

interTwin

D5.2 – First DTE Infrastructure Software Release

Status: FINAL
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 status of the development and integration of all the Digital Twin Engine Infrastructure WP5 software products available with the first release

It focuses on the subsystems that have been developed and selected during the first part of the project in order to implement the data and compute federations, being the main missions of the WP5. A short overview of the architecture and its main pillars is given in order to introduce the components and the related subsystems

Finally, it provides the status and the roadmap toward the integration with providers and other core components from other Work Packages, particularly WP6, the seed of early testbeds and pilot systems.



Document Description			
D5.2 – First DTE Infrastructure software release			
Work Package number 5			
Document type	Deliverable		
Document status	FINAL	Version	1
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	Cyfronet AGH		
Document link	https://documents.egi.eu/document/3946		
DOI	https://doi.org/10.5281/zenodo.10224150		
Author(s)	<ul style="list-style-type: none"> • Daniele Spiga (INFN) • Paul Millar (DESY) • Liam Atherton (STFC UKRI) • Marica Antonacci (INFN) • Diego Ciangottini (INFN) 		
Reviewers	<ul style="list-style-type: none"> • Christian Pagé (CERFACS) • Alexander Zochbauer (CERN) 		
Moderated by:	<ul style="list-style-type: none"> • Sjomara Specht (EGI) 		
Approved by	AMB		

Revision History			
Version	Date	Description	Contributors
V0.1	01/10/2023	ToC	Daniele Spiga (INFN)
V0.2	1/11/2023	Intro, abstract, and summary	Daniele Spiga (INFN)
V0.3	13/11/2023	First Full Draft	Daniele Spiga (INFN) Paul Millar (DESY) Liam Atherton (STFC UKRI) Marica Antonacci (INFN) Diego Ciangottini (INFN)
V0.4	21/11/2023	Reviewed version	Christian Pagé (CERFACS), Alexander Zochbauer (CERN)
V0.5	23/11/2023	Comments addressed	All Task Leaders
V1.0	24/11/2023	Final	

Terminology / Acronyms	
Term/Acronym	Definition
DTE	The Digital Twin Engine developed by interTwin
FTS	The File Transfer Service. A software component maintained by a development team at CERN.
Rucio	A third-generation data management software component, maintained by the Rucio development team.
gfal	Grid File Access Library; software that provides an abstraction for POSIX-like access.
CVMFS	The CernVM File System provides a scalable, reliable and low-maintenance software distribution service.
K8s	Kubernetes. Container Orchestration Technology
LHC	Large Hadron Collider. A large scientific facility, located in Geneva, that supports high-energy particle physics research.
OIDC	OpenID-Connect. OpenID is an open standard and decentralised authentication protocol promoted by the non-profit OpenID Foundation. OpenID-Connect is the third generation of OpenID technology; a widely adopted solution that provides an authentication layer on top of the OAuth 2.0 authorization framework.
QoS	Quality of Service
NVMe	Non-volatile memory express is a new storage access and transport protocol for flash and next-generation solid-state drives (SSDs)
scp	Secure copy protocol (SCP) is a means of securely transferring computer files between a local host and a remote host or between two remote hosts.



D5.2 – First DTE Infrastructure Software Release

UFTP	UNICORE FTP is a file transfer tool similar to Unix'
WLCG	The Worldwide LHC Computing Grid is a collaboration of resource (storage and computing) providers that collectively support key research. Although originally conceived to support analysis of LHC data, the collaboration has broadened to support related fields of scientific research.

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



Table of Contents

Introduction	8
1.1 Scope	8
1.2 Document Structure.....	8
2 WP5 Architecture	9
3 Components.....	11
3.1 Federated Compute	11
3.1.1 Release Notes.....	12
3.1.2 Future Plans.....	14
3.2 Federated Data Infrastructure	14
3.2.1 Release Notes.....	18
3.2.2 Future plans	18
3.3 Intelligent Providers Orchestration.....	19
3.3.1 AI Based Orchestrator	19
3.3.2 Release Notes.....	20
3.3.3 Future Plans.....	20
3.4 Resource Accounting System.....	21
3.4.1 Release notes	22
3.4.2 Future plans	22
4 Conclusions	23

Table of Figures

<i>Figure 1 - System landscape diagram (in the C4 model) of the DTE Infrastructure. This is a high-level view that highlights the user interactions with WP5.</i>	<i>10</i>
--	-----------

Table of Tables

<i>Table 1 – Current status of the calls.....</i>	<i>13</i>
---	-----------



Executive summary

The objective of the deliverable D5.2 is to provide a comprehensive collection of information of all the software components that belong to the first release of the interTwin Digital Twins Engine infrastructure.

For each software component, the document includes a list of supported features along with future plans

It has been composed by experts working on the various pillars of the architecture with a deep knowledge of the adopted services.



Introduction

1.1 Scope

This deliverable summarises the status of development of all the components that compose the architecture of the Digital Twin Engine infrastructure developed by the interTwin project.

1.2 Document Structure

This document is organised in two main sections. [Section 2](#) provides a recap of the technical design of the architecture for the Digital Twin Engine infrastructure with the aim to highlight the principal components that will be discussed in [Section 3](#). The latter introduces the list of core services under development and provides details on their functionalities, release notes and future plans.

[Section 4](#) summarises the status of integration of WP5 components with other core components mainly from WP6. It also provides an overview of the current status of the testbeds and pilot deployment.



2 WP5 Architecture

The architecture of the DTE Infrastructure has 4 main pillars as shown in **Figure 1**. The Artificial Intelligence (AI) based orchestration together with the Federated Compute module are key components of WP5 designed to complement each other's features. The resource orchestration is aware of various cloud resource providers and possesses the capability to determine, at any given point in time, the 'best provider' for supporting the deployment of a service defined by WP6. After deploying a high-level service, the objective is to enable a Compute Federation in a heterogeneous environment. This, in turn, means fully exploiting any available resource, including possibly specialised hardware, to cater to specific workflows such as high performance instead of high throughput (HPC vs HTC). The interTwin WP5 proposes the offloading mechanism as a potential solution for this challenge. The purpose of offloading is to facilitate a seamless extension of services to a remote provider. In practice, this means that the offloading mechanism enables the distribution of a unit of work (i.e. a job, a JupyterLab, a function in a serverless system, or a containerized application) to various locations. The decision to adopt a container-based approach is fully compliant with requirements and discussions made with WP6.

Science does not only require access to computing resources in a distributed environment but also to effectively process data exploiting a distributed federation of storages. This is another important component of WP5. In practice we need to guarantee that scientific data required by Digital Twins are accessible by the service as well as by the offloaded process. This necessitates the implementation of components for effective data handling, management, and storage federation. The latter involves dealing with diverse backends both in terms of technology and policy, scattered across different locations. Consequently, we must develop software that implements abstraction layers to allow an easy interaction with a complex storage topology.

Finally, the last pillar of the DTE infrastructure is the resource accounting system, which plays a crucial role in ensuring fair allocation and utilisation of resources in a heterogeneous distributed environment. For this purpose, a centralised resource accounting system is necessary to aggregate usage records from across the infrastructure.

D5.2 – First DTE Infrastructure Software Release

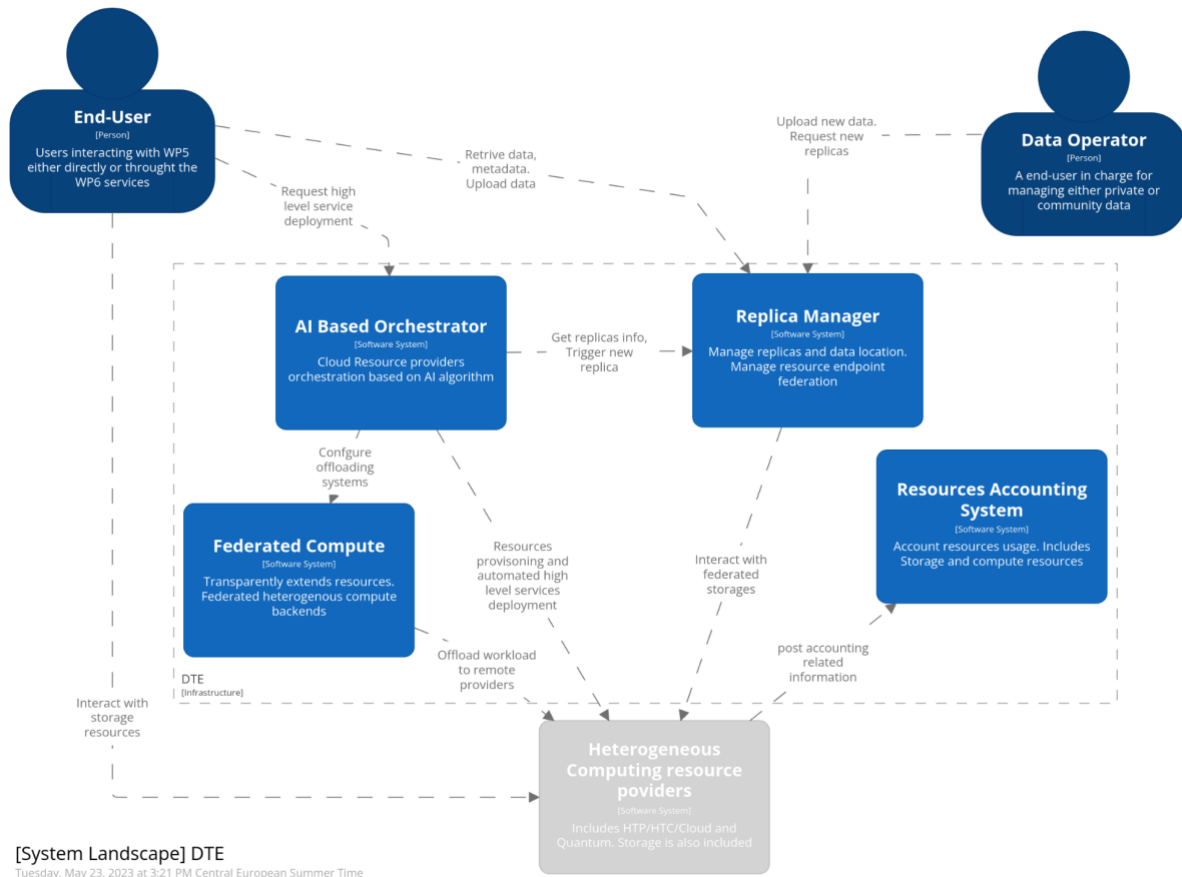


Figure 1 - System landscape diagram (in the C4 model) of the DTE Infrastructure. This is a high-level view that highlights the user interactions with WP5.



3 Components

3.1 Federated Compute

Component name	interLink
Description	<p>InterLink is an open-source service to enable transparent access to heterogeneous computing providers. It provides an abstraction for the execution of a Kubernetes pod on any remote resource capable of managing a Container execution lifecycle. The interLink component extends the Kubernetes Virtual Kubelet solution with a generic API layer for delegating pod execution on ANY remote backend. Kubernetes POD requests are digested through the API layer (e.g. deployed on an HPC edge) into batch job execution of a container.</p> <p>The API layer foresees a plugin structure to accommodate any possible backend integration.</p>
Value proposition	<p>Executing payloads in response to an external trigger like a storage event or a web server call</p> <p>Frameworks for DAG¹ workflow managements are usually well integrated with Kubernetes APIs</p> <p>Users can exploit interLink via self-provisioned deployment (i.e. through already integrated high level services) or standalone Kubernetes deployment creating and deploying a simple container that can be scheduled on a remote system such as a Slurm batch on an HPC center. High Level services getting integrated are e.g.: Airflow/Kubeflow pipelines/Argo workflows/MLFlow, Jupyter notebooks</p>
Users of the Component	Scientific users adopting Docker container-based workflow management or pipeline management system.
User Documentation	https://github.com/interTwin-eu/interLink#fast forward-quick-start
Technical Documentation	https://github.com/interTwin-eu/interLink#digging-in-the-project
Responsible	Italian National Institute for Nuclear Physics (INFN) Contact point: Daniele Spiga - spiga@pg.infn.it

¹ <https://hazelcast.com/glossary/directed-acyclic-graph/>



Licence	Apache 2.0
Source code	https://github.com/interTwin-eu/interLink

3.1.1 Release Notes

interLink is an open-source software product under development in the context of interTwin project. Its development started in order to provide an open-source solution capable of extending the container orchestration de-facto standard (kubernetes) to support offloading to any type of resource provider (Cloud/HTC/HPC) in a transparent manner, where little to no knowledge is required for the end user.. The key objective of interLink is to enable a Kubernetes cluster to send containers/pod to a “virtual” node. This node seamlessly manages the entire lifecycle of the user's applications, whether on a remote server or, preferably, within an HPC batch queue.

Transparency is granted by the fact that it keeps exposing the very same experience of running a pod on cloud resources, thanks to the API layer that exposes the regular set of Kubernetes APIs.

The interLink project is organised in 3 major subcomponents. Namely:

- A Virtual Kubelet
- The InterLink API
- Sidecars

Below a detailed description per each subcomponent:

1. Virtual Kubelet

The Virtual Kubelet is based on the latest release provided by Kubernetes.. It is responsible for the following actions:

- Attempting communication with the InterLink API to send a Service Account configuration. This configuration, utilised by the InterLink API, retrieves everything needed from the Kubernetes cluster for Sidecars. The configuration is then stored in a temporary file and sent in the HTTP request body.
- Initiating a Ping call to InterLink every 10 seconds to verify the connection's availability. Upon unsuccessful 'restore', a new Service Account is generated and sent again.
- Managing basic Pod operation, such as adding, removing, querying pods to/from the cluster.

For every registered Pod the Virtual Kubelet sends a Create HTTP call to the InterLink API. In addition:

- Every 5 seconds, a Status HTTP call to the InterLink API is automatically issued to check every Pod's health.
- If a job executed (by a Pod) is terminated, Pods are removed

HTTP is implemented following the REST standard. The body of every call is the same: a marshalled (JSON type) list of Pod descriptors, which are stored in variables of type v1.Pod, a built-in go-client standard Kubernetes type. The current status of the calls is summarised in **Table 1**.



Table 1 – Current status of the calls

Call	Outgoing URL
PingInterLink	InterLinkUrl:InterLinkPort/ping
SetKubeConfig	InterLinkUrl:InterLinkPort/setKubeCFG
Create	InterLinkUrl:InterLinkPort/create
Delete	InterLinkUrl:InterLinkPort/delete
Status	InterLinkUrl:InterLinkPort/status

2. InterLink API

InterLink API is the core of the service. It is responsible for translating Virtual Kubelet's HTTP calls into standard, agnostic outputs for the plugin-based structure (Sidecar). Below a description of the actions for each implemented call:

- *Ping*: allows the VK to check the InterLink API status
- *SetKubeConfig*: After receiving the configuration over the HTTP request body, it is stored inside `DataRootFolder/.kube/config` and set the `KUBECONFIG` env variable.
- *Create*: for each Pod registered to the Kubernetes cluster, the VK sends a Create Call to InterLink. In this phase, InterLink can retrieve all ConfigMaps, Secrets and EmptyDirs data. For all the containers in every single submitted Pod, secrets, ConfigMaps and, possibly, EmptyDirs are retrieved. All these values are then assembled together as a single struct of type `RetrievedPodData`. As a result, a JSON is sent over the HTTP request body to the Sidecar.
- *Status/Delete*: The received body is just forwarded to the Sidecar and any error is then handled and forwarded to the VK.

3. Sidecars

The architecture is plugin-based and foresees a plugin for each distinct backend to be supported. So far Docker and SLURM are available. The HTCondor plugin is in progress. For each API (VERB) the sidecars perform specific operations based on what the actual backend is. Submitting a Pod to the cluster means the Sidecar will receive from InterLink the list of all related Secrets, ConfigMaps, EmptyDirs and the description of the Pod itself. Utilising this information, the sidecar takes specific actions accordingly. Each Sidecar accepts the three standard outgoing calls described above the InterLink API.



3.1.2 Future Plans

In terms of features and capabilities, the next steps and priorities are further integrating the support for the Datalake and data management interoperability. The plan also foresees the offloading integration with INDIGO-PaaS intelligent orchestration.

Regarding the functionalities of the current software more in general we expect to focus more on logging capabilities in order to allow easier debugging in case of problems. Also, this information can be used for high-level reporting to users in case of problems

Additional improvements already planned based on early feedback include enhancements to volume management, Apptainer/Singularity integration. These updates aim to address site resources access policies and improve the management of File System permissions on remote resources.

3.2 Federated Data Infrastructure

3.2.1 Teapot

Component name	Teapot
Description	Teapot is an easy-to-install edge service that a facility may deploy to provide remote access to their 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.
Value proposition	Unlike existing storage solutions that are compatible with 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 of the Component	<p>Teapot is a core component that most users do not access directly. Instead, teapot is used by any DT manager or user who is (directly or otherwise) managing data locality.</p> <p>One reason for transferring data is to satisfy the data requirements of some workflow when the required data is missing from the local storage of the facility that will run the workflow. Under these circumstances, the user that triggered the workflow is using the teapot software if either the source facility (hosting the data) or destination facility (requiring the data) has deployed the teapot software.</p>



User Documentation	https://github.com/interTwin-eu/teapot/blob/main/CONFIGURATION.md
Technical Documentation	n/a
Responsible	Deutsches Elektronen-Synchrotron (DESY). Contact point: Paul Millar < paul.millar@desy.de >
Licence	Apache 2.0
Source code	https://github.com/interTwin-eu/teapot

3.2.2 ALISE

Component name	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.</p> <p>ALISE is an easy-to-deploy stand-alone 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 of the Component	All DT users that (directly or otherwise) make use of a facility that does not allow users to register their federated identity.
User Documentation	n/a

Technical Documentation	n/a
Responsible	Karlsruhe Institute of Technology (KIT). Contact point: Marcus Hardt < hardt@kit.edu >
Licence	MIT
Source code	https://git.scc.kit.edu/m-team/alise

3.2.3 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	<p>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).</p> <p>Deploying FTS provides a common service for handling such transfers, allowing higher-level data management; e.g., bandwidth shaping links between facilities.</p>
Users of the Component	<p>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.</p> <p>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.</p>
User Documentation	https://fts3-docs.web.cern.ch/fts3-docs/docs/cli.html
Technical Documentation	https://fts3-docs.web.cern.ch/fts3-docs/docs/developers.html
Responsible	CERN
Licence	Apache 2.0



Source code	https://gitlab.cern.ch/fts/fts3
-------------	---

3.2.4 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>
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

3.2.4.1 Release Notes

This release represents a technical preview of the interTwin federated data management solution.

There are two external software components: FTS and Rucio. They are fully established projects, independent of the interTwin project. The software is production-ready, at TRL 9, and hardened with many years of production-critical use. Both projects have multiple deployments of their software, operated by different communities.

The ALISE software is currently in a development phase, under the aegis of interTwin. At the time of release, ALISE is TRL 4. The user-facing functionality of ALISE is mostly feature-complete; however, anticipated changes to the API imply that the necessary integration work (whereby a service uses ALISE to identify a user) should be considered experimental. Feedback from early adopters is encouraged, but any plans to deploy ALISE should be tempered by the anticipated changes to the API.

The teapot software is also currently in a development phase, within the interTwin project. Teapot is TRL 3–4. The current release is sufficient to build a limited proof-of-concept demonstration, supporting the data transfer requirements of a single user.

3.2.4.2 Future plans

A number of improvements are planned for teapot. Teapot will be updated to support multiple, concurrent users. The per-user WebDAV instance management will be automated, starting new services on demand, and terminating them if there is sufficient idle time. It will also be updated to integrate with ALISE, to support automated identity management.

For ALISE, we anticipate possible improvements and stabilisation of the service-integration API, based on experience gained from integrating ALISE into various services. In addition, we plan to add support for client authentication in future versions of ALISE. This will limit access to the identity mapping information, providing this information to authorised services only.

There are currently no immediate plans to improve FTS. Future experience from integrating the testbed with the interTwin service providers may identify necessary enhancements to FTS.

Plan to enhance Rucio by improving support for integrating Rucio with external catalogues; e.g., Lattice QCD’s catalogue. Other plans to enhance Rucio may emerge from our experience when integrating Rucio with DTE core service, DTE thematic modules, and with the science use cases.



3.3 Intelligent Providers Orchestration

3.3.1 AI Based Orchestrator

Component name	INDIGO PaaS Orchestrator
Description	<p>INDIGO – PaaS Orchestrator is the core component of the INDIGO PaaS layer. It collects high-level deployment requests and translates them into action to coordinate resources interacting with the underlying cloud infrastructures.</p> <p>It allows the provisioning of virtualized compute and storage resources on different Cloud Management Frameworks (like OpenStack, OpenNebula, AWS, MS Azure, Google Cloud, etc.). The PaaS orchestrator features advanced federation and scheduling capabilities. It ensures transparent access to heterogeneous cloud environments and the selection of resource providers based on criteria like user’s SLAs, services availability, special hardware availability and data location.</p> <p>It manages deployment requests, expressed through templates written in TOSCA², the standard language for describing application topologies in cloud, and coordinates the deployment on the most suitable cloud site. To achieve this:</p> <ol style="list-style-type: none"> 1. it gathers SLAs, monitoring data and additional information from other platform services; 2. it asks the cloud provider ranker for a list of the best cloud sites.
Value proposition	<p>The INDIGO PaaS Orchestrator open source is a scientific gateway that allows users to easily access federated cloud systems, on top of which both simple and complex scientific virtual computational environments can be automatically deployed and configured.</p> <p>The provided dashboard implements a catalogue of services. When a user selects a specific service from the catalogue, the PaaS Orchestrator processes the inputs defined in the corresponding TOSCA template, completely automates the interaction with the backends, and hides the related complexity.</p>

² <http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html>



	The dashboard allows to simply define customised views of the Service Catalogue.
Users of the Component	In principle, all DT users that use the clouds to deploy their high-level services. Depending on the use cases, DT users may interact directly with the PaaS Orchestrator via Dashboard interface.
User Documentation	https://indigo-dc.gitbook.io/indigo-paas-orchestrator/
Technical Documentation	https://indigo-dc.gitbook.io/indigo-paas-orchestrator/
Responsible	Marica Antonacci - marica.antonacci@ba.infn.it
Licence	Apache 2.0
Source code	https://github.com/indigo-dc/orchestrator

3.3.1.1 Release Notes

The Orchestrator collects all the information needed to deploy the virtual infra/service/job consuming other PaaS APIs. It depends on external services such as:

- SLAM Service: get the prioritised list of SLAs per user/group;
- Configuration Management DB: get the capabilities of the underlying IaaS platforms;
- Data Management Service: get the status of the data files and storage resources needed by the service/application;
- Monitoring Service: get the IaaS services availability and their metrics;
- CloudProviderRanker Service (Rule Engine): sort the list of sites on the basis of rules defined per user/group/use-case.

The orchestrator-dashboard Simple Graphical UI is a Python application built with the Flask microframework. Flask-Dance is used for Openid-Connect/OAuth2 integration supporting the following functionalities:

- IAM authentication
- Display user's deployments
- Display deployment details, template, and log
- Delete deployment
- Create new deployment

3.3.1.2 Future Plans

The INDIGO PaaS Orchestrator will be enhanced in order to support a monitoring and Metering System. This will be a new component to collect metrics from the federated providers, not only for monitoring purposes but mainly to provide the Orchestrator with useful information for selecting the optimal deployment path. CMDB Component, an auxiliary service of the PaaS orchestration system that plays a crucial role in maintaining



and managing configuration information of the federated providers, has been completely re-engineered to provide a more flexible and general solution to support even more heterogeneous infrastructures. The algorithm used by the orchestrator to choose the resources where to deploy services will be enhanced. To achieve this, we intend to leverage AI in order to optimise the deployment time and enhance workflow performance, especially when dealing with big data analysis. Various techniques, such as machine learning algorithms, optimization algorithms, or other artificial intelligence methods aimed at improving the decision-making process within the Orchestrator will be explored and evaluated.

3.4 Resource Accounting System

3.4.1 APEL Accounting

Component name	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://wiki.egi.eu/wiki/APEL
Technical Documentation	https://wiki.egi.eu/wiki/APEL
Responsible	STFC UKRI
Licence	Apache License, Version 2.0
Source code	https://github.com/apel/apel (client and server software) https://github.com/apel/ssm (messaging tool) https://github.com/apel/grafana-dashboards/ (dashboard)



3.4.1.1 Release notes

This is a baseline release using mostly existing sub-components, configured for interTwin. These include the SSM (Secure STOMP Messenger) messaging tool to exchange accounting records, the APEL accounting repository software, and the Grafana-based Accounting Dashboard. It will be used to test integration with the other components and resource providers and form a basis for future development.

3.4.1.2 Future plans

The Accounting Dashboard will be customised to suit the needs of interTwin. As for the types of accounting collected by the accounting system, these will be extended to include GPUs and other accelerators.



4 Conclusions

The first release of WP5 software components includes the packages belonging to the four main assets of DTE Infrastructure. At the current stage of the project the integration between various components has just started while the bulk of the effort has been dedicated to the development of the subsystems itself.

Several brand-new systems are under development such as interLink ALISE and Teapot while other packages are building on top of existing software's and tools, such as RUCIO, FTS, INDIGO PaaS Orchestrator and APEL. Each of them is enhanced with new functionalities and features.

A first implementation of our packages, including the brand new one, is ready for a first round of validation.

The collaboration with resource providers, an integral part of WP5, remains crucial in this phase as well. All the prototypes under development are indeed co-designed, taking into account not only the perspectives of science communities but also those of the providers.

Although the presented software is not yet mature for an end-to-end workflow integration, there are few pilots built in collaboration with WP6 that are in progress. In particular:

- dCNiOS (dCache + Nifi + OSCAR) pilot to test and validate the payload execution on a virtual node. This has the objective to show how the system can be used in a system that executes payload in response to an external trigger, being it a storage event or a web server call
- interTwin AI Workflow management activities technically produced a demo container image on remote resources through interLink. Its aim is to show the integration example for DAG/workflow managements such as Airflow/Kubeflow pipelines/Argo workflows/MLFlow.

These initiatives are starting with Data and Storage integration as well. Both are starting the early testing of the DataLake features, particularly data injection and data retrieval, thanks to the services deployed at DESY.

The above-mentioned pilots are currently under integration benefiting from the testbed infrastructures supported by sites. In particular the federation under a common Kubernetes cluster deployed on cloud resources has been implemented on two EuroHPC sites:

- Vega: Deployed interLink Slurm layer on a VM on the edge of the HPC
- Juelich Deployed "remote docker run" interLink on a login node