D6.5 Core development and integration on DTs report

Status: Under EC Review
Dissemination Level: public

Funded by the European Union

Abstract

Key Words DTE, Core, Workflow, Quality, AI/ML, event driven, serverless

The interTwin project is developing a robust Digital Twin Environment (DTE) that integrates real-time data, simulations, and system orchestration. Key challenges include connecting workflow orchestration with real-time data systems, interfacing with cloud and HPC infrastructure, and providing scalable data processing and accessible APIs.

Core components include the Infrastructure Manager for deploying Digital Twins, OSCAR for triggering real-time jobs in Kubernetes, and SQAaaS for automated quality assurance in ML workflows. Ophidia supports multidimensional data analytics with provenance tracking, while yProv manages workflow traceability in coordination with SQAaaS, openEO, and Ophidia. openEO provides an API for Earth Observation workflows, now enhanced for real-time data via OSCAR. To support scalable ML, the itwinai Toolkit offers an open-source solution for running ML workloads on HPC and cloud systems, featuring unified logging, provenance tracking, and quality metrics.

This document describes the final release of the core components in terms of functionalities, integration with the DTE ecosystem, and testing performed.

Document Description			
D6.5 Core development and integration on DTs report			
Work Package number 6			
Document type	Deliverable		
Document status	UNDER EC REVIEW	**Version**	1.0
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	CSIC		
Document link	https://documents.egi.eu/document/3954		
DOI	https://zenodo.org/records/16529402		
Author(s)	• Matteo Bunino (CERN) • Linus Maximilian Eickhoff (CERN) • Jarl Sondre Saether (CERN) • Rakesh Sarma (Juelich) • Estibaliz Parcero (UPV) • Miguel Caballer (UPV) • Sandro Fiore (UNITN) • Ivan Palomo (CSIC) • Samuel Bernardo (LIP) • Donatello Elia (CMCC) • Cosimo Palazzo (CMCC) • Juraj Zvolensky (EURAC) • Alexander Jacob (EURAC) • Andrea Manzi (EGI Foundation) • Isabel Campos (CSIC)		
Reviewers	• Giacinto Donvito (INFN) • Michele Claus (EURAC)		
Moderated by:	• Andrea Anzanello (EGI Foundation)		
Approved by	Andrea Cristofori (EGI Foundation) on behalf of TCB		

Revision History

Version	Date	Description	Contributors
V0.1	12/05/2025	Created template	Andrea Manzi(EGI Foundation)
v0.2	11/07/2025	Version ready for internal review	All authors
v0.3	17/07/2025	Internal review	Michele Claus (EURAC)
V0.4	25/07/2025	internal and TCB Review	Giacinto Donvito (INFN), Andrea Cristofori (EGI Foundation)
v0.5	25/07/2025	Version ready for QA	Isabel Campos (CSIC)
V1.0		**Final**	

Terminology / Acronyms

Term/Acronym	Definition
AI	Artificial Intelligence
API	Application Programming Interface, aka programmatic interface of a computer system through which other computer systems can interact with it
CI/CD	In software engineering, CI/CD is the combined practices of continuous integration and continuous delivery
CLI	Command Line Interface
CRUD	Create Read Update and Delete
CWL	Common Workflow Language
DAG	Directed Acyclic Graph
DT	Digital Twin, a digital representation of an actual physical product, system or process that serves as the effectively indistinguishable digital counterpart of it for practical purposes, such as simulation, integration, testing, monitoring, and maintenance.
DTE	Digital Twin Engine, a platform to build DTs
FaaS	Function as a Service
GAN	Generative Adversarial Network
HL-LHC	High Luminosity Large Hydron Collider
HPC	High Performance Computing
HPO	Hyperparameter Optimization
IaC	Infrastructure as Code
ICT	Information and Communication Technology
GUI	Graphical User Interface
JSON	JavaScript Object Notation
LLM	Large Language Model
LUMI	Large Unified Modern Infrastructure
ML	Machine Learning is a branch of AI and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy
OSCAR	Open Source Serverless Computing for Data-Processing Applications
ROCm	Radeon Open Compute

REST API	API that conforms to the design principles of REST, or representational state transfer architectural style
SQAaaS	Software Quality Assurance as a Service
TOSCA	Topology and Orchestration Specification for Cloud Applications
XML-RPC	XML-RPC is a remote procedure call (RPC) protocol that uses XML to encode its messages and HTTP as the transport mechanism

Table of Contents

List of Figures

List of Tables

Executive summary

The **interTwin project** focuses on building a robust **Digital Twin Environment (DTE)** that provides support to simulations of real systems and real-world data. This requires dynamic integration of simulation execution, real-time data ingestion, system orchestration, and behavioural monitoring.

The Information and Communication Technology (ICT) Architecture and Integration Challenges comprise: seamless integration of workflow orchestration with real-time data acquisition systems; interface directly with infrastructure services (cloud, HPC); provide scalable, embedded data processing pipelines; offer interoperable and accessible APIs for end-users.

The core of the DTE consists of several components. The **Infrastructure Manager** (IM) automates the deployment of Digital Twins (DTs) across hybrid environments, including cloud and High-Performance Computing (HPC) infrastructures. Open Source Serverless Computing for Data-Processing Applications (**OSCAR)** is a serverless, event-driven processing framework for Kubernetes that triggers real-time jobs based on storage events, such as data uploads. **SQAaaS platform** is a DevOps-driven quality assurance platform extended to support automated machine learning (ML) models and data validation workflows. **Ophidia** is a multidimensional scientific data analytics framework that now supports W3C PROV[1] provenance standards, Common Workflow Language (CWL) based workflows, and integrates with SQAaaS. **yProv** is a provenance management tool that works in conjunction with SQAaaS, openEO, and Ophidia to ensure traceability in scientific workflows. **openEO** provides an API for cross-backend workflow execution, mainly targeted at Earth Observation applications, and now integrates with OSCAR for handling real-time data streams.

In terms of ML-specific developments, DTs must accommodate scalable ML pipelines capable of processing high-volume data analytics. The **Itwinai** Toolkit is an open-source ML/AI toolkit developed to minimize engineering overhead. It supports scalable ML workloads on HPC systems through interlinking with WP5, while also enabling workflows in cloud environments. The toolkit includes unified logging to track ML metadata, provenance (through yProv), and quality metrics (via SQAaaS).

The document is structured to provide an overview of the enhanced core DTE components, followed by detailed task-level (WP6) integration with other work packages, particularly WP4 (Digital Twin applications) and WP5 (infrastructure). It concludes with a summary of pilot and testing activities and outlines the future roadmap.

[1] https://www.w3.org/TR/prov-overview/

1 Introduction

1.1 Scope

A fundamental requirement of a DT is the continuous validation of simulation outputs against real-world data, enabling dynamic synchronization between the virtual and physical systems. Core functionalities include simulation execution, system integration, ingestion of real-world data, and behavioral monitoring.

The ICT challenge in the interTwin context involves seamless integration of workflow orchestration tools with real-time data acquisition systems, direct connectivity with infrastructure-level services, and embedding scalable data processing capabilities within the environment. Ensuring accessibility and usability for end-users via interoperable interfaces is one of the key features that the DTE design has addressed.

In the course of the project a number of tools have been developed and expanded to support such core capabilities:

- **IM** – Deploys customized computing clusters across Cloud and HPC (High-Performance Computing) resources. In the course of interTwin the IM has been enriched with support for recipes for automated deployment of DTs.
- **OSCAR** – is an open-source, serverless platform designed for event-driven data processing on elastic Kubernetes clusters. It focuses on real-time handling of storage-related events. For example, when a user uploads data to storage, an event is triggered that launches an OSCAR Job to process it. This is a key requirement to satisfy for most DTs.
- **SQAaaS** – A DevOps-based platform for validating software and data quality. In the course of interTwin SQAaaS has been enriched with features to enable automated model validation.
- **Ophidia** – Open-source framework for analyzing scientific multidimensional data with workflow support. Ophidia now supports W3C PROV provenance documents, and includes validation features integrated with the SQAaaS platform.
- **yProv** – Open-source service for managing provenance in scientific workflows. In the course of interTwin it has been integrated with the SQAaaS system to address traceability. Support for openEO, ML and integration with Ophidia is now also present.
- **openEO** – API for managing and interacting with scientific workflows across different backends designed for the Earth Observation community. In the course of interTwin it has been integrated with OSCAR to support features involving real-time data streams.

The Machine Learning challenges in interTwin require implementing a high level of interoperability. DTs need to perform scalable analysis of high-volume datasets by leveraging machine learning and advanced data analytics frameworks. In parallel the learning process needs to integrate with workflow orchestration tools that support real-time data ingestion. Additionally, the DTE needs to be able to connect to the infrastructure level to use data processing capabilities. Robust connectivity and integration with core infrastructure services and platforms are essential to maintain system performance and reliability.

In interTwin we have designed an open-source toolkit focusing on accelerating AI and ML workflows for researchers, **itwinai**. The toolkit focuses on reducing the engineering burden researchers face when dealing with generic AI and ML. Using interlink (WP5) it enables scaling machine learning projects to HPC resources while seamlessly integrating with cloud-based services. itwinai features a unified logger interface that helps track ML metadata, provenance (yProv), and quality assurance (SQAaaS).

1.2 Document Structure

The document is structured as follows: we begin with a high-level overview of the core modules that have been developed or enhanced to meet the requirements of DTs.

Next, we detail these core modules in the context of the individual tasks within WP6. This includes a description of their main functionalities, recent developments during the current reporting period, integration with other DTE components, and alignment with DT applications from WP4. We also provide a summary of the pilot and testing activities, carried out primarily in collaboration with WP5 at the infrastructure level.

Finally, we outline the conclusions and outlook for future usage.

2 Overview of The DTE Core Modules

The list of core components part of Work Package 6 is shown in **Table 1**.

Table 1 The table provides a compact view of the modules developed and enhanced by WP6, including a short description, the task responsible for developing and operating the module, and finally highlighting the added value each one brings to the DTE Core

Module name	Short Description	Task	Key Feature in the context of DTE Core
OSCAR	open-source platform to support the event-driven serverless computing model for data-processing applications	6.1	Easy execution of containerized workflows in multiple backends
DCNiOS	open-source command-line tool to easily manage the creation of event-driven data processing flows	6.1	Execution of OSCAR workflows following the ingestion of data to a storage/Data Lake.
openEO	application programming interface (API) that supports i) the management of workflows, ii) job handling, and iii) linking to data sources and processing capabilities on compatible cloud platform providers in a standardised way.	6.1 and 6.3	Easy composition of EO data processing flows, including data fusion and execution on cloud backends
Ophidia	open-source solution for the analysis of scientific multi-dimensional data, joining HPC paradigms and Big Data approaches	6.1	Supports the execution of complex analytics workflows in the form of Directed Acyclic Graphs (DAGs)
yProv	open-source software ecosystem to support provenance management within scientific workflows	6.1	Management (i.e. store, retrieve, explore, visualise) of the provenance information associated with scientific datasets and AI models

SQAaaS	Platform for quality assessment and awarding of multiple digital objects (source code, services, data)	6.2	Provides a module for quality validation within the interTwin's DTE
SQAaaS Github action	Trigger SQAaaS quality assessment service from GitHub actions	6.2	Integrate interTwin's GitHub organisation with the SQAaaS platform for software quality (incl. workflow and model code)
Ophidia Workflow Validation tool	SQAaaS tool to validate the Ophidia workflow.	6.2	Integrate interTwin's workflows within pipeline steps, creating assessments, getting outputs, checking pipeline status
Ophidia Workflow Reporting plugin	SQAaaS reporting plugin for Ophidia workflow validation	6.2	Helps users identify any errors or issues in their Ophidia workflows, allowing them to make necessary corrections and improvements.
IM	Infrastructure as Code (IaC) tool that receives Topology and Orchestration Specification for Cloud Applications (TOSCA) based virtual infrastructure provision requests for their deployment on IaaS cloud back-ends	6.4	Users can self-provision any kind of complex virtual infrastructure on whichever cloud infrastructure they can access.
TOSCA templates	As a set of TOSCA templates to deploy Big Data Analytics tools	6.4	TOSCA templates enable the description, in a cloud-agnostic way, of the virtual infrastructures needed in the available Big Data Analytics tools.
Configuration artefacts	Ansible playbooks and roles and any other artefact needed in the deployment of the selected Big Data Analytics tools.	6.4	TOSCA templates refer to these artefacts to enable the configuration of the Big Data Analytics tools using the Ansible tool.

itwinai	Toolkit focussing on accelerating AI and ML workflows for researcher	6.5	Development of AI pipelines and execution on different computing backends

3 Core Modules and final development and integration activities

3.1 Task 6.1

3.1.1 OSCAR

Component name	OSCAR **https://oscar.grycap.net**
Description	OSCAR is an open-source platform to support the event-driven serverless computing model for data-processing applications. It can be automatically deployed on multi-clouds, and even on low-powered devices, to create highly parallel event-driven data-processing serverless applications along the computing continuum. These applications execute on customised runtime environments, provided by Docker containers which run on elastic Kubernetes clusters.
Value proposition	Users can set up an OSCAR cluster themselves on any available cloud infrastructure. The automatically scalable cluster can be used to scale file-based on-demand processing (e.g. automatically when a file is uploaded to an object store such as MinIO[2]). Furthermore, execution can be triggered via HTTP-based calls for programmatic interaction with auto-scaled, stateless, user-defined services.
Users of the Component	Scientific users require data-driven processing on multiple cloud back-ends. Non-expert users can interact via high-level web-based GUIs, while advanced users can use a command-line interface (CLI).

[2] https://min.io/

User Documentation	https://oscar.grycap.net/blog/
Technical Documentation	https://docs.oscar.grycap.net/
Responsible	GRyCAP-I3M-UPV - products@grycap.upv.es Contact point: Germán Moltó - gmolto@dsic.upv.es
License	Apache 2.0
Source code	https://github.com/grycap/oscar

3.1.2 DCNiOS

Component name	DCNiOS
Description	DCNiOS (Data Connector through NiFi for OSCAR) is an open-source command-line tool to easily manage the creation of event-driven data processing flows.
Value proposition	When files are uploaded to a data storage, events are ingested by Apache NiFi[3], which can queue them up depending on the (modifiable at runtime) ingestion rate. Then, they are delegated for processing into a scalable OSCAR cluster, where a user-defined application, based on a Docker image, can process the data file.
Users of the Component	Technology Integrators
User Documentation	http://github.com/interTwin-eu/dcnios
Technical Documentation	http://github.com/interTwin-eu/dcnios
Responsible	GRyCAP-I3M-UPV - products@grycap.upv.es Contact point: Germán Moltó - gmolto@dsic.upv.es
License	Apache 2.0
Source code	https://github.com/grycap/dcnios

[3] https://nifi.apache.org/

3.1.3 openEO

Component name	openEO
Description	openEO is an API that supports i) the management of workflows, ii) job handling, and iii) linking to data sources and processing capabilities on compatible cloud platform providers in a standardised way.
Value proposition	openEO can be extended to support execution of containerized software packages using the OSCAR component. This integration allows for seamless execution of containers in the cloud or on HPC directly from an openEO process graph.
Users of the Component	DT Developers
User Documentation	**https://openeo.org/documentation/1.0/**
Technical Documentation	**https://openeo.org/documentation/1.0/developers/**
Responsible	Eurac Research, EODC, Münster University

3.1.4 Ophidia

Component name	Ophidia/PyOphidia
Description	PyOphidia provides the Python bindings for Ophidia, a High-Performance Data Analytics framework
Value proposition	Ophidia framework is an open-source solution for the analysis of scientific multi-dimensional data, joining HPC paradigms and Big Data approaches. It provides an environment targeting High Performance Data Analytics through parallel and in-memory data processing, data-driven task scheduling and server-side analysis. The framework supports the execution of complex analytics workflows in the form of DAGs of Ophidia operators. Integration with other Python libraries and tools (e.g., Jupyter notebooks) is supported through the PyOphidia module.

Users of the Component	Data scientists, DT developers
User Documentation	**https://pyophidia.readthedocs.io/en/latest/**
Technical Documentation	**https://pyophidia.readthedocs.io/en/latest/installation.html**
Responsible	CMCC, Ophidia dev team
License	GPLv3
Source code	PyOphidia: **https://github.com/OphidiaBigData/PyOphidia** Ophidia: **https://github.com/OphidiaBigData**

3.1.5 yProv

Component name	yProv
Description	yProv is an open-source software ecosystem to support provenance management within scientific workflows. It relies on the W3C PROV family of standards, a RESTful interface and a graph database back-end based on Neo4J. The yProv web service (main component) is implemented in Python by using the Flask micro-framework which is based on the Jinja2 Template Engine and Werkzeug WSGI Toolkit. The service is domain-agnostic, though its primary case studies in the project come from the climate change domain. The service aims at implementing the multi-level provenance concept, to navigate within the provenance space across different dimensions (e.g., horizontal & vertical). yProv includes also the Command Line Interface and additionally, it delivers support for provenance tracking in AI and Workflows, which adds extra capabilities in key and recurring use cases across different DTs.
Value proposition	Users can exploit the yProv service to manage (i.e. store, retrieve, explore, visualise) the provenance of information associated with scientific datasets, thus getting a better understanding about specific datasets. The value proposition is about (i) stronger traceability, transparency, and trust (through a richer set of metadata) and (ii) multi-level exploration/navigation of provenance metadata information.

Users of the Component	Scientific users, both producers and consumers of datasets. End users can interact via the yProv RESTful API to manage (i.e., Create Read Update and Delete (CRUD) operations) the provenance information.
User Documentation	**yProv service:** https://github.com/HPCI-Lab/yProv/blob/main/README.md **yProv CLI:** https://github.com/HPCI-Lab/yProv-CLI/blob/main/README.md **yProv4ML:** https://github.com/HPCI-Lab/yProvML https://hpci-lab.github.io/yProv4ML.github.io/index.html **yProv4WFs:** https://github.com/HPCI-Lab/yProv4WFs/blob/main/README.md **yProv4SQA:** https://github.com/HPCI-Lab/yProv4SQA/blob/main/README.md
Technical Documentation	**yProv service:** https://github.com/HPCI-Lab/yProv/blob/main/README.md **yProv CLI:** https://github.com/HPCI-Lab/yProv-CLI/blob/main/README.md **yProv4ML:** https://github.com/HPCI-Lab/yProvML https://hpci-lab.github.io/yProv4ML.github.io/index.html **yProv4WFs:** https://github.com/HPCI-Lab/yProv4WFs/blob/main/README.md **yProv4SQA:** https://github.com/HPCI-Lab/yProv4SQA/blob/main/README.md
Responsible	University of Trento
License	GPLv3
Source code	**yProv:** https://github.com/HPCI-Lab/yProv/ **yProv-CLI:** https://github.com/HPCI-Lab/yProv-CLI **yProvML:** https://github.com/HPCI-Lab/yProvML **yProv4WFs:** https://github.com/HPCI-Lab/yProv4WFs **yProv4SQA:** https://github.com/HPCI-Lab/yProv4SQA

3.1.6 Functionalities developed since the last release

OSCAR has been used as a generic framework for real-time data acquisition and processing that builds on event-triggered execution of workflows, to i) improve response times and minimise data transfers, and to ii) support new event sources. For this release:

- Rucio, a scientific data management system, has been added both as data storage and event source; its configuration is managed via DCNiOS (see DCNiOS) that supports the creation of Apache NiFi dataflows, now including a ProcessGroup for Rucio integration.

yProv has been extended in this release to better support provenance tracking in scientific workflows and Software Quality Assurance (SQA) processes.
To this end, the following modules:

- yProv4WFs: has been extended with the openEO support, in close collaboration with EODC, TUWien and Eurac Research to better support environmental digital twins based on openEO orchestration. The extension has been also documented in a paper[4] by Omidi et al., accepted at the eScience2025 conference;
- yProv4SQA: has been developed to address traceability in SQA processes. The library has been integrated with the SQAaaS platform (in collaboration with CSIC) as well as GitHub.

Finally, in this release the core yProv service has been further improved in terms of robustness by fixing minor bugs in the code.

Developments in PyOphidia and Ophidia framework, in such release, include the final extensions to support the workflows needed by the DT applications from T4.5. In particular:

- PyOphidia: (i) the support of CWL has been extended to additional Ophidia operators and (ii) the validation capabilities have been further improved to control the workflow structures used in the DT applications.
- Ophidia: (i) the computation of new operations (derivatives, mathematical expressions) and (ii) new aggregation levels have been added to the framework.

Moreover, a set of minor bugs has also been fixed and the documentation has been updated.

3.1.7 Integrations with other DTE components

OSCAR and DCNiOS have been integrated with several DTE components from the interTwin project:

- An OSCAR cluster can be provisioned using the **Infrastructure Manager** Dashboard, which provides a streamlined process for deploying the cluster on any Cloud that you have access to.
- DCNiOS is a tool that can be executed locally to deploy Apache NiFi dataflows. Apache NiFi can also be deployed using the IM.

[4] https://arxiv.org/abs/2506.08597

- The tool DCNiOS allows OSCAR to expand the supported event sources for deploying scientific workflows by creating Apache NiFi dataflows that connect the data source with the OSCAR services that compound the workflow.
- The integration of OSCAR with interLink (WP5) allows OSCAR to offload jobs to any remote resource capable of managing a Container execution lifecycle. Specifically, interLink deploys a Virtual Node in the OSCAR cluster; when OSCAR assigns a pod to that node, interLink translates the pod into a remote call, allowing the execution of the container in an HPC cluster.
- The integration between OSCAR and **Itwinai**, a Python library that streamlines AI workflows while reducing engineering complexity, was performed by containerising itwinai and deploying it as an OSCAR Service, which allowed the creation of event-driven AI pipelines.
- The integration of OSCAR with Rucio (WP5), a storage solution for large-scale scientific data, through DCNiOS allows users to employ Rucio as data storage for their event-driven scientific workflows based on OSCAR Services.
- The integration between OSCAR and **openEO**, enables users to interact with OSCAR directly through the openEO API, adding a new interface in addition to the OSCAR API, the OSCAR-CLI, the OSCAR-Python library, and the web interface.
- The integration with the **SQAaaS platform** platform enables OSCAR and DCNiOS to automatically receive quality assessments for each release, facilitated by a GitHub action on every pull request.

During the project, yProv has been integrated with several interTwin components, namely:

- IM: to enable automated deployment of yProv in cloud settings;
- itwinai: to provide provenance tracking in ML training processes, via itwinai "logger" component;
- SQAaaS platform: to integrate traceability within SQA processes;
- openEO: to generate a provenance-based representation of a workflow execution;
- Ophidia: to integrate provenance-tracking within climate analytics workflows;

PyOphidia has been integrated with different interTwin components, such as:

- The module now supports the generation of provenance documents in W3C PROV which can be directly stored and explored via yProv.
- PyOphidia is used by two thematic modules from WP7, namely ML TC Detection and ML4Fires for supporting the workflows of the DT applications.
- The extended validation capabilities have been also integrated with the SQAaaS platform (see **3.2**).
- Docker images, based on the itwinai image, have been designed and built as part of the Jupyter-based Ophidia templates (see **3.4**).

3.1.8 Integrations with DT Applications

OSCAR and DCNiOS have been used in the following DT applications:

- OSCAR has been employed to automatically run an AI pipeline containing inference of the model 3DGAN developed by CERN.
- DCNiOS has been employed to configure the data source dCache that was used to store the needed data for the inference of the 3DGAN model.
- A Jupyter Notebook deployed inside OSCAR was employed in the development of the FloodAdapt Notebooks by Deltares.
- OSCAR was employed as the running platform of the following models developed by Deltares:
    - SFINCS, simulation of floods model,
    - WFLOW, a hydrological processes simulation model,
    - RA2CE, resilience assessment and adaptation decision-making model.
- OSCAR was employed as the running platform for the HydroMT and WFLOW model by Eurac Research.

PyOphidia/Ophidia has been used for developing CWL-based workflows for the Tropical Cyclones DT application (T4.5) as well as the Wildfires DT application (T4.5). Moreover, the provenance of such workflows can be also tracked.
yProv has been integrated via yProv4ML API within the Tropical Cyclones, Eddies and Wildfires DT applications (T4.5) and Detector Simulation DT Application (T4.2), to track the provenance during the execution of the related training processes.

3.1.9 Pilots and testing activities

OSCAR and DCNiOS have been employed in the following pilots and testing activities:

- Event-driven 3DGAN inference pipeline for High Luminosity Large Hydron Collider (HL-LHC) particle simulations using containerised itwinai, triggered via dCache/DCNiOS, with OSCAR orchestrating model deployment and inference on HPC clusters through interLink.
- OSCAR executed key steps of a CWL-based workflow integrating Deltares models: SFINCS, WFLOW, and RA2CE, for flood simulation and risk assessment, by automating service creation, data upload, execution triggering, and results retrieval within the CWL pipeline.
- Deployed a Jupyter Notebook within the OSCAR environment to create and invoke OSCAR services, enabling execution of the HydroMT model deployed by Eurac Research for drought early warning in the Alps.
- Tested OSCAR integration via the openEO API by verifying cluster status, provisioning the HydroMT service if absent, and invoking the service, demonstrating initial interoperability and workflow feasibility.
- Tested Rucio integration with OSCAR by creating a dataflow that listens to RabbitMQ notifications, triggering an OSCAR service to download datasets from Rucio, process them, and upload the results back into Rucio.

As mentioned above, specific Docker images with PyOphidia/Ophidia have been created and published on DockerHub and have been tested on the interTwin testbed at Vega. Such images are used for the Ophidia/Jupyter TOSCA template from T6.4.

Activities about the integration of yProv within environmental DTs (i.e., TC, eddies and wildfires) have been carried out on the VEGA testbed in close collaboration with CMCC.

3.2 Task 6.2

3.2.1 SQAaaS platform

Component name	SQAaaS **https://sqaaas.eosc-synergy.eu**
Description	Platform for quality assessment and awarding of multiple digital objects (source code, services, data)
Value proposition	Provide a module for quality validation within the interTwin's DTE
Users of the Component	DTE developers & users
User Documentation	**https://docs.sqaaas.eosc-synergy.eu** **https://indigo-dc.github.io/jenkins-pipeline-library**
Technical Documentation	**https://github.com/eosc-synergy/sqaaas-api-spec**
Responsible	CSIC, LIP, UPV
License	GPL-3.0-only
Source code	**https://github.com/eosc-synergy/sqaaas-api-server** **https://github.com/eosc-synergy/sqaaas-web** **https://github.com/indigo-dc/jenkins-pipeline-library**

3.2.2 SQAaaS GitHub Actions

Component name	SQAaaS assessment GitHub Action
Description	Trigger SQAaaS quality assessment service from GitHub actions
Value proposition	Integrate interTwin's GitHub organisation with the SQAaaS platform for software quality (incl. workflow and model code)
Users of the Component	DTE developers & users

User Documentation	https://github.com/EOSC-synergy/sqaaas-assessment-action https://github.com/EOSC-synergy/sqaaas-step-action
Technical Documentation	https://github.com/eosc-synergy/sqaaas-gh-action https://github.com/EOSC-synergy/sqaaas-step-action
Responsible	CSIC
License	GPL-3.0-only
Source code	https://github.com/EOSC-synergy/sqaaas-assessment-action https://github.com/EOSC-synergy/sqaaas-step-action

3.2.3 Ophidia Workflow Validation Tool

Component name	Ophidia Workflow Validation tool
Description	SQAaaS tool to validate the Ophidia workflow.
Value proposition	Integrate interTwin's workflows within pipeline steps, creating assessments, getting outputs, checking pipeline status
Users of the Component	DTE developers & users
User Documentation	https://github.com/EOSC-synergy/sqaaas-tooling/tree/feature/Ophidia/QC.Sty/pyophidia
Technical Documentation	https://github.com/EOSC-synergy/sqaaas-tooling/tree/feature/Ophidia/QC.Sty/pyophidia
Responsible	CSIC
License	GPL-3.0-only
Source code	https://github.com/EOSC-synergy/sqaaas-tooling/tree/feature/Ophidia/QC.Sty/pyophidia

3.2.4 Ophidia Workflow Reporting Plugin

Component name	Ophidia Workflow Reporting Plugin
Description	SQAaaS reporting plugin for Ophidia workflow validation.
Value proposition	This tool helps users identify any errors or issues in their workflows, allowing them to make necessary corrections and improvements.
Users of the Component	DTE developers & users
User Documentation	https://github.com/EOSC-synergy/sqaaas-reporting-plugins/tree/feature/pyophidia/qc_sty_Ophidia
Technical Documentation	https://github.com/EOSC-synergy/sqaaas-reporting-plugins/tree/feature/pyophidia/qc_sty_Ophidia
Responsible	CSIC
License	GPL-3.0-only
Source code	https://github.com/EOSC-synergy/sqaaas-reporting-plugins/tree/feature/pyophidia/qc_sty_Ophidia

3.2.5 Functionalities developed since the last release

In the SQAaaS platform, we have added new tools for the user, including the capability to perform partial evaluations by evaluating only specific sets of tests. The benefits of this option are manifold.

First, it allows users to save time and resources when conducting evaluations, which is particularly helpful for large repositories that may take a significant amount of time to evaluate. Additionally, it enables specific tests to be conducted outside of the general tests, such as evaluations tailored to particular repositories, like the validation of Ophidia workflows, which are only relevant for niche repositories.

We have also extended this option to Github actions by creating actions that enable the evaluation of specific SQAaaS tests without evaluating the entire repository.

3.2.6 Integrations with other DTE components

An Ophidia plugin has been created that allows for the validation of all the Ophidia (WP6) workflow files in a repository/folder. The objective is to synergize with the upgrades to GitHub actions so whenever a file is pushed to the repository, just the Ophidia workflow evaluation is triggered and the user can know if their workflow follows the Ophidia schema. Such validation features work both on JavaScript Object Notation (JSON) and CWL documents.

3.2.7 SQAaaS adoption by workflow implementations

SQAaaS CLI component development was not prioritized for the integration goals. In the initial review of the project for the workflow interaction with the SQAaaS API, we foresaw that the CLI tools would help with the integration of the custom assessment. However, during the project, we realized that the workflow tools would integrate better using the SQAaaS API directly. Therefore, the effort was more directly focused on the workflow-scoped implementation rather than the command-line execution. As a result, the SQAaaS CLI project does not have a release.
For the Ophidia workflow, the integration with the SQAaaS platform adds a new tool called Ophidia Workflow Validation tool (section 3.2.3) and the corresponding reporting plugin (section 3.2.4). Both tools work together to perform the workflow validation and provide the report.

3.2.8 Integrations with DT Applications

SQAaaS has been used by all the developers of DTE Thematic modules to improve the quality of the components which are then integrated directly by the DT Applications. Among the components, 3 have gained quality badges.

3.2.9 Pilots and testing activities

Nothing to report

3.3 Task 6.3

3.3.1 openEO

Component name	openEO
Description	openEO is an API, supporting management of workflows and handling of jobs, as well as linking to available data sources and processing capabilities in a harmonised way – front-end users don't need knowledge on the data structure.
Value proposition	The openEO syntax is already used for several specific predefined processes, dealing with fusion of raster data. Those use cases can be further extended for integrating vector data and possibly data coming as output from different models from both the physical and data driven domain as well as preparing data in a harmonised way for ingestion into such models.
Users of the Component	DT Developers

User Documentation	https://openeo.org/documentation/1.0/
Technical Documentation	https://openeo.org/documentation/1.0/developers/ , https://processes.openeo.org/
Responsible	Eurac Research, EODC, Münster University

3.3.2 Functionalities developed since the last release

Since the last release, we have been working on integrating the OSCAR component to allow container execution directly from openEO. A new process called run_oscar has been defined. The implementation involves integrating the OSCAR Client library into openeo-processes-dask. This work is ongoing but will be completed by the end of the project.

3.3.3 Integrations with other DTE components

Other than OSCAR integration, openEO has also been integrated with the DownscaleML component to provide downscaling capabilities in the alpine drought use case. Additionally, the openEO service TOSCA template has been developed in the context of Task 6.4 for automatic deployment over cloud backends via the Infrastructure Manager service.

3.3.4 Integrations with DT Applications

openEO integration with other interTwin components is driven by the Alpine drought forecasting use case (T4.6). This use case combines many components and can be executed from start to end using openEO. openEO has been integrated in the context of the Post Flood analysis DT (T4.7) as well.

3.3.5 Pilots and testing activities

An openEO backend has been deployed in the interTwin infrastructure (in particular at GRNET cloud) using a TOSCA template via the IM service (T6.4). We have utilized the Alpine drought forecasting use case containers to validate the run_oscar process implementation.

3.4 Task 6.4

3.4.1 Infrastructure Manager

Component name	Infrastructure Manager (IM)
Description	The Infrastructure Manager (IM) is an open-source Infrastructure as Code (IaC) tool that provides both an XML-RPC and REST API to receive TOSCA-based virtual infrastructure provision requests for their deployment on IaaS cloud back-ends. These requests may

	come from a web-based graphical user interface such as the IM-Dashboard, through the IM CLI or via an HTTP-based client.
Value proposition	Users can self-provision any kind of complex virtual infrastructure on whichever cloud infrastructure they can access.
Users of the Component	Scientific users, requiring Big Data Analytics tools to be deployed on different Cloud back-ends.
User Documentation	**https://imdocs.readthedocs.io/en/latest/**
Technical Documentation	**https://imdocs.readthedocs.io/en/latest/**
Responsible	GRyCAP-I3M-UPV
License	GPL 3.0
Source code	**https://github.com/grycap/im**

3.4.2 Big Data Analytics TOSCA templates

Component name	Big Data Analytics TOSCA templates
Description	As a set of TOSCA templates to deploy Big Data Analytics tools
Value proposition	TOSCA templates enable the description, in a cloud-agnostic way, of the virtual infrastructures needed in the available Big Data Analytics tools.
Users of the Component	TOSCA template developers.
User Documentation	**https://confluence.egi.eu/display/interTwin/TOSCA+Templates**
Technical Documentation	**https://confluence.egi.eu/display/interTwin/TOSCA+Templates**
Responsible	UPV
License	Apache 2.0

Source code	**https://github.com/grycap/tosca/tree/main/templates**

3.4.3 Configuration artefacts

Component name	Configuration artefacts
Description	Ansible playbooks and roles and any other artefact needed in the deployment of the selected Big Data Analytics tools. Those Ansible playbooks are used by the TOSCA templates defined in Section **3.4.2** in order to perform the actual deployments on cloud nodes.
Value proposition	TOSCA templates refer to these artefacts to enable the configuration of the Big Data Analytics tools using the Ansible tool.
Users of the Component	TOSCA template/Ansible playbooks developers.
User Documentation	-
Technical Documentation	-
Responsible	UPV
License	Apache 2.0
Source code	**https://github.com/grycap/tosca/tree/main/artifacts**

3.4.4 Functionalities developed since the last release

There are no new templates or configuration artefacts since the last release. Regarding the IM, only minor maintenance fixes have been made.

3.4.5 Integrations with other DTE components

The IM is integrated by the AI based Orchestrator from WP5 in order to deploy TOSCA templates on selected Cloud backends.
In addition, a set of templates and their associated configuration artefacts has been created to enable the automated deployment of the following DTE components:

- **openEO**
- **yProv**
- **Ophidia**
- **OSCAR**

Furthermore, the "**MLFlow + Auth GUI**" template has been used by itwinai component in several deployments.

The IM has used the SQAaaS GitHub Actions to assess the quality of the IM code.

3.4.6 Integrations with DT Applications

The Kubeflow TOSCA template was used in collaboration with itwinai (T6.5) for Particle detector simulation (CERN, WP4). A demo[5] has been produced as well and can be viewed online.
EOEPCA Zoo Project-DRU tool was tested by the DT Application (T4.6, Drought Early Warning), however, after extensive testing the DT application focused on integrating openEO with OSCAR.
The Ophidia template, and in particular the Docker images, has been used from the JupyterHub testbed of the project for the DT Applications developed by T4.5 and finally the MLFlow template has been used by all the DT Applications based on AI/ML.

3.4.7 Pilots and testing activities

STAC, OpenEO and MLFlow services have been deployed in the GRNET Cloud and they have been actively used by the use cases in WP4.
The Kubeflow template has been used to deploy an instance on the JSC cloud platform.

3.5 Task 6.5 – Advanced AI workflows and methods lifecycle

3.5.1 Itwinai

Component name	itwinai
Description	itwinai is a Python library that streamlines AI workflows, while reducing engineering complexity. It seamlessly integrates with HPC resources, making workflows highly scalable and promoting code reuse. itwinai empowers AI researchers with built-in tools for hyperparameter optimization, distributed machine learning, tracking of ML metadata, and management of ML models registry. It integrates smoothly with Jupyter-like GUIs, enhancing accessibility and usability. It also supports containerized execution of the library, allowing easy deployment on both cloud and HPC.

[5] interTwin demo: itwinai (WP6) - https://youtu.be/NoVCfSxwtX0

Value proposition	Different interfaces are provided to lower the entry barrier for users coming from different fields of expertise: Python API and command line interface. Moreover, itwinai integrates with high-level GUIs provided by third-party tools such as Jupyter. itwinai provides out-of-the-box state of the art AI tools, such as MLflow-based models repository. It encourages code reuse, to further simplify and streamline development of ML workflows, on top of seamless integration with HPC and cloud resources.
Users of the Component	DT developers, ML engineers and researchers.
User Documentation	**https://itwinai.readthedocs.io/**
Technical Documentation	**https://itwinai.readthedocs.io/**
Responsible	CERN, FZ Juelich
License	Apache 2.0
Source code	**https://github.com/interTwin-eu/itwinai**

3.5.2 Functionalities developed since the last release

Code optimization and bottleneck detection via profiling
The itwinai library has been extended with integrated support for an established Python profiling tool, enabling users to systematically analyze the performance of their training code, even in a distributed setting. The profiling functionality highlights which parts of the training code are most time-consuming, making it easier to identify performance bottlenecks. Additionally, itwinai now includes automatic aggregation and summarization of profiling results. The output is presented as a clear, time-sorted table, improving interpretability and significantly reducing the effort required for performance analysis.

Integration with LUMI HPC
Another substantial effort in the last part of the project has been the integration of itwinai with Large Unified Modern Infrastructure (LUMI[6]) HPC supercomputer, in particular on the LUMI-G partition. This system is different from the ones on which we previously worked during the project, as its compute nodes provide 4 dual-die MI250X AMD GPUs, AMD Trento CPUs and Slingshot interconnect. LUMI has the same hardware setup of Frontier[7]. The successful integration with LUMI marks an important step towards cross-

[6] https://lumi-supercomputer.eu/
[7] https://www.olcf.ornl.gov/frontier/

platform support in itwinai, automatically enabling the integration of the interTwin use cases on such systems.

Integration with JUPITER Booster

An initial integration with the JUPITER Booster system at JSC, the first Exascale system in Europe, has been performed. This system is equipped with the Grace Hopper GH 200 Superchip, where NVIDIA's first CPU Grace and the Hopper GPU are deployed in the same chip. This integration provides the first step towards scaling itwinai on an Exascale system. In the next months, detailed performance benchmarks will be performed.

Improved scalability report

When working on HPC and applying for compute time it is often important for the users to analyze and demonstrate the scalability of their code. The scalability report feature provided by itwinai accounts for this as it gives the user visual and quantitative insight into the scalability of their distributed training runs and potential inefficiencies, by comparing relative computation time for GPU counts per strategy.
During the last iteration, the existing scalability report feature has been refined for greater consistency across GPU architectures (e.g., AMD and NVIDIA). It now allows users to more reliably compare the proportion of total time spent on computation versus other activities (notably communication) across multiple runs and scales. Given the Torch Profiler's limitations, such as the lack of detailed information on overlapping operations and type of calls, which vary between device types, the revised report adopts a more conservative approach: computation time is compared directly to overall run time. The plots have been redesigned for clarity, with improved labelling of call types (e.g. computation, see **Figure 1**), which catches AMD calls too, and a more intuitive structure that shows scalability over GPU counts per strategy, rather than directly comparing strategies, which is not meaningful given current data constraints. The previous method of comparing computation and communication times remains available but is now deprecated due to its lack of precision.

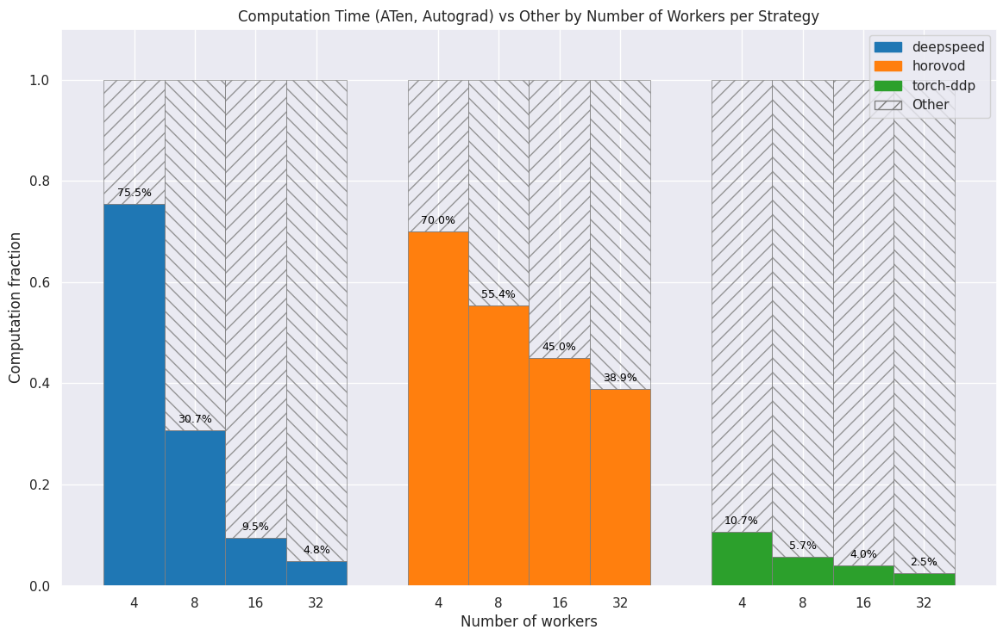

Figure 1 Example of improved scalability analysis plot focusing on summarizing the fraction of computation operations on GPU versus other operations (e.g., communication, memory operations). In this plot we can see the comparison between three distributed ML str

CI/CD for containers integrating HPC in GitHub

An important aspect of itwinai is the focus on user experience and support. To this end, it is important for us to provide our users, represented by the interTwin scientific use cases, with container images ready to be deployed on HPC. Such images provide all the dependencies necessary to run distributed AI workloads on HPC, including both distributed AI training and hyperparameter optimization.

The process of building and testing containers is repetitive and time-consuming, therefore, we developed a novel CI/CD pipeline based on Dagger[8] to automate the creation and tests on HPC of itwinai containers. A new container image is built and automatically tested on HPC every time something is pushed to the main branch of the itwinai repository. The integration of HPC into GitHub is made possible by interLink[9], a component developed in the interTwin project.

Distributed inference

In the final part of the project, the inference class of the itwinai core library has also been extended to enable users to run inference in a distributed setting. The implementation reuses many definitions from the distributed trainer class, while extending the inference class from the serial case. This is expected to allow inference to be useful for future Large Language Model (LLM) based inference routines, where distributed resources are essential.

[8] https://dagger.io/

[9] https://www.intertwin.eu/article/infrastructure-component-interlink

3.5.3 Integrations with other DTE components

In this section we summarize the integrations performed with other DTE components throughout this project:

- **SQAaaS**: the software quality assessment component has been integrated since the first part of the project in our GitHub CI workflows, providing quality badges encompassing both software quality and unit/integration tests.
- **OSCAR**: integration with OSCAR allowed integrating itwinai in a workflow for event-driven real-time data processing, which was demonstrated with the detector simulation use case in high-energy physics.
- **interLink**: on multiple occasions, itwinai containers have been transparently offloaded from Kubernetes to remote HPC resources through interLink, including both single-container jobs and more complex jobs involving the offloading of multiple workers – an example is the offloading of a Ray cluster initially instantiated on Kubernetes via the KubeRay operator.
- **IM and TOSCA**: the use of IM and TOSCA templates enabled the user-friendly deployment of AI cloud services (e.g., on OpenStack resources on JSC cloud). Relevant examples of AI cloud services deployed during the project to support testing activities and interTwin use cases include Kubeflow[10] and MLflow[11] tracking server (serving also as an AI models repository).
- **yProvML[12]:** the yProvML service was integrated in itwinai to support the users with advanced capabilities for provenance tracking in AI workflows.

3.5.4 Integrations with DT Applications

The integration with other DT applications has been one of the primary activities of this task and, after multiple iterations, the use case code integration has produced a series of itwinai plugins. Each plugin constitutes the fundamental unit of code supporting AI workflows for each scientific use case – each plugin enables easy distribution and reuse of best practices and tools developed by the use cases. Below you can find a summary of the itwinai plugins currently maintained by the use cases.

Physics Sciences

- **3DGAN – Fast Simulation of Particles in Calorimeters**
- **GlitchFlow – Noise Generation for Gravitational Waves Analysis at Virgo**
- **Pulsar Detection (Radio Astronomy)**

[10] https://www.kubeflow.org/
[11] https://mlflow.org/
[12] https://github.com/HPCI-Lab/yProvML

- **Normflow - Normalizing flows as a generative model for lattice field theory**

Environmental Sciences

- **Hython – Hydrological Modelling for Drought Early Warnings**
- **xtclim – ML-based extreme events detection and characterization (CERFACS)**

The full list of (currently) existing itwinai plugins can be found **here**.

In addition to the use cases integration mentioned above, there have been integration activities with the environmental use cases for wildfires and tropical cyclones detection, developed by CMCC (T4.5). In this case, the work focused on the integration of MLflow and yProvML for advanced ML logging and experiment tracking, including the storage of ML models to a dedicated ML models repository, using the itwinai logger module. This has been made possible by the deployment of a MLflow server on EGI cloud, providing a centralized location for storing and sharing ML models trained by scientific digital twin use cases.

3.5.5 Pilots and testing activities

Pilots and tests mostly focused on the integration with different HPC systems (i.e., Vega, JSC, PSNC, LUMI) and with other DTE components (i.e., interLink, OSCAR, Rucio). In this section we organize the test activities by HPC center, explaining which DTE components were tested on each site.

Vega HPC

Most of the pilot activities have been carried out on Vega[13], on the GPU partition featuring compute nodes with four A100 Nvidia GPUs[14]. On this HPC system we ran a wide variety of tests including distributed machine learning training and hyperparameter optimization, which entails interacting directly with the SLURM scheduler and with the software stack available through software modules.

The first relevant integration test was done with interLink, demonstrating the possibility of transparently offloading AI jobs from Kubernetes (on cloud) to HPC. Since the compute nodes on Vega allow internet connection – a feature not always available on HPC – we were able to persist the results of AI training jobs on a remote MLflow tracking server, deployed on cloud. A successful example is the integration with JupyterHub/JupyterLab,

[13] https://doc.vega.izum.si/
[14] https://doc.vega.izum.si/architecture/

where itwinai container images for JupyterLab were provided to enable Jupyter sessions with itwinai and its dependencies already installed, and their seamless offloading to remote HPC resources (virtually not limited to Vega HPC).

Another relevant test activity foresaw the integration of itwinai with OSCAR, the component for event-driven data ingestion, allowing to trigger AI inference workflows on HPC whenever a new piece of data was uploaded to the dCache storage system. In preparation for this test, OSCAR had been deployed on cloud and the AI inference code had been packaged in a Docker container. Whenever a new piece of data was made available, OSCAR would launch the Function as a Service (FaaS) Supervisor and the inference code, which would in turn be offloaded to HPC via interLink. Once on Vega HPC, the Docker container is automatically converted to Singularity, the Kubernetes pod is translated into a SLURM job script by the interLink sidecar, and the inference workflow is executed. The OSCAR Supervisor, offloaded to HPC, takes care of downloading the input data to the HPC and uploading the results back to the remote dCache instance.

Another test focused on demonstrating the integration of the offloading mechanism provided by interLink with Ray and KubeRay[15] operator. We ran itwinai on the Vega supercomputer by offloading a Ray cluster from Kubernetes to the HPC nodes through InterLink. Both itwinai and its Hython plugin[16] now ship as Docker and Singularity container images, so you can launch them in Kubernetes with KubeRay in just a few commands. The Hython itwinai plugin containers can be used to create a Ray cluster that spans multiple compute nodes, which allowed us to perform distributed hyper-parameter optimization at scale. This workflow illustrates the seamless integration of itwinai with the interTwin ecosystem.
Additional documentation on itwinai is available online[17]

[15] https://github.com/ray-project/kuberay

[16] The itwinai Hython plugin has been developed in collaboration with Eurac Research and represents the final version of the integration between itwinai and the use case targeting early warnings for droughts in alpine regions.

[17] https://itwinai.readthedocs.io/latest/tutorials/distrib-ml/kuberay-setup-tutorial.html#start-the-ray-cluster

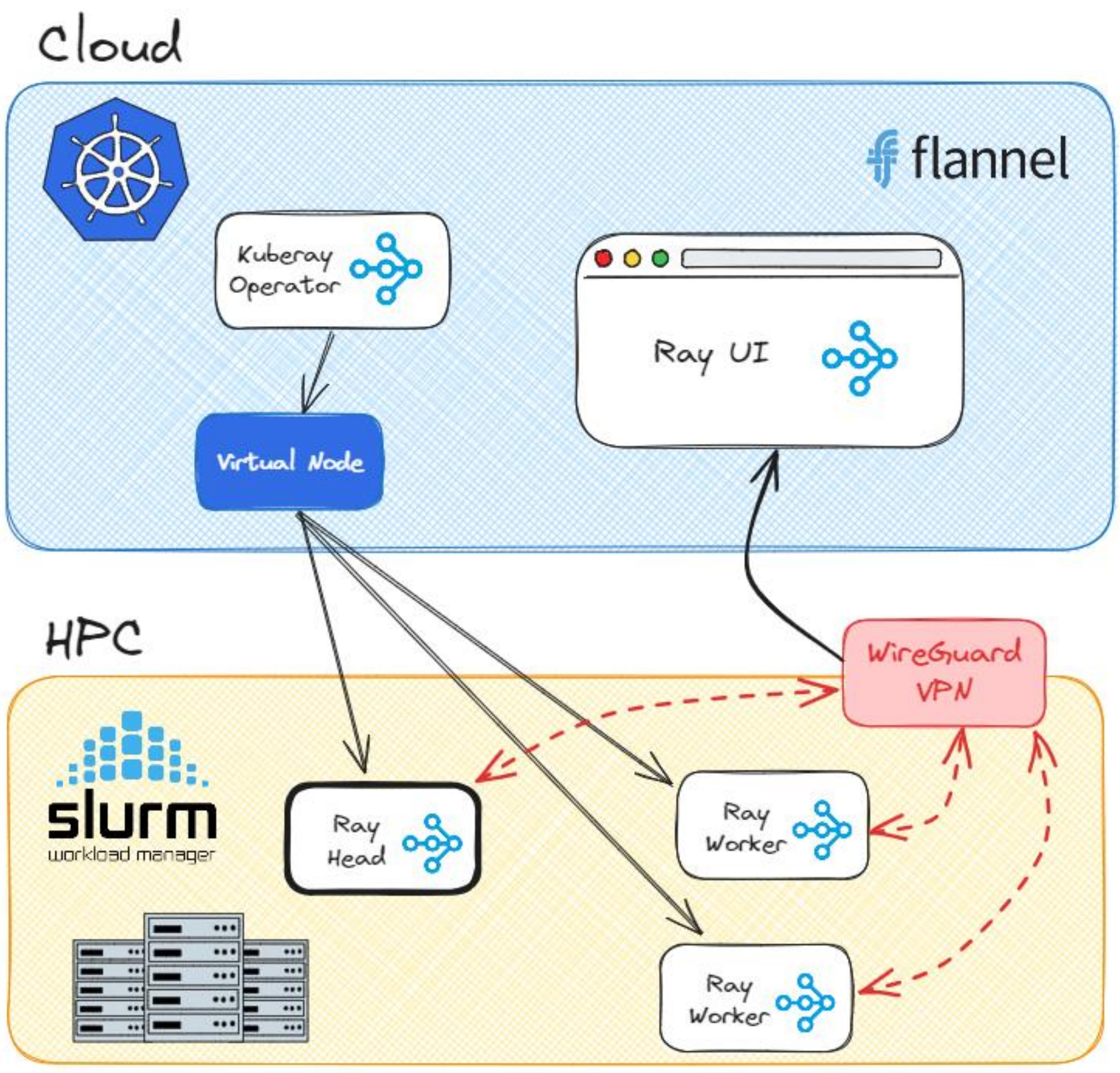

Figure 2 Graphical representation of the integration of KubeRay and interLink on Vega, allowing the offloading to HPC of an entire Ray job

One more integration test on Vega included access to datasets from the Rucio data lake, meant to provide a consistent mechanism for asynchronously replicating large-scale datasets across different computing sites. A notable example of this integration was the access to the datasets needed for AI training of the high-energy physics use case for detector simulation, using three-dimensional Generative Adversarial Networks (GANs).

JSC HPC

The JSC systems HDF-ML (now out of production) and JUWELS Booster[18] have been utilized for many development activities of itwinai. While the HDF-ML system has NVIDIA V100 GPUs, the JUWELS Booster system features NVIDIA A100 GPUs. Many integration activities of use-cases were performed on these systems, including the Thematic modules normflow, PulsarDT, 3DGAN, GlitchFlow, xtclim, and Hython. The integration included

[18] https://apps.fz-juelich.de/jsc/hps/juwels/booster-overview.html (accessed on 23.06.2025)

activities such as distributed training, profiling, and HPO. Apart from this, tests were conducted to evaluate the performance of the models on a large number of GPUs (up to 64 GPUs).

In the initial phase of the project, Kubeflow and Airflow instances were deployed on the JSC Cloud[19] in collaboration with T6.4. The intention here was to test ML pipelines on these platforms. Furthermore, on JSC Cloud, the interLink framework from WP5 was tested. A Virtual Kubelet (also referred to as an edge node) was deployed, which allows external users to access JSC's HPC cluster through Kubernetes-native workflows. This was achieved by running a specialized interLink plugin, based on the UNICORE middleware, on the edge node. When an end user submits a pod creation request to the central Kubernetes cluster, the request is routed to the specified site, in this case, JSC's edge node. The edge node then translates and forwards the request to the UNICORE service for HPC batch job submission.

LUMI HPC

On LUMI we ran some tests on AMD MI250X GPUs. A container image for LUMI was added to the repository showcasing the compatibility of itwinai with AMD GPUs. Another container for the Hython-itwinai-plugin has been developed and was tested for distributed training with all available distributed strategies (torch-ddp, DeepSpeed, Horovod), as well as HPO using ray and its different strategies. The access to LUMI allowed us to make itwinai more cross-platform, in particular concerning the scalability report and the development of container images compatible with Radeon Open Compute (ROCm). The scalability report feature was extended to collect energy consumption and utilization metrics also for AMD GPUs.

PSNC HPC

On PSNC[20] we ran some simple tests on Nvidia H100 GPUs, targeting distributed machine learning training of the high-energy physics use case for detector simulation.

4 Conclusions

Overall, this final release delivers a comprehensive and modular ecosystem that effectively supports complex, event-driven scientific workflows, provenance tracking, and quality assurance in large-scale DT environments.

More in detail, the integrations of the OSCAR framework, yProv provenance system, and the PyOphidia/Ophidia scientific workflow tools, enhance the support for real-time, event-driven data acquisition, processing, and provenance tracking in DT applications. The

[19] https://apps.fz-juelich.de/jsc/hps/jsccloud/index.html (accessed on 23.06.2025)
[20] https://www.psnc.pl/

integration of Rucio as a data storage and event source, managed via DCNiOS and Apache NiFi dataflows, broadens the scope and flexibility of data workflows supported by OSCAR.

The updates to the SQAaaS platform have significantly enhanced its flexibility and efficiency by introducing partial evaluation capabilities. Integration with GitHub Actions further streamlines the testing process by enabling targeted evaluations triggered by repository changes. The widespread adoption of SQAaaS across DTE thematic modules has contributed to improved component quality, as evidenced by multiple components receiving quality badges, underscoring the platform's value in supporting robust software development within the ecosystem.

The improvements to yProv include extended support for openEO orchestration and new modules for Software Quality Assurance. This strengthens provenance tracking across scientific workflows and ML training processes. The extended PyOphidia and Ophidia frameworks provide enhanced CWL workflow support, new computational operations, and improved validation features tailored for DT applications.

Since the last release, the ML-oriented toolkit itwinai has improved performance profiling with integrated tools for easier bottleneck detection and clearer visualization, including distributed training. It expanded support to new HPC systems like LUMI and began integration with the JUPITER Booster Exascale platform[21], enhancing cross-platform compatibility. The scalability report was refined for more consistent and reliable analysis across GPU types. A new CI/CD pipeline automates building and testing of HPC-ready containers via GitHub, improving deployment. Distributed inference now supports data parallelism, with plans for model parallelism.

Scientific plugins for physics and environmental sciences have been developed to promote code reuse. Pilot tests on HPC centres such as Vega confirmed smooth offloading of AI jobs from Kubernetes to HPC, effective data-driven workflow triggering, and scalable hyperparameter optimization with Ray. Overall, these updates boost itwinai's performance, scalability, portability, and usability across diverse HPC and scientific domains.

In conclusion, the seamless integration of the core DTE components with interTwin infrastructure elements and their successful deployment within multiple environmental and scientific DT applications demonstrate their maturity and operational readiness. Pilots and testing activities validate the robustness, scalability, and interoperability of the system, showcasing real-world use cases from climate modelling to high-energy physics simulations.

[21] https://www.fz-juelich.de/en/ias/jsc/jupiter/tech

5 References

Reference	
No	**Description / Link**
R1	**interTwin D6.1 Report on requirements and core modules definition** Isabel Campos, Donatello Elia, Germán Moltó, Ignacio Blanquer, Alexander Zoechbauer, Eric Wulff, Matteo Bunino, Andreas Lintermann, Rakesh Sarma, Pablo Orviz, Alexander Jacob, Sandro Fiore, Miguel Caballer, Bjorn Backeberg, Mariapina Castelli, Levente Farkas, & Andrea Manzi. DOI. **https://doi.org/10.5281/zenodo.10417153**
R2	**interTwin D6.2 First release of the DTE core modules** Isabel Campos, Germán Moltó, Alexander Jacob, Pablo Orviz, Miguel Caballer, Matteo Bunino, Alexander Zoechbauer, Sandro Fiore DOI. **https://doi.org/10.5281/zenodo.10224213**
R3	**interTwin D6.3 Updated report on requirements and core modules definition** Isabel Campos, Donatello Elia, Germán Moltó, Ignacio Blanquer, Estíbaliz Parcero, Alexander Zoechbauer, Eric Wulff, Matteo Bunino, Andreas Lintermann, Rakesh Sarma, Pablo Orviz, Alexander Jacob, Sandro Fiore, Miguel Caballer, Bjorn Backeberg, Mariapina Castelli, Levente Farkas, & Andrea Manzi. DOI. **https://zenodo.org/records/13709618**