. Some of these bugs have been fixed within the interTwin project while others are seeing a more community-driven approach, where interTwin and other Rucio user communities are cooperating in solving the problem.

One use-case involves ingesting data into the DataLake that is formatted using Zarr¹³. Zarr format uses a strategy where the data is stored in multiple files under a common root directory. The root directory is freely chosen by the application, but the relative paths of the files (beneath the common root directory) are stipulated by the Zarr specification. This is possible if the files of Zarr data are identified with Rucio DIDs where the name contains multiple path elements. However, this did not work with the `rucio upload` command. Investigating this problem identified a limitation in how DIDs are encoded in

¹² <https://dmc-docs.web.cern.ch/dmc-docs/gfal2/gfal2.html>

¹³ <https://zarr.dev>



the Rucio REST API for the register command. Work continues in collaboration with the Rucio team to devise a solution to this problem. A workaround has been devised, where data is copied into a Rucio Storage Elements (RSE), at the correct location manually and subsequently registered in Rucio.

4.3 Integrations with other DTE components

DataLake integration with other DTE components is designed to ensure that compute-related components can access the DataLake and the data stored within it.

Integration with interLink is achieved by enabling offloaded jobs to access the data on the site-local storage registered as an RSE in the DataLake while running on the site's compute nodes. At Vega and PSNC, this was done by mounting local dCache¹⁴ storage directly onto the compute nodes. At other sites, Teapot was installed to provide access to the HPC storage, allowing data to be written directly to storage already accessible from the compute nodes.

Integration with JupyterHub was implemented using the Rucio JupyterLab extension, which enables users to interact with data on the local RSE directly from within JupyterHub. Additionally, Rucio, the data management software governing the DataLake, provides a FILE protocol for RSEs at HPC/HTC centers, which offers file system paths to data that are directly accessible from the compute nodes. This allows users to locate and access local data replicas efficiently within the HPC environment.

In collaboration with other interTwin development teams, we have also achieved integration with OSCAR¹⁵, an open-source, serverless platform for event-driven computing. From the user's perspective, the user can upload input data for their analysis into the DataLake, if the files are not already present. The user can then create a dataset that provides the input for the desired analysis workflow. The files of that dataset are the input data for the analysis, while the dataset metadata (as stored in Rucio) identifies the dataset as input for some workflow and provides the necessary workflow arguments and options to drive the analysis. Once the dataset is correctly configured, it is marked "closed" in Rucio. This makes the dataset immutable and triggers the OSCAR platform to process the data.

To achieve this integration, the OSCAR platform development team extended the functionality of OSCAR to support Rucio and the DataLake. This Rucio-specific functionality is described more fully in the WP6 deliverable D6.5 "Core development and integration on DTs report".

On the DataLake side, this integration was supported by updating the interTwin Rucio service to include an event broker service, using RabbitMQ. Through this service, authorised clients may subscribe to Rucio data lifecycle events. Using this service, the interTwin OSCAR service becomes aware of datasets being marked "closed" and can identify which of these datasets corresponds to an analysis request.

This functionality has been integrated with a use case where a flood map generated by TU-Wien is used as input for a Deltares-developed workflow, as executed by OSCAR. For further details, see task T4.6 "Flood early warning in coastal and inland regions" from WP4

¹⁴ <https://dcache.org/>

¹⁵ <https://oscar.grycap.net>



and the upcoming deliverable D4.7 (Final version of the DTs capabilities for climate change and impact decision support tools including validation reports)

4.4 Integration with Sites

Throughout the project, site integration has focused on leveraging existing storage technologies to enable their inclusion in the DataLake, as well as extending or developing new solutions - such as Teapot and ALISE - where existing systems were incompatible. This process involved overcoming challenges related to diverse storage backends and protocols.

Data transfers between RSEs were handled automatically by Rucio, which leveraged FTS to orchestrate direct site-to-site transfers. To enable this, trust relationships had to be established between servers in the DataLake, relying on certificate authorities from different, and sometimes incompatible, trust stores.

Teapot and ALISE, both developed during the project, address different integration challenges. Teapot provides a WebDAV interface for storage systems lacking native internet-accessible WebDAV, enabling seamless integration into the DataLake. ALISE facilitates site-local account linking, allowing sites to reconcile users' local and federated identities.

Using these approaches, the project has integrated multiple sites with diverse storage backends—including dCache, Storm, S3, and Teapot. An overview of the DataLake components overlaid on a map of Europe is shown in **Figure 5** Overview of the DataLake components on the map of Europe. Storage sites are depicted as white cylinders with logos; those deploying Teapot display a small Teapot icon. Compute resources are shown as black rectangles. A public repository is represented by a cylinder with thick black outlines., marking sites providing storage and compute resources alongside key components like Rucio and FTS.

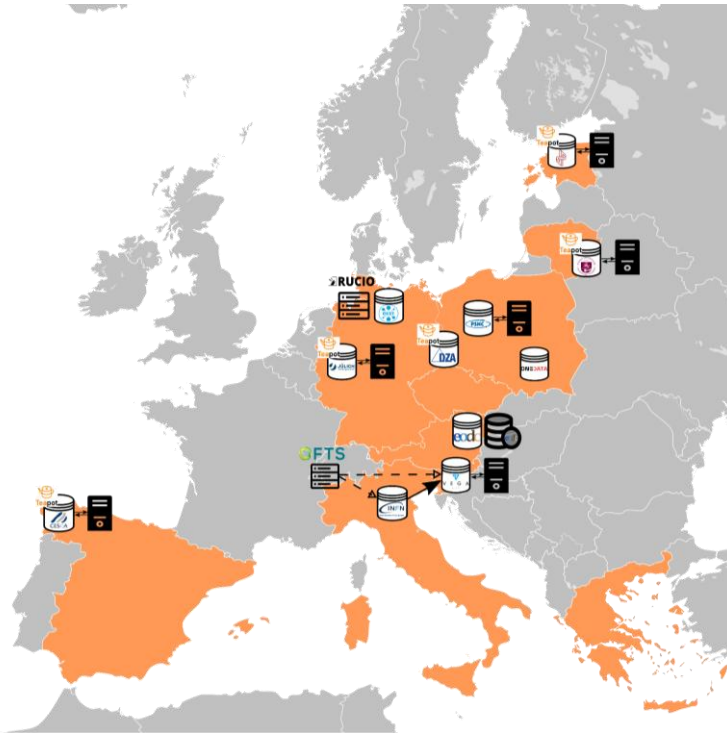


Figure 5 Overview of the DataLake components on the map of Europe. Storage sites are depicted as white cylinders with logos; those deploying Teapot display a small Teapot icon. Compute resources are shown as black rectangles. A public repository is represented by a cylinder with thick black outlines.

4.4.1 Storage integration

The following sites have been successfully integrated into the DataLake:

- **DESY** – While not part of the interTwin resource providers, DESY contributed to the project by offering a storage endpoint based on their local dCache storage. The dCache service natively supports WebDAV, and access was authorized by mapping all users of a given EGI Check-in VO to a single functional account. This setup enabled straightforward integration with the interTwin DataLake.
- **INFN** – INFN contributed a storage endpoint used by the Virgo scientific community. Their storage service is based on Storm¹⁶, which supports WebDAV natively, and user authorization was implemented based on external identities. This allowed for seamless integration with interTwin’s federated authentication and data management layers.
- **IZUM** – VEGA was the first HPC center integrated into the DataLake. It provides a dCache-based storage endpoint, where all users of a VO are mapped to a shared functional account. In addition to storage, VEGA offers compute capabilities; the dCache storage is mounted on the compute nodes, allowing POSIX access to data. The integration was implemented in close collaboration with the VEGA team.
- **EODC** – EODC contributed a bucket on their object storage, accessible via the S3 protocol. Authorization is based on signed URLs rather than user identities, with Rucio automatically generating time-limited credentials. At EODC, the public data repository was also integrated into the DataLake as a non-deterministic RSE. This integration facilitates automated ingestion of selected publicly available climate

¹⁶ <https://italiangrid.github.io/storm/index.html>

datasets. Because non-deterministic RSEs do not follow Rucio's deterministic naming logic, full file paths must be registered manually.

- **EGI:** EGI DataHub – Cyfronet implemented the Onedata support for S3 in the project and contributed a bucket. Authorization is handled through time-limited signed URLs automatically generated by Rucio. As with EODC, this enables seamless integration of the EGI DataHub storage into the DataLake.
- **PSNC** – Integration of PSNC into the interTwin DTE platform involves integrating both their HPC compute and corresponding storage into the federated data management layer. After discussion with the PSNC team, the adopted approach involved updating their dCache storage service to support the interTwin DataLake (protocols and authentication scheme) and integrating their dCache storage service into their HPC system. To achieve this, they updated their dCache service, deployed ALISE and NFS-mounted dCache on the HPC nodes accessible by interTwin jobs.
- **KBFI** – Integration of KBFI into the interTwin DTE platform has demonstrated the ability to include resources from HTC-style providers. The KBFI storage was provided as a POSIX-like layer. We developed a container image of Teapot that allowed KBFI admins to deploy Teapot according to their preferred deployment strategy. The desired mapping is that all interTwin users have the same identity on the KBFI infrastructure. Therefore, the Teapot instance was configured to map all interTwin users to a single KBFI identity, alleviating the need to use ALISE. This storage is also available to KBFI compute jobs, with the exploitation of the computing activity being an ongoing activity.

4.4.2 Certificate Authorities

Another key point involves which certificate authorities are trusted. The interTwin DTE platform has used the IGTF¹⁷ trust distribution as the basis for trusting services within the project.

Note that the IGTF trust distribution is distinct from the CA/B trust distributions¹⁸. The CA/B trust distribution is commonly used by web browsers, operating systems and other global contexts. Until recently, there was no overlap between these two trust distributions; however, this has changed with the Terena Certification Service adopting HARICA as the certificate authority that issues TCS certificates. TCS-issued server certificates effectively bridge the IGTF and CA/B trust realms, allowing services to be useful in both trust contexts.

Obtaining TCS certificates requires an up-front investment that is not always practical. For example, there is often no direct path through which the people running community-specific services can obtain TCS-issued certificates for their services. Moreover, when CA/B is the agreed trust distribution, a service might find limited motivation for deploying certification for compatibility with potential clients that only trust IGTF.

In order to support these services, a set of additional trust anchors were developed that allows the inclusion of common CA/B certificate authorities (e.g., Let's Encrypt, Sectigo, Amazon) into a site's IGTF trust store. In effect, this allows admins running a service to

¹⁷ <https://www.igtf.net/>

¹⁸ <https://cabforum.org/>



extend that service's set of trust anchors to include both IGTF and the necessary subset of CA/B.

These CA/B certificate authority packages have been developed and deployed at three key storage sites: VEGA, PSNC, and KBFI. This allows them to transfer data from services deploying (for example) Let's Encrypt certificates.

4.5 Pilots and testing activities

We have distributed our storage resources across three DataLake testbeds: the development DataLake, the Virgo DataLake, and the LatticeQCD DataLake.

4.5.1 Development DataLake

The development DataLake is the primary environment used by most of the project's use cases. Federated identities are provided by EGI Check-in, with authorization based on membership in the dev.intertwin.eu VO. The DataLake consists of:

- Rucio as the data management layer
- FTS for orchestrating data transfers
- Storage endpoints at the following sites: DESY, VEGA, EODC, ONEDATA, PSNC, and KBFI
- Integration of EODC's public data repository as a non-deterministic RSE

We are currently working to integrate storage from three additional HPC sites into the development DataLake by the end of the project: JSC, DZA (TU Dresden), and the University of Vilnius. In each case, the integration approach is based on deploying Teapot. At JSC, Teapot has been deployed in a virtual machine with NFS access to the HPC filesystem. A challenge here is that the available NFS variant does not support extended attributes, which are required by Teapot.

DZA, although not part of the interTwin consortium, is contributing its storage to the DataLake, supporting the broader goals of interoperability and adoption. DZA's facilities are currently concentrated at TU Dresden. An adoption plan has been agreed upon, and work is ongoing on a minimum viable implementation to demonstrate the concept. These efforts are expected to continue beyond the project's lifetime to adopt the technology in their premises.

At the University of Vilnius, integration targets their HPC facility and its corresponding filesystem. The agreed strategy involves deploying Teapot on a login node with direct access to storage. Identity mapping will be handled within Vilnius's existing authentication system, avoiding the need to deploy ALISE. As the university joined the project at a later stage, implementation efforts began during the final project phase. However, we aim to complete the integration by the project's end.

4.5.2 Virgo DataLake

The Virgo DataLake supports the needs of the Virgo gravitational wave detector use case (Task 4.4). Due to the private nature of their data — accessible only to members of the Virgo Collaboration — the DataLake needed to be isolated from the broader development DataLake. A separate instance of Rucio governs this environment, with authorization based on membership in the virgo.intertwin.eu VO. Identity federation is provided by EGI Check-in, and FTS handles data transfers.

This DataLake currently includes two storage endpoints: INFN, where Virgo's data is stored, and VEGA, which provides HPC compute for data processing.

4.5.3 LatticeQCD DataLake

The LatticeQCD DataLake supports the specific needs of the LatticeQCD scientific community. In this community data is typically registered and managed using the ILDG¹⁹ catalog, which is well-established and when needed enforces embargo periods before data becomes public.

The ILDG catalog uses INDIGO IAM²⁰ for identity management, supporting fine-grained authorization based on scopes. This model is incompatible with Rucio's current authorization mechanisms. Instead of reimplementing access control in Rucio, we chose to integrate the ILDG catalog directly into a DataLake-based solution governed by ILDG itself. This environment:

- Uses INDIGO IAM as the identity provider instead of EGI Check-in
- Does not use Rucio, but still leverages FTS for data transfers

To support long-running transfers (which exceed the lifetime of a typical access token), FTS normally requires a refresh token. However, ILDG policy prohibits the use of refresh tokens for scope-sensitive transfers. In collaboration with the FTS development team, we configured FTS to operate without requesting new tokens. As a result, users must manually restart expired transfers with new access tokens.

This LatticeQCD DataLake includes three storage endpoints using local dCache: DESY-Hamburg, DESY-Zeuthen, and JSC.

A fourth storage endpoint is being added at CESSGA, where integration is being pursued through Teapot. A Teapot instance has been deployed in a virtual machine with access to non-HPC storage, and data management has been demonstrated as a proof-of-concept. Discussions are underway on how to integrate Teapot into the facility, to gain access to the HPC filesystem. We expect to have a functional setup by the end of the project, capable of transferring data to CESSGA and using it in HPC jobs.

4.5.4 STAC Integration Pilot

An additional pilot within the development DataLake aimed to integrate STAC²¹, a standard used by climate science communities to describe datasets. This work was developed in collaboration with the scientific communities behind the climate-based use cases and the teams at EODC and Eurac.

Community users expressed a need to:

1. Generate STAC catalogs from their DataLake analysis outputs
2. Determine whether data referenced in an existing STAC catalog already exists in the DataLake

To support this, we implemented a mapping between STAC's flexible hierarchy (Catalog → Collection → Item → Asset) and Rucio's strict structure (Container → Dataset → File). A reconciliation scheme was developed, and cross-references were added:

¹⁹ <https://hpc.desy.de/ildg/>

²⁰ <https://iam-ildg.cloud.cnaf.infn.it>

²¹ <https://stacspec.org>



D5.5 DTE infrastructure development and integration report

- In STAC, a new metadata entry such as [rucio://rucio-intertwin-testbed.desy.de/MY_SCOPE:MY_NAME](#) identifies the corresponding Rucio data
- In Rucio, metadata was added referencing the associated STAC entry and catalogue


Additionally, data from a given STAC catalogue were grouped under a common Rucio scope to improve organization and discoverability.



5 Task 5.3 Federation Services and Policies


The main goals of Task 5.3 are security and accountability for users and providers by creating a bi-directional policy chain of trust across the stack between User and Resource. This includes Access policies and rules of participation, to be mediated by the DTE. Task 5.3 delivers a dedicated Resource usage accounting repository as well whose architecture and integration are reported below.

5.1 APEL Accounting

Component name and logo	APEL Accounting 
Page on interTwin website	https://www.intertwin.eu/article/infrastructure-component-apel-accounting
Description	The Accounting Repository stores compute (serial and parallel jobs), storage, and cloud resource usage data collected from resource centres within an e-infrastructure. Accounting information is gathered from distributed sensors into a central Accounting Repository where it is processed to generate summaries that are available through an Accounting Dashboard.
Value proposition	Aggregated statistics on resource usage available via a central dashboard to enable resourcing and funding decisions to be made.
Users of the Component	Resource centre admins; Research community managers
User Documentation	https://docs.egi.eu/providers/operations-manuals/man09_accounting_data_publishing/
Technical Documentation	https://github.com/apel/ssm/blob/dev/README.md
Responsible	STFC UKRI
Licence	Apache 2.0
Source code	<ul style="list-style-type: none"> • https://github.com/apel/apel (client and server software) • https://github.com/apel/ssm (messaging tool)

	<ul style="list-style-type: none"> • https://github.com/apel/grafana-dashboards/ (dashboard)
Language	Python

5.2 Kubernetes Usage Accounting Probe

Component name and logo	Kubernetes Usage Accounting Probe 
Page on interTwin website	https://www.intertwin.eu/article/infrastructure-component-kubernetes-usage-accounting-probe
Description	The tool extracts usage information from a local Prometheus database in a Kubernetes Cluster and produces an OGF Cloud Usage Record that can be consumed by APEL. It is designed for regular execution at arbitrarily chosen intervals, to provide usage updates at selected granularity.
Value proposition	Usage statistics are provided in sufficient detail to track, over time, the resource consumption by any supported user group or individual.
Users of the Component	Resource centre admins
User Documentation	N/A
Technical Documentation	https://github.com/intertwin-eu/kubernetes-usage-accounting/blob/master/README.md
Responsible	CESNET, EGI
Licence	ASL 2.0
Source code	https://github.com/intertwin-eu/kubernetes-usage-accounting

5.3 Functionalities developed since the last release

To enable data to be loaded from the interTwin Kubernetes systems, a modification was required to APEL Accounting as Kubernetes can allocate fractions of CPUs. Previously the APEL Accounting handled CPU counts as integers. Both the APEL software and its



database schema were modified to allow decimal values of CPU count to be handled through the accounting workflow.

The APEL Accounting server and interTwin Accounting Dashboard are now both fully configuration-managed, meaning that their configuration is version-controlled and reviewed before being deployed to a production environment. The interTwin Accounting Dashboard has been made externally accessible²² through an HAProxy load balancer, which simplifies changing server hosts behind the scenes.

The Dashboard has been successfully integrated with the production instance of EGI Check-in. Although no login is required to view the data, EGI Check-in integration makes it easier to manage administrators of the service and can in future be used to protect any data that should only be viewed by select users.

Several improvements have been introduced to the Kubernetes Usage Accounting Probe during its development, particularly under the interTwin project. Configuration has been greatly simplified: the default mode does nothing, and individual outputs or targets can be enabled selectively. Support has been added for both grid SSL certificates and messaging tokens required by APEL, improving compatibility with secure environments. The APEL messaging client has been updated to version 3.4.1 to reflect upstream changes, and metrics queries have been revised to maintain compatibility with recent versions of Prometheus.

To ensure accuracy in usage reporting, CPU durations are now rounded to the nearest second as expected by APEL. Additionally, containers that do not request any CPU resources are excluded by default from accounting output to avoid misleading usage data.

While some of these changes are specific to APEL, others — such as the improvements to metrics handling and simplified deployment — are beneficial to a broader range of use cases and projects. These enhancements improve not only the flexibility and maintainability of the probe, but also its suitability for integration into diverse infrastructures where consistent and auditable usage tracking is needed.

5.4 Integrations with sites and other DTE components

As a centralised accounting service, APEL Accounting can accept usage records from any site that can provide the data in a compatible format and that complies with the authorisation and authentication. Therefore, we have set up an accounting flow from CESNET and UKRI to the APEL Accounting system at STFC. This collects the usage data for the Kubernetes cluster.

From the other perspective, the described accounting system is a completely high-level framework agnostic and, by design, it is not expected to integrate DTE components. As the account enables the underlying K8s layer this implies that K8s has the responsibility to provide resource usage information irrespective of the actual service (and thus any component) that runs on top of it.

²² <https://intertwin-accounting.stfc.ac.uk>



5.5 Pilots and testing activities

APEL Accounting and the interTwin Accounting Dashboard have been running in a production configuration since the start of the year. Data from both development and production Kubernetes systems has been sent from CESNET and UKRI and loaded into the APEL Accounting system. An example view of the data can be seen in **Figure 6** interTwin Accounting Dashboard, with development and production data.

Once data appeared in the Dashboard, feedback was collected so that improvements could be made to the UI, such as the terminology used on the charts.




Figure 6 interTwin Accounting Dashboard, with development and production data

6 Task 5.4 AI-based orchestrator

The objective of Task 5.4 is to orchestrate compute resources "intelligently", taking into account data location and performance metrics. It implements algorithms for task placement using information about federated clouds. It enhances the existing scheduling capabilities of INDIGO PaaS Orchestrator

6.1 AI-based orchestrator

Component name and logo	AI-based orchestrator 
Page on interTwin website	https://www.intertwin.eu/article/infrastructure-component-ai-based-orchestrator
Description	<p>The AI-based orchestrator extends the INDIGO PaaS Orchestrator with:</p> <ul style="list-style-type: none"> - A new service exploiting Machine Learning techniques to infer the best federated provider to submit the user deployment. - A new service for data management able to communicate with federated Rucio to improve the deployment procedure. - Support for EGI-Checkin through INDIGO IAM. - Support for Kubernetes-based deployments through the Infrastructure Manager (IM).
Value proposition	<p>The INDIGO PaaS Orchestrator enables seamless orchestration of workloads across heterogeneous infrastructures, including cloud platforms like OpenStack and Kubernetes, and even public cloud providers. It allows for transparent deployment of containerized or virtualized applications, supporting automated scheduling, elasticity, and full lifecycle management of both applications and data services.</p> <p>One of its key strengths lies in its application-centric abstraction model. Users can define their applications and resource requirements through simple TOSCA YAML templates, effectively hiding the complexity of the underlying infrastructure. This approach makes it easy to deploy even complex, multi-component applications that rely on persistent storage, stateful execution, and advanced networking capabilities.</p> <p>The orchestrator is particularly well-suited for data-</p>

	<p>intensive and AI-driven workflows. It is designed to handle scenarios that involve massive data movement, machine learning training, or scientific simulation tasks. Thanks to its integration with data federation services and scientific repositories, it ensures data-aware and on-demand execution of such workflows.</p> <p>Built on open standards and technologies like TOSCA and OASIS, the INDIGO PaaS Orchestrator guarantees interoperability and avoids vendor lock-in. It also integrates natively with other INDIGO-DataCloud components, such as the Identity and Access Management (INDIGO-IAM) system, and data management services, providing a cohesive and extensible ecosystem.</p> <p>Finally, it supports federated identity and policy-based access control, making it ideal for collaborative research contexts where resources and users span across multiple institutions.</p>
Users of the Component	The INDIGO PaaS orchestrator
User Documentation	https://github.com/inf-n-datacloud/orchestrator
Technical Documentation	https://github.com/inf-n-datacloud/ai-ranker/blob/main/README.md
Responsible	INFN
Licence	Apache 2.0
Source code	<ul style="list-style-type: none"> - https://github.com/inf-n-datacloud/ai-ranker - https://github.com/inf-n-datacloud/rucio-connector
Language	Python

6.2 Functionalities developed since the last release

The AI-Ranker is a new service replacing the legacy Cloud Provider Ranker.

The service consists of two main components:

- One for training Machine Learning (ML) models and storing them in an MLflow registry;
- One for performing inference using the trained models retrieved from the registry.

The system relies on an MLflow instance to manage model versions and a Kafka instance to handle message-based communications for both training and inference. The Kafka instance is not mandatory, as messages can also be read from a local file, although this is



intended for debugging purposes only. Currently, only models built with the Scikit-learn library are supported.

The typical usage pattern is as follows:

- The training script is scheduled to run at regular intervals (e.g., via a job scheduler).
- It consumes messages from the Kafka training queue, preprocesses them, and uses them to train two types of ML models: a classification model for predicting the success or failure of a deployment, and a regression model for estimating deployment creation or failure time.
- The trained models, along with their metadata, are stored in the MLflow registry.
- The inference script listens to the Kafka inference queue, reads new messages as they arrive, preprocesses them, loads the latest ML models from the MLflow registry, and uses them to make predictions for each Cloud provider in the message.
- The predictions are then combined to produce an ordered list of providers based on expected performance.

A set of ancillary components is required to collect the user deployment requests.

For data management the PaaS Orchestrator and the dashboard interact with a new Python component dealing with Rucio.

The component exposes a set of REST API to communicate with multiple federated Rucio instances and query the list of RSEs with a replica of the given data identifier (DID). A user authenticated using a specific identity provider is able to list only the Rucio instances supporting that identity provider and giving the user the correct access rights.

The same component has a background thread listening on a dedicated Kafka topic where requests to replicate data are sent in case the target deployment requires moving the data into another RSE.

The typical usage patterns are the following.

- Lists all RSEs (and corresponding federated sites) providing a replica for an existing file owned by the user, allowing the user to submit a scheduled deployment on one of the sites with an RSE with the replica of the desired data.
- Given the list of RSEs (and corresponding federated sites) providing a replica for an existing file owned by a user and the size of the file, the PaaS Orchestrator can determine which is the most suitable provider and eventually replicate the data on another RSEs to boost the efficiency of the deployment. The data replication happens after the AI-ranker has detected the best provider.

The PaaS Orchestrator can deploy high level services on kubernetes clusters. Based on the metadata written in the TOSCA template, the PaaS Orchestrator sends the deployment request to the IM with the correct parameters. Although with some limitations, the IM can deploy application architectures described as TOSCA templates on Kubernetes clusters.

The federation-registry service and federation-registry-feeder script have been improved to federate kubernetes clusters and Rucio instances.

Finally, the support for EGI Check-in is given through INDIGO-IAM²³.

²³ <https://github.com/indigo-iam/iam>

6.3 Integrations with other DTE components

In the case of the PaaS Orchestrator, we only have direct interactions with services belonging to the group of the core services of the platform.

The main services integrated with the PaaS are EGI Check-in, Infrastructure Manager, and Rucio.

The Rucio instance is used to manage the replica of the data across multiple sites. In order to let the PaaS leverage the Rucio capabilities, both of them have to be connected to the same identity providers. The configuration of authorisation access rights is in charge of the Rucio instances. Users can't create new DIDs through the PaaS but they can replicate existing ones.

6.4 Integration with Sites

Within the context of the interTwin project the PaaS orchestrator has been configured to federate and leverage 4 different sites, INFN-Bari-ReCaS, INFN-CNAF, INFN-Backbone and GRNET.

More general the federated sites must grant access to a service user with read access to the resources usage and limits.

6.5 Pilots and testing activities

The main testing activity for the INDIGO PaaS Orchestrator focused on two new components:

- The AI-ranker service was tested in an INFN testbed where a kafka instance was installed together with various producers and consumers in charge of collecting, elaborating and forwarding data between them. Of course, the *federation-registry-feeder* and *rally collector* were also made available. The *inference service* is responsible for the interaction with MLFlow and to infer the best provider for deploying the user specific request.
- The Rucio-connector has been tested on a pre-production instance supporting multiple OpenStack providers and a single Rucio instance with multiple RSEs and supporting an identity provider different from the one supported by the whole PaaS Orchestration system. It has been verified that through the dashboard, the users specify the target file name and this request is forwarded to the rucio-connector.

Further integrations are ongoing in order to enable more high-level services developed by interTwin to be deployed via INDIGO PaaS Orchestrator.

7 Conclusions

The D5.5 reports on the activities made by WP5 in order to successfully implement the architecture of the DTE infrastructure. A description of the composable toolkit is provided in order to show the variety of integration patterns that can be supported and how it could be adopted by new use cases. In this respect, as shown in **Table 1**. Seven major components have been used to build the architecture of the DTE covering all the areas (pillars) identified since the beginning of the project: compute federation, data and storage federation and resource accounting.

A lot of effort has been invested to provide a harmonic system to enable the continuum model both for compute and data. Two brand new systems have been developed, interLink and Teapot, coping with compute and data respectively. Since a peculiarity of interTwin has been to involve highly heterogeneous sites offering different provisioning models (Cloud, HTC, and HPC) a lot of co-design was possible. WP5 was able to build several testbeds to successfully implement pilot integrations with communities.

Sites nicely supported early scale tests in order to prove the scalability of the architecture. In this respect, the first results were shown in terms of concurrent GPUs used through the offloading exploiting the VEGA computing capacity. So far up to 160 GPUs were accessed simultaneously, however more tests are ongoing at the time of writing. Finally, during this very final part of the project, we are proving that the DTE Infrastructure can enable access to Quantum machines. This is an activity we are carrying out in collaboration with CESGA.

8 References

Reference	
No	Description / Link
R1	interTwin D5.1 First Architecture design and Implementation Plan https://zenodo.org/records/10417147
R2	interTwin D6.4 Final release of the DTE core modules https://zenodo.org/records/14778361
R3	M. Bunino, R. Sarma, J. S. Sæther, A. E. Lappe, K. Tsolaki, K. Verder, H. Mutegeki, R. Machacek, A. Zoechbauer, M. Ruettgers, I. Luise, E. Wulff, M. Girone, and A. Lintermann, "itwinai," Jan. 2025. [Online]. Available: https://github.com/interTwin-eu/itwinai
R4	S. Risco, G. Moltó, D. M. Naranjo, and I. Blanquer, "Serverless Workflows for Containerised Applications in the Cloud Continuum," Journal of Grid Computing, vol. 19, no. 3, p. 30, Jul. 2021. [Online]. Available: https://doi.org/10.1007/s10723-021-09570-2
R5	S. Risco, C. Alarcón, S. Langarita, M. Caballer, and G. Moltó, "Rescheduling serverless workloads across the cloud-to-edge continuum," Future Generation Computer Systems, vol. 153, pp. 457–466, Apr. 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X23004764

