IS-ENES3 Deliverable D10.2

First release of the ENES CDI software stack

Reporting period: 01/07/2020 – 31/12/2021

Authors: A. Spinuso (KNMI), W. Som de Cerff (KNMI), P. Nassisi (CMCC), C. Pagé (CERFACS)

Reviewers: P. Kershaw (UKRI), S. Kindermann (DKRZ), M. Lautenschlager (DKRZ)

Release date: 19/01/2021

ABSTRACT

The deliverable illustrates the first release of the ENES CDI software stack (software repositories, licensing information, change logs, links to technical documentation). We report about the implementation of the core data distribution services, climate4impact, ES-DOC, compute services, data request schema and tools for MIPs, and file metadata specifications. The deliverable reports on the realisation of the technical requirements and provides an update of the software architecture.

Dissemination Level		
PU	Public	X
CO	Confidential, only for the partners of the IS-ENES3 project	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824084

Revision Table			
Version	**Date**	**Name**	**Comments**
Document Structure and Contributors	8/10/2020	Alessandro Spinuso	Preliminary Structure and Objectives discussed on 9/10/2020
Collection and review of first round of contributions	13/10/2020	Alessandro Spinuso	Edited section 7.1 on Climate4Impact
Collection and review of second round of contributions	27/11/2020	Alessandro Spinuso	Updates on Tables and first ENES-CDI Release, overview
Completed Sections	4/12/2020	Paola Nassisi, Alessandro Spinuso	Updates to the Architecture, Executive Summary and Conclusion
Finalised work on reviewers' comments	11/01/2021	All	Rephrasing and reorganisation of a few sections and references to the architecture.

Table of contents

List Of Images

List Of Tables

Executive Summary

The ENES Climate Data Infrastructure (CDI) consists of a collection of services, software and metadata specifications, which are functional to the sustained access, evaluation and analysis of the data generated by CMIP and CORDEX simulations. In D10.1 [1] we have provided the general architecture of the envisaged infrastructure, with technical expectations for the mid to long term implementation. These are made concrete in this report, which provides the details on the progress made in the realisation of the architectural principles, leading to the first release of the CDI. For some of the components (ie. Climate4Impact), we motivate the need for redesign and reimplementation, to better approach the challenges set by the requirements of the CDI. We explain how these new developments are conducted, introducing recent software stacks and tooling. After the general introduction, the document is organised into 5 main sections. Section 2 provides the overview of the first release, providing updates to the architecture and the complete list of software and specifications. All the components are illustrated in the following four sections, respectively addressing (i) core data services, (ii) metadata schemas and reference tools, (iii) gateways for dissemination and access to computational facilities, (iv) solutions for authentication and authorisation. In the final conclusions we particularly highlight commonalities in the approaches pursued by some of the components. These involve technical choices, especially addressing flexibility of computation, and the exploitation of cloud technologies, as well as usability and exploitation challenges. The latter require the active participation and engagement of the target communities to foster the delivery of FAIR[1] derived products (PID, Metadata Conventions and Data Request Standards), towards the improved quality of user-centered services. Finally, we illustrate the most relevant future work, which we will undertake for the second release of the CDI.

[1] https://www.force11.org/group/fairgroup/fairprinciples

1 Introduction

This document provides a technical overview of the progress achieved with the implementation of the requirements and architecture presented in D10.1 "Architectural document of the ENES CDI software stack" [1] . The work presented in the report is packaged in a comprehensive official release, which addresses the different capabilities of the ENES infrastructure, from Data and Metadata, to Computation and Dissemination services. Each component will be described in respect to the progress made, the issues encountered, how these have been solved and what is left to be addressed. It will be clear what each component offers in the current release and how it connects and exploits the other capabilities of the infrastructure. We specify the means of access (eg. Available as a service / Software Package) and provide references to technical documentation and official repositories. Where applicable, deviations from D10.1 [1] are highlighted together with appropriate justification. In addition, any relevant upcoming developments will be highlighted together with plans for incorporation in a future release of the integrated ENES services.

2 CDI Release Overview

We provide here the updates of the ENES CDI Architecture and a summary of the software components which are available in the release. Sections are broken down by component and where appropriate details of the development of new software and services are also described. As these are under development they may not at time of writing be available as public releases.

2.1 Updates to the Architecture

As widely described in the document D10.1 "Architectural document on the ENES CDI software stack"[1], the ENES CDI is organized into multiple tiers and layers, through which the distributed components of the architecture interact with each other to provide the ENES community with a comprehensive set of services related to data and metadata access, dissemination and computation capabilities.
The ENES CDI architecture will be continuously updated during the project lifetime, allowing for the new requirements gathered in WP5/NA4 and coming from the IS-ENES community, also including external initiatives at both European (e.g. EOSC, EGI/EUDAT, Copernicus, etc.) and International level (e.g. ESGF). According to an *agile* approach, the next scheduled releases will reflect such inputs with an updated architecture diagram, thus properly adapting the development of individual components.

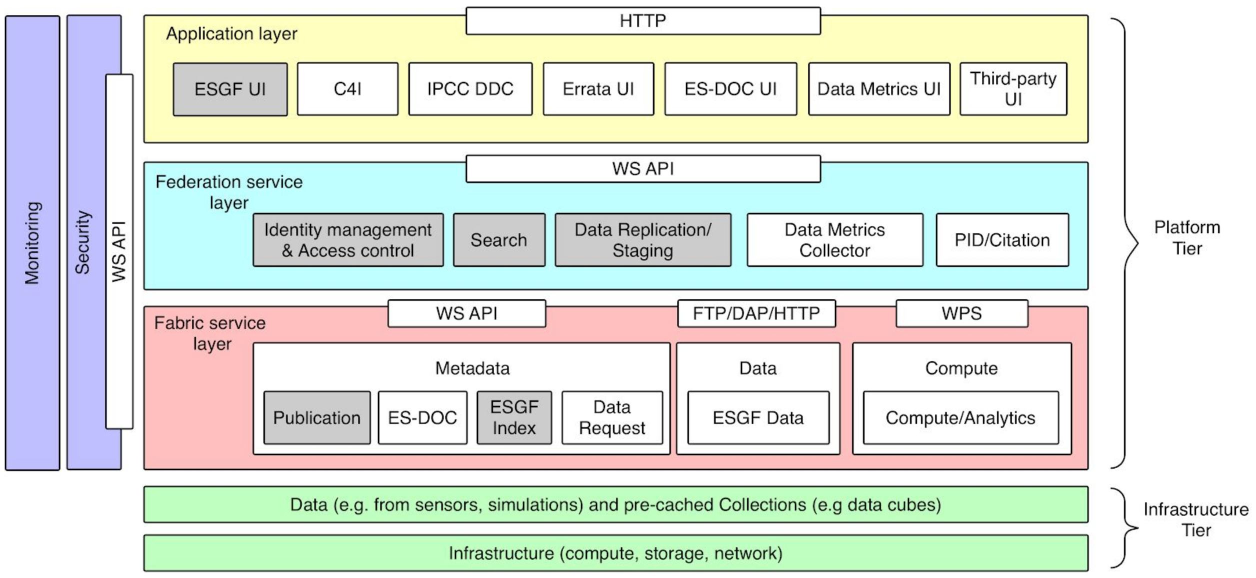

Figure 1. ENES CDI software stack architecture.

As a recap, Figure 1 above highlights the following structure of the ENES CDI software stack:

- **Platform Tier**: it includes the ENES CDI stack, a suite of software components that can be selectively deployed according to specific needs and goals. It consists of the following layers :
  - **Fabric** *(pink)*: it provides basic data, metadata and compute services;
  - **Federation** *(cyan)*: it federates and integrates different services at the Fabric layer across multiple collaborating organisations, thus providing federation-level capabilities (unified view) as well as a single entry point to the user;
  - **Application** *(yellow)*: it provides end-users applications to (i) perform data analysis, (i) get access to documentation, (iii) run/visualize climate indicators, (iv) report data usage/publication metrics, etc.
  - **Security** *(purple)*: it is an orthogonal layer of the stack that goes across the Fabric, Federation and Application layers. It includes, among others, firewall settings, OS/applications/services security updates, etc.
  - **Monitoring** *(purple)*: as in the case of Security, it represents a cross-architectural layer that addresses monitoring aspects at different levels (e.g. from infrastructure to services up to applications, with different sets of metrics).

  The picture also highlights the main protocols by which each layer exposes its services; they are very diverse on the *Fabric* layer and more convergent towards Web Service API and HTTP respectively on the *Federation* and *Application* layers.

The Platform Tier relates to any service of the ENES CDI that *could* be deployed according to a PaaS or SaaS approach in a public or private virtualized environment.

- **Infrastructure Tier** (*green*): it consists of compute, storage and network physical resources and, on top of it, the *Data*. These resources relate to sensors and to the output of numerical simulations.
  The Infrastructure Tier *could* be virtualized, thus providing access to the resources according to a IaaS approach either in a public or private cloud environment.

Finally, in grey, a set of ESGF services exploited in the ENES-CDI that are associated with collaborative development efforts carried out with partners outside Europe.

Figure 2 below shows an updated version of the UML component diagram of the ENES CDI software stack. There have been no major updates to the architecture with respect to the first version presented in document D10.1; however, the new diagram puts more emphasis on the authentication and authorisation aspects along with the interactions among components.

To make it easier to read it, only the arrows that connect different components have been left on the diagram, while internal connections have been left out (e.g. between ES-DOC Web UI in the Application layer and ES-DOC Service in the Fabric layer). Moreover, the different colours of the arrows reflect the type of connections respectively related to:

- data access in red;
- metadata access, search and publishing in blue;
- data processing submission in green;
- authentication and authorisation aspects in black;
- logging information about data download in orange.

As regards authorisation and authentication, the Policy Enforcement Point (PEP), is represented in full in the "Data Provider :Data Node" component. For simplicity, the PEP box, in dark grey, has been added to all the ENES CDI modules that require identity management and access control. In the future, more components could include a PEP to implement authorisation and authentication control.

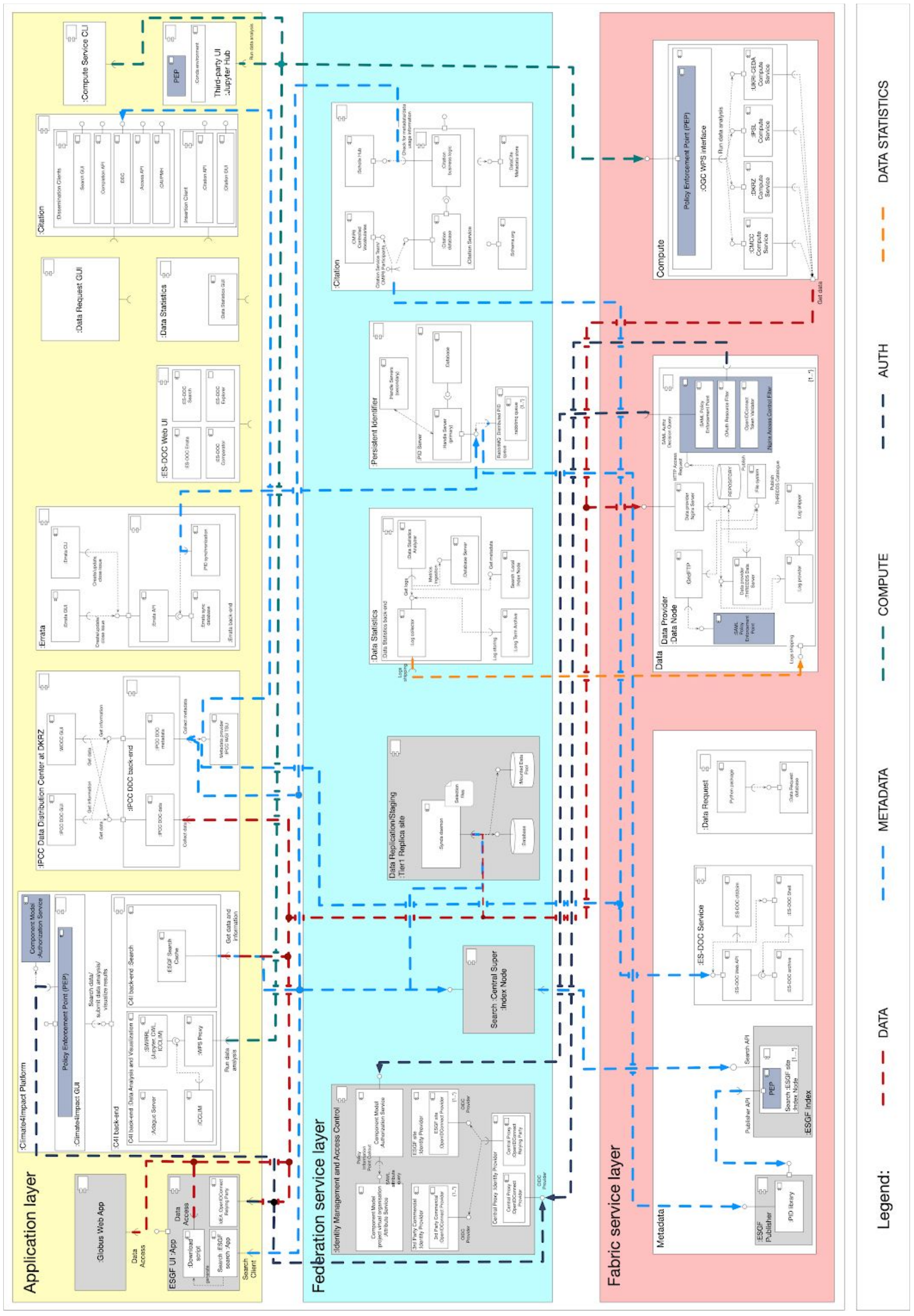

Figure 2. Detailed component diagram of the ENES CDI architecture.

More specifically, two types of interactions are required with the identity and access entitlement (IdEA) component, and two black arrows should start from all the components that include a PEP, respectively connected to the Central Proxy OpenID Connect Provider and to the Authorisation Service. For the sake of simplicity and to reduce the number of arrows represented, only a few examples of modules have been connected to the central authentication/authorisation service, the Data Node and the Climate4Impact portal. For more technical details, please refer to section 6 of this document that describes the IdEA system in more depth.
Once the new system has been integrated into the ENES CDI architecture, each site requiring access control will have one or more instances of a local authorisation service that will manage access policies for the site's secured applications. This type of implementation has been reported in the Climate4Impact platform diagram, which hosts a local authorisation service that the PEP points to. In this case, only the arrow pointing to the Central OpenID Connect Provider has been highlighted in black. The other arrow is an internal interaction between the PEP and the local authorisation service.

2.2 Integrated Available Software and Services

We present here an overview of the software available in the current release. We include access URLs (where the component is deployed and available as a service), its source code repository (if public) and the current version tags associated with the release, if these have been produced.

ENES CDI Service	**Software components/ Service access url / Repositories**	**Release Tag/ Version/ Branch**
ESGF Data	*esg-publisher* doc: https://esgf.github.io/esg-publisher/index.html repo: https://github.com/ESGF/esg-publisher	4.0.0-beta2
	esgf prepare repo: https://github.com/ESGF/esgf-prepare	2.9.6
	esgf-pyclient repo: https://github.com/ESGF/esgf-pyclient	0.2.2

	CoG repo: https://github.com/EarthSystemCoG/COG	master branch
Data Citation	doc: http://cmip6cite.wdc-climate.de repo: internal gitlab software versioning at DKRZ	(restricted access)
Persistent Identifier (PID)	*ESGF data publication pid library* repo: https://github.com/IS-ENES-Data/esgf-pid	0.8.0
	RabbitMQ federation doc: https://acme-climate.atlassian.net/wiki/spaces/ESGF/pages/107708573/PID+Services+Working+Team+esgf-pidwt	(restricted access)
	PID consumer library doc and repo: https://gitlab.dkrz.de/esgf/handlequeueconsumer	(restricted access)
IPCC Data Distribution Centre at DKRZ	*IPCC Data Distribution Centre at DKRZ:* http://ipcc.wdc-climate.de DDC web pages on server hosted at CEDA: http://www.ipcc-data.org/sim/	
Errata	*ESGF Errata Service*: https://errata.es-doc.org/ doc: https://es-doc.github.io/esdoc-errata-client/ repos: ● Web-Service: https://github.com/ES-DOC/esdoc-errata-ws ● Front-end: https://github.com/ES-DOC/esdoc-errata-fe ● CLI: https://github.com/ES-DOC/esdoc-errata-client	Web-Service**:** master branch Front-end: master branch CLI: 2.3.1

Data Statistics	*ESGF Data Statistics UI:* http://esgf-ui.cmcc.it:8080/esgf-dashboard-ui/ doc: ● https://acme-climate.atlassian.net/wiki/spaces/ESGF/pages/1043464194/Federated+data+usage+statistics+ESGF+Dashboard ● https://acme-climate.atlassian.net/wiki/spaces/ESGF/pages/1054113816/Proposed+ESGF+Usage+of+Filebeat+and+Logstash repo: https://github.com/ESGF/esgf-dashboard https://github.com/ESGF/esgf-dashboard-ui	esgf-dashboard: 2.0.0 esgf-dashboard-ui: 1.0.0
Data Replication	*Synda replication software package* doc: http://prodiguer.github.io/synda/ repo: https://github.com/Prodiguer/synda	3.15
Compute	*ECAS* service: https://ecaslab.cmcc.it/jupyter/hub/login doc: https://ecaslab.cmcc.it/web/home.html repo: https://github.com/ECAS-Lab	
	Ophidia doc: http://ophidia.cmcc.it/ repo: https://github.com/OphidiaBigData	
	Birdhouse WPS framework doc: https://birdhouse.readthedocs.io/en/latest/ repo: https://github.com/bird-house security proxy: https://github.com/bird-house/twitcher	twitcher: 0.5.4
	ESGF-specific WPS framework under development for C3S prototype repos under development at: https://github.com/roocs underlying library: https://github.com/pydata/xarray	

	Third party components: - *JupyterHub:* https://jupyter.org/hub - *xarray:* *http://xarray.pydata.org/en/stable/*	
Climate4Impact	*C4I front-end, C4I backend, C4I storybook, C4I errorhandler, C4I front end content, C4I search portal backend, C4I map preview, C4I frontend dataset preview* service: https://climate4impact.eu/impactportal/general/index.jsp repo: https://gitlab.com/is-enes-cdi-c4i	c4i-backend 0.1.0 c4i-frontend 0.2.3
ES-DOC	*ES-DOC service and documentation*: http://es-doc.org CIM repo: https://github.com/ES-DOC/esdoc-cim-v2-schema pyesdoc repo: https://github.com/ES-DOC/esdoc-py-client pyessv repo: https://github.com/ES-DOC/pyessv CMIP6 content repos: https://github.com/ES-DOC-INSTITUTIONAL cdf2cim repo: https://github.com/ES-DOC/esdoc-cdf2cim	CIM v2.2, pyesdoc: v0.14.2.0, pyessv: v0.8.4.3 cdf2cim, v1.0.3.0
Climate Forecast (CF)	service: http://cfconventions.org/ doc: http://cfconventions.org/ repo: https://github.com/cf-convention/	
cfdm: a Python reference implementation of the CF data model.	software: https://pypi.org/project/cfdm/	1.8.7.0

cfchecker: the NetCDF Climate Forecast Conventions compliance checker	software: https://pypi.org/project/cfchecker/	4.0
cf-python: a CF-compliant earth science data analysis library	software: https://pypi.org/project/cf-python/	3.7
Identity Management and Access Entitlement	doc: https://github.com/ESGF/esgf.github.io/wiki/Security%7CInterfaceControlDocument service: Attribute and Authorisation Services repo: https://github.com/ESGF/esgf-security service: OAuth 2.0 and Short-lived Credential Service repo: https://github.com/ESGF/esgf-slcs-server	esgf-security: v2.8.11 esgf-slsc-server: 0.1.0

Table 1. First ENES-CDI Release, Software and Services overview.

3 Data Services

The core ENES-CDI data services provide the capabilities needed to meet the functional requirements mentioned in Table 2 of D10.1 [1], Section 3.2.1. More specifically, those labelled as [DATAFR#-], [CITFR#-], [PIDFR#-], [DDCFR#-], [ERRFR#-], [STATSFR#-], [REPLICFR#-].

3.1 ESGF Data

The ESGF data service helps to publish data and make it available to users. The service consists of several components, all are freely available on Github:

3.1.1 esgf-prepare:

The esgf-prepare module (https://github.com/ESGF/esgf-prepare) helps data providers to create a project-related standardized directory structure for better organization of data files. It also provides a command to iterate over all files and create mapfiles for the publication to the EGSF.

3.1.2 esg-publisher:

The esg-publisher (https://github.com/ESGF/esg-publisher) can be used to publish the data to all data related components in the ESGF, ie. a Postgres database, a THREDDS[2] data server and a Solr Index. It takes the mapfiles from 5.2.1 as input and reads all related files to extract the required metadata. The publisher component is also related to the PID server and creates the dataset PIDs and sends all PID information to the RabbitMQ[3] servers to register the PIDs.

3.1.3 esg-search:

The esg-search component (https://github.com/ESGF/esg-search) is integrated in the ESGF frontend (CoG - https://github.com/ESGF/COG) so users can easily search and download data using different search facets.

3.2 Data Citation

Data Citation has become an integral part of scholarly publications. Initiatives like COPDESS[4], ESIP[5], FORCE11[6] or Scholix[7] work on standardizations and guidelines for data citations. IPCC WGI of the current IPCC cycle integrates data citations in the AR6 to improve the traceability and transparency of the key findings of the climate assessment. In order to enable the citation of CMIP6 data, the data has to be provided for humans as well as for machine-readable access. Another key consideration is to disseminate the information about CMIP6 data references outside the project context (http://cmip6cite.wdc-climate.de).

3.2.1 Brief reminder of Data Citation Service functionality

The service provides three functions:

1. Collection of information and support of the CMIP6 participants is provided. Data citation information and further information on e.g. relevant paper reference or author details can be maintained via a GUI and an API.

[2] https://www.unidata.ucar.edu/software/tds/
[3] https://www.rabbitmq.com/
[4] https://copdess.org/
[5] https://www.esipfed.org/
[6] https://www.force11.org/
[7] http://www.scholix.org/

2. Automated processing of information checks metadata completeness and availability of data for DOIs registration, adds data usage information of papers citing CMIP6 data, and monitors and corrects information in a semi-automated fashion.
3. Citation information is provided for human and machine-readable access via project-specific interfaces and via traditional ways such as XML serialisation of data on an OAI[8] server.

3.2.2 Accessing citation information

The existing tools for accessing citation information were consolidated, especially the OAI server application was updated. The interface to Scholix to add information on data usage in scientific papers was improved using actual cases. The citation search functionality was enhanced to provide data license information and the documentation was extended with a user guide (http://bit.ly/CMIP6_Citation_Search).

3.3 Persistent Identifiers

The persistent identifier (PID) service consists of multiple interacting independently deployed components with clear APIs and interfaces which are integrated into the ESGF ENES CDI infrastructure:

- A distributed message transport layer
- ENES CDI specific message server components deployed at DKRZ
- Handle system[9] backend components for handling storage and generic PID CRUD (Create, read, update and delete) operations.
- PID publication client tools (integrated into the ESGF publication software and interacting with the distributed message transport layer)
- PID curation tools to correct missing or erroneous PID registrations

3.3.1 Current status

For the current release the following key activities were performed and related future work are planned:

- The PID registration infrastructure layer based on a distributed RabbitMQ message queue was updated for the current release to exclude erroneous message relay servers and include new (e.g. updated) ones. This did not affect the overall APIs, integrations as well as operations of the PID service, yet improved the resilience of the system.

[8] https://www.openarchives.org/
[9] http://www.handle.net/

- To handle additional PID namespaces additional PID handle backends are deployed at DKRZ, to e.g. manage CORDEX and Copernicus related PID registrations
- The PID publication client tools after migration to Python 3 stayed stable, only small updates were required to handle specific error situations.

3.3.2 Next steps

The curation of the existing overall ESGF PID collection required an improvement to the existing set of PID curation tools. There is a constant need to clean up PID metadata after PID ESGF publication because of erroneous publication/depublication activities by some ESGF data nodes. Currently this is centrally done at DKRZ by a data manager based on ad hoc developed curation tools. Future work will integrate these tools into a consistently managed tool set managed on Github. The growing demand for PIDs means that additional PID prefixes need to be allocated and associated handle system backbones need to be deployed. Work will be needed to automate the deployment and scaling of the handle system backends.

3.4 IPCC Data Distribution Centre at DKRZ

The IPCC WG1[10] is one of the early users of CMIP6 data. The IPCC DDC supports the authors in the writing process, especially in the analysis of data to derive key findings by providing Virtual Workspaces. The CMIP6 data subset underlying the AR6 will be long-term archived in the IPCC DDC AR6 Reference Data Archive as part of the traceability of AR6 key findings and as well as for data re-use. The quality requirements for the IPCC DDC data and metadata are high, complying to the TRUST principles (Transparency, Responsibility, User Focus, Sustainability, Technology) [8] as implemented in e.g. the Core Trust Seal.

3.4.1 Brief reminder of IPCC DDC core functionality

The main functions of the IPCC DDC at DKRZ are:

- Long-Term Archival of CMIP6 data subset at a snapshot date to build the IPCC AR6 Reference Data Archive including metadata enrichment from different documentation services like ES-DOC and the Citation Service.
- Long-Term Data Stewardship and data access for users from all IPCC member states.

3.4.2 Long-Term Data Archival to build the IPCC AR6 Reference Data Archive

In discussions with the WG1 Technical Support Unit (TSU), the format for the CMIP6 dataset list was defined and a timeline for metadata provision was agreed. Long-term data archival is planned to start at the end of March 2021. CORDEX data from European domains used in AR6 WG1 will be added to AR6 Reference Data Archive as second priority using the workflow and interfaces defined for the CMIP6 data archival.

[10] https://www.ipcc.ch/working-group/wg1/

The interface to the citation service to add mandatory metadata not available in the ESGF index was coded. It is currently tested and improved in the long-term data archival of the input4MIPs[11] data. The interface to ES-DOC to add ancillary documentation on experiments, models and simulations is still under discussion.

3.4.3 Long-Term Data Stewardship and Data Access

There are no updates to report because of the timeline. A gap analysis among the DDC Partners was carried out, which provides guidance and sets priorities for future developments for data access and data discovery improvements.

3.5 Errata

The errata service is an answer to the problem raised by the inherent complexity of projects like CMIP5 and CMIP6. It provides a platform to record and track the reasons motivating a dataset version change. The quality of data increases, when the principles of proper handling of errata information are clearly set and adhered to. This information is specified in the documentation and within the ESGF publication workflow. Version changes should be documented and justified by detailing what has been updated, retracted and/or removed. This chapter is a brief reminder of the design and implementation of the Errata service as well as a change log for the different components.

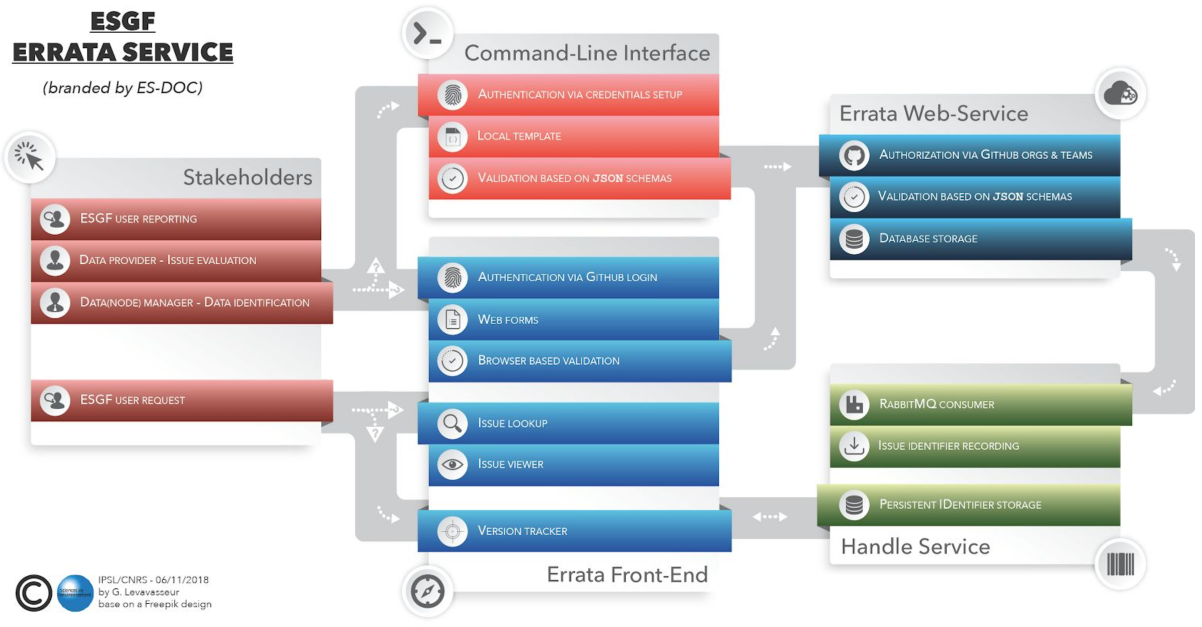

Figure 3. Errata Service architecture.

[11] https://esgf-node.llnl.gov/projects/input4mips/

3.5.1 Brief reminder of system architecture

As a part of the ES-DOC ecosystem, the Errata Service offers a user-friendly front-end and a dedicated API to provide timely information about known issues affecting ESGF data (Figure 3).

ESGF users can query for modifications and/or corrections applied to the data in different ways:

- through the centralized and filtered list of ESGF known issues;
- through the “PID lookup” interface to get the version history of a (set of) file/dataset(s).

The Errata service provides support for every project hosted on ESGF. This is made possible through pyessv (Python Earth Science Standard Vocabularies) integration, that automates the interpretation of the controlled vocabulary, to extract the essential facet values as long as the project’s vocabulary is described in the *ini* file that feeds pyessv. This mechanism enables the Errata service to become somewhat project-agnostic, and able to host new project datasets with no required codebase update.

Contributions to the Errata service are, for the time being, subject to access restrictions based on users’ identity and affiliation. Entries to the service can be made through a lightweight CLI (command line interface) or a web-form.
As shown in the architectural diagram (Figure 2), the Errata Service uses the Persistent IDentifier (PID) attached to each dataset and file during the ESGF publication process to persist the errata information. The documentation has been fully revised to guide users through the errata procedure.

3.5.2 Authentication and authorization

In its current state, the errata service is only open in write mode for authenticated and authorized users. This is however subject to change very soon as we work on a system evolution requested by the community to open contributions to everyone. The process is still controlled through Github OAuth service as detailed in the previous reports.

3.5.3 Issue life-cycle

The lifecycle of an issue will also be subject to change to reflect the development proposals mentioned in the previous section. In the future, an issue can be a contribution from any member of the community. In this case it will remain in a probation period waiting for moderation from the specific data managing party. Once this period is over, the issue is published, whether it has been approved or not. The original way of publishing issues by identified errata officers remains supported and is possible through either the command line client or the webform as detailed in the previous report.

3.5.4 Issue body requirements

This was detailed in previous reports and is available on-demand in the errata system documentation here : https://es-doc.github.io/esdoc-errata-client/

3.5.5 PID integration

When a new issue is created/updated through either the command line errata client or the web hosted form, the PID (Persistent Identifier) attached to every dataset declared with the issue is subsequently updated reflecting the changes.
This mechanism has been implemented in an asynchronous fashion. This leads to a fluid workflow for the errata creation/update, while the dirty work of making sure the errata information is properly persisted in the designated PID handles is performed independently through a sync cron job to ensure that it is failsafe. This however also means there will be a slight delay between the errata information creation and when it will appear in the dataset and file handles. However, given the information will immediately be present on the index of the errata service, we deemed this trade-off is exactly what we need.
In the process of adding more resilience to this integration with the Errata system, a new node of RabbitMQ has been deployed at IPSL. This enables the Errata system to have a choice of messaging queues to use in order to update the contents of the dataset handles with the newly discovered errata.

3.5.6 API

The errata service still exposes the same API endpoints that were detailed in D10.1 [1] the previous report and in the user documentation. These endpoints facilitate the interaction with third party software.

3.6 ESGF Data Statistics

The ESGF Data Statistics service provides a distributed and scalable framework responsible for capturing, analyzing and providing data usage and data publication metrics at a single data node level, for within ENES and at the scope of the whole of ESGF.
With respect to the ENES CDI Architecture diagram in Figure 2, the ESGF Data Statistics is located under the Federation service layer and directly interacts with i) the connected data nodes and ii) a dedicated local index node to retrieve all the metadata information about the data downloaded by the nodes.
From a high level perspective, it i) collects and stores a high volume of heterogeneous metrics, covering general and project-specific measures, ii) aggregates such metrics through an ad-hoc ETL system, iii) stores them into a dedicated data warehouse iv) and provides a rich set of charts

and reports through a web interface, allowing users and system managers to visualize the status of the IS-ENES/ESGF infrastructure through a set of smart and attractive web gadgets.

3.6.1 General architecture

The overall architecture (Figure 4) relies on a set of different components, ranging from the logging system for the collection of basic logging information on downloads, to the Collector module that is composed of the Dashboard Analyzer and the data warehouse/data marts repository. The LTA (Long Term Archival) Agent manages archival of the log info and, at the end of the chain, the user interface displays the metrics through graphical widgets (like charts, maps, tables and so on), giving the user a comprehensive view of how services in the federation are being used.

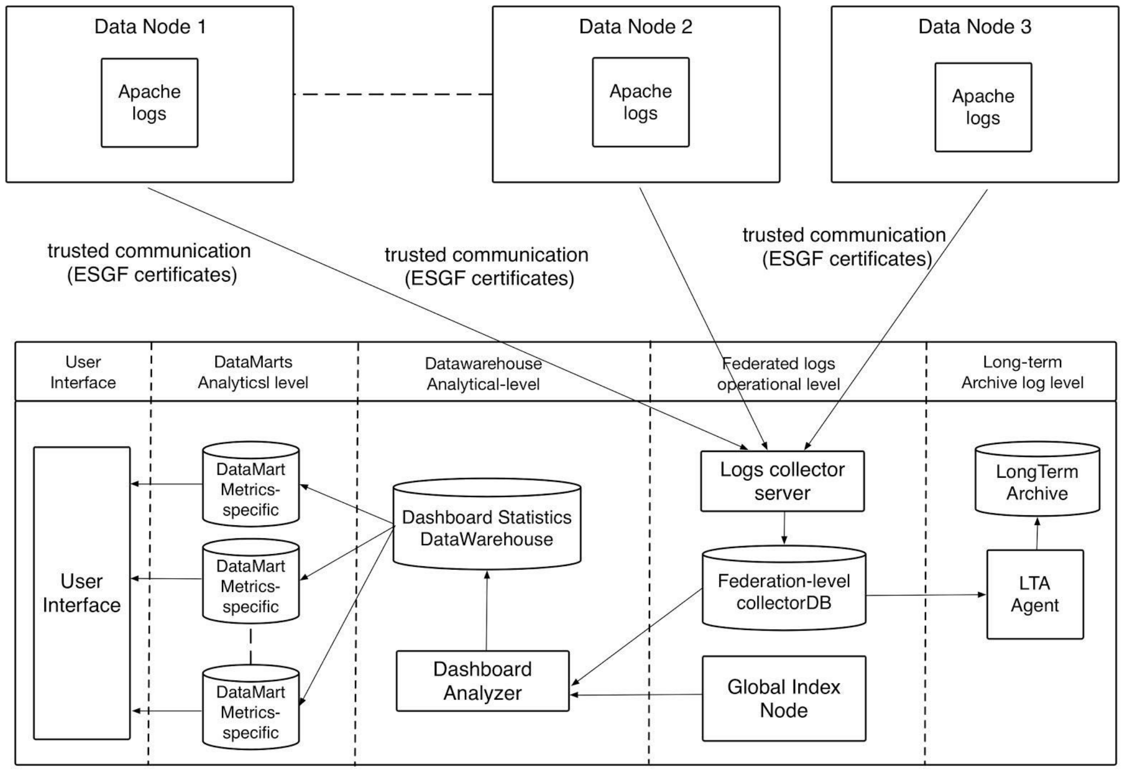
Data Node 1
Apache logs
Data Node 2
Apache logs
Data Node 3
Apache logs
trusted communication (ESGF certificates)
trusted communication (ESGF certificates)
trusted communication (ESGF certificates)
User Interface
DataMarts Analyticsl level
Datawarehouse Analytical-level
Federated logs operational level
Long-term Archive log level
User Interface
DataMart Metrics-specific
DataMart Metrics-specific
DataMart Metrics-specific
Dashboard Statistics DataWarehouse
Dashboard Analyzer
Logs collector server
Federation-level collectorDB
Global Index Node
LongTerm Archive
LTA Agent

Figure 4. The ESGF Data Statistics Architecture

3.6.2 First release details

The implementation of the first service release consisted of three main activities carried out in parallel that can be grouped as follows:

1. setup of the logs transferring environment from the data nodes to the collector node;
2. implementation of the log analysis, storage chain and deployment of a local index node;

3. implementation and deployment of the graphical user interface.

With regard to the first activity, two industrial tools produced by Elastic, have been deployed, i.e. Filebeat[12] and Logstash[13]: the former on the data nodes and the latter on the collector node. Once the Apache web server has been properly configured to produce the logs in a specific format, Filebeat reports log events as they occur to the Logstash instance deployed at the CMCC Supercomputing Center. Since sensitive information, such as IP addresses, can't be transmitted to the Logstash instance, an *httpd* module has been introduced to automatically translate IPs into country codes.
The core of the log analysis occurs on the collector node. A specific operational chain has been implemented to process every log related to a single download.
For each log entry, the process contacts the ESGF Solr module of a local Index Node deployed at the collector side to gather additional information such as the file size and other metadata closely related to data that the file refers to (variable, experiment, model, …) [STATSFR#2 and STATSFR#3] (Milestone 10.1 "Technical requirements on the software stack").
In addition, an appropriate backup mechanism has been implemented to protect against data loss.
An ad-hoc application has been implemented to periodically query the Solr APIs of the local Index Node and get information about the data volume and the number of published datasets, for both the whole federation and the top projects [STATSFR#1]. The Solr response is parsed by the application and information is stored into specific tables of the collector database.
The considerable amount of data usage information produced at this stage is stored into a specific data warehouse system, which collects an extended set of statistics not only about logging information but also about project-specific download statistics and geolocation of clients.
The Data Statistics user interface represents the last step of the whole Data metrics architecture and the main entry point for final users who want to get information about data usage and data publication metrics, and also gauge the level of interest from the community in specific climate datasets, projects, variables, etc.

In this first release, no particular issues have been noticed and the system is properly working; indeed, data usage and data publication metrics have been provided for the first IS-ENES3 periodic report and for the mid-term review [STATSFR#5].

The code of the logs analyzer and the Data Statistics user interface are respectively available at the following GitHub repositories: https://github.com/ESGF/esgf-dashboard and https://github.com/ESGF/esgf-dashboard-ui.

[12] https://www.elastic.co/beats/filebeat
[13] https://www.elastic.co/logstash

3.6.3 Current status

In its current state, the Data Statistics service regularly gathers logs from 23 data nodes in Europe, United States and Australia and provides insights into the exploitation of the ESGF federation through a web user interface at this link: esgf-ui.cmcc.it/esgf-dashboard-ui.

3.6.4 Next steps

The plan is to keep improving the user experience and enhance the visualization capabilities offered by the current user interface. New project-specific views will be provided according to users' requirements. Interactions with the final users and the data node administrators will be fundamental in order to continue to provide a useful tool for their studies and work.

3.7 Data Replication

Data replication involves the following architectural components:

- ESGF data nodes ("Tier 2") providing access to the originally published data collections from individual modelling centres;
- ESGF replica nodes ("Tier 1") providing access to original data as well as replicated datasets. These replica nodes are associated to larger data pools hosting replicated datasets and also to high performance data transfer nodes supporting Globus-based data transfer;
- a replication management software component ("Synda") hosted at replica nodes, which triggers and manages parallel data replication streams involving different data nodes and different transfer protocols;
- a site specific data ingest and publication workflow integrating the replica datasets in the local data pools and publishing these datasets via ESGF.

The replication software "Synda" currently sits at version 3.14, and is available for download through conda, counting 288 total downloads since the migration from RPM style packaging. We are currently increasing the number of releases per month in the hope of achieving an agile workflow that best answers the ever increasing requirements of the community.

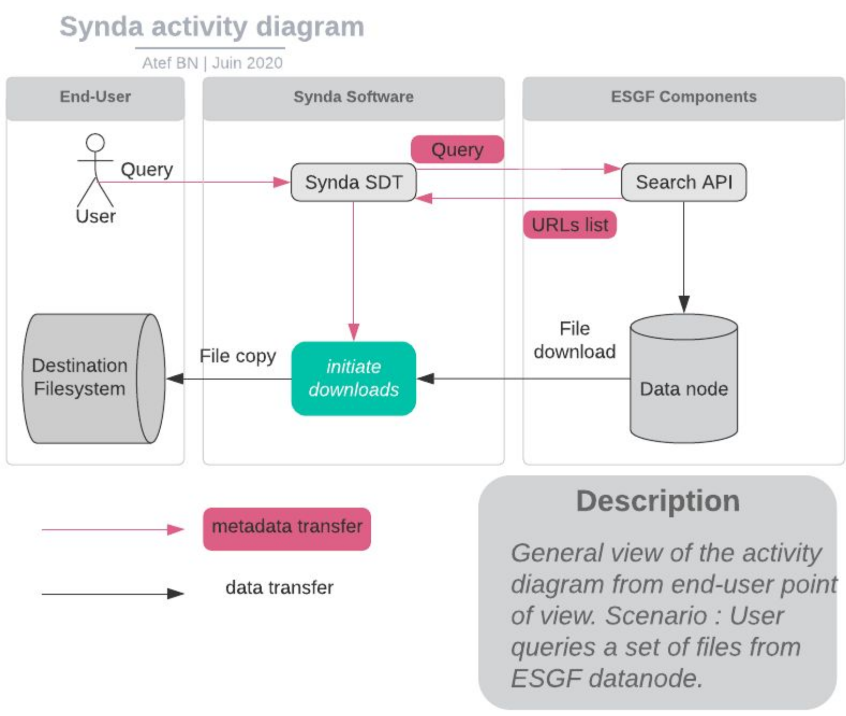

Figure 5. Synda software general view.

4 Metadata Schema and Services

The capability of efficiently handling metadata and data request schema is at the foundation to build the ENES CDI system in such a way to comply with the basic FAIR principles [4]. It addresses functional and non-functional requirements of the infrastructure. The former [CFFR#-] are mostly concerned with guaranteeing the provisioning of understandable standards to enable findability (F) and access to the data (A), while the latter [NFR#11] enables those mechanisms that, through interoperability (I) foster data reuse (R) and, to some extent, its reproducibility. These efforts will be further reinforced in the coming release of the CDI, which will also include the specification of the Metadata for Climate Indices.

4.1 Climate and Forecast Convention

Substantial efforts have been conducted to develop software that validates compliance with the agreed CF standards[14], which aim at providing a description of the physical meaning of data and of their spatial and temporal properties. We list below the main contributions.

[14] http://cfconventions.org/

Software Description	Relevant Versions
cfdm: a Python reference implementation of the CF data model. https://pypi.org/project/cfdm/	1.8.7.0
cfchecker: the NetCDF Climate Forecast Conventions compliance checkerhttps://pypi.org/project/cfchecker/	4.0
cf-python: a CF-compliant earth science data analysis library https://pypi.org/project/cf-python/	3.7

Table 2. Software modules compliant to the Climate and Forecast Convention.

The tools have been developed by taking into account the official specification of the standards. Documentation web pages and discussion repositories, which led to the current definition of the vocabularies are listed below.

- Document: CF Convention Version 1.8 Document: http://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.html
- Document: CF Standard Names Version 76 http://cfconventions.org/Data/cf-standard-names/76/build/cf-standard-name-table.html
- Discussion: Conventions: https://github.com/cf-convention/cf-conventions/issues
- Discussion: Standard Names: https://github.com/cf-convention/discuss

4.2 CMIP Data Request

The Coupled Model Intercomparison Project Phase 6 (CMIP6) seeks to improve understanding of climate and climate change by encouraging climate research centres to perform a series of coordinated climate model experiments that produce a standardized set of output. Twenty-three independently led model intercomparison projects (MIPs) have designed the experiments and have been endorsed for inclusion in CMIP6 (Eyring et al., 2016). An essential requirement of CMIP6 is that the thousands of diagnostics generated at each centre from hundreds of simulations should be produced and documented in a consistent manner to facilitate meaningful comparisons across models. Hence, for each experiment, the MIPs have requested specific output to be archived and shared via the Earth System Grid Federation (ESGF), and the CMIP6 organizers have imposed requirements on file format and metadata. The resulting collection of output variables (usually in a gridded form covering the globe and evolving in time) and the associated temporal and/or spatial constraints on them are referred to as the CMIP6 Data Request (DREQ).

The modelling centres participating in CMIP6 are now archiving the requested model output and making it available for analysis. The DREQ is significantly more complicated than the data requests from previous CMIP phases, complexity which arises from the size of CMIP6 and the inter-relationships of MIPs. We describe the challenges, introduce the tools which were provided to capture and communicate the DREQ, provide some headline statistics associated with the DREQ and outline some of the problems encountered and potential solutions for future exercises.

4.2.1 Current Status

The 01.00.33 release of the CMIP6 data request incorporates 6 experiments which have been added to the request at a late stage to look at the response of the atmosphere to sudden changes in emissions resulting from the global response to the COVID-19 pandemic. There are also a small number of minor corrections to variable descriptions.

In this release, the software and document are still coupled, though the software can work with multiple releases. This is expected to change in the near future, allowing independent release cycles for software and database content.

4.2.2 Next Steps

The IS-ENES3 Milestone M10.2 [9] sets out the high-level structure for the next generation of the data request, including the data model and a discussion of the match to requirements. The new schema is designed to support greater flexibility in dealing with expected evolving requirements from the CMIP community.

The next steps will involve gradual implementation, starting with a separation between the software and the database components.

4.2.3 Summary of resources

We provide here a comprehensive list of resources that can be examined to gain detailed understanding of the process behind the specification of the standard. They highlight pending issues and include the reference implementation of a programmatic tool and an interactive browser of the schema.

    - Forms: XLS Templates (https://w3id.org/cmip6dr)
    - Discussion: Github issues:
        - Variables (https://github.com/cmip6dr/CMIP6_DataRequest_VariableDefinitions/issues)
        - Request (https://github.com/cmip6dr/Request/issues)

- Tools for user-access
    - Software: dreqPy (https://pypi.org/search/?q=dreqPy)
    - Database: XML document within the dreqPy software package'
    - Service: Data Request browser (https://w3id.org/cmip6dr/browse.html)

5 Dissemination and Computational Services

In respect to the Requirements Overview (Table 2 of D10.1 [1]), in this section we address the Compute & Analytics functional requirements [COMPFR#-]. Non functional aspects (Table 3 of D10.1 [1]) will address mostly [NFR#8] and [NFR#11], concerning flexibility and interoperability of the services, respectively.

The services illustrated in this section are improving the capabilities of the CDI for the provision of datasets and documentation, and the allocation of computational workspaces. They will adopt and further develop innovative technologies to better address the way researchers conduct their work, with built in mechanisms for FAIRness [3] and reproducibility [4].

Thereby, interoperability standards, such as WPS[15] will be complemented and sometimes replaced by more flexible tools, where users benefit from enhanced freedom and control in experimenting with the data. This justified the redesign efforts undertaken for some of the CDI components (Climate4Impact) and the growing adoption of solutions based on workflow management systems, digital notebooks [2] and domain-specific software libraries. These are coupled with persistent, as well as disposable storage and computational resources, which are deployed at the ENES CDI sites or on commercial cloud services.

[15] WPS https://www.ogc.org/standards/wps

5.1 Climate4Impact

The IS-ENES Climate4Impact (C4I)[16] is an integrated portal and analysis platform that provides an easier and single access point to climate simulations for end users [C4IFR#1], especially the climate change impact modelling community. The current release available to the public provides a range of processing capabilities, from time and spatial subsetting [C4IFR#5] to simple statistics, such as time average, to more complex calculations, such as climate indices and indicators [C4IFR#7]. Its codebase has been refactored in order to migrate to a microservice based deployment model. This facilitated the management of its operations in the current production environment, which is now hosted on the Amazon Cloud Platform (AWS[17]).

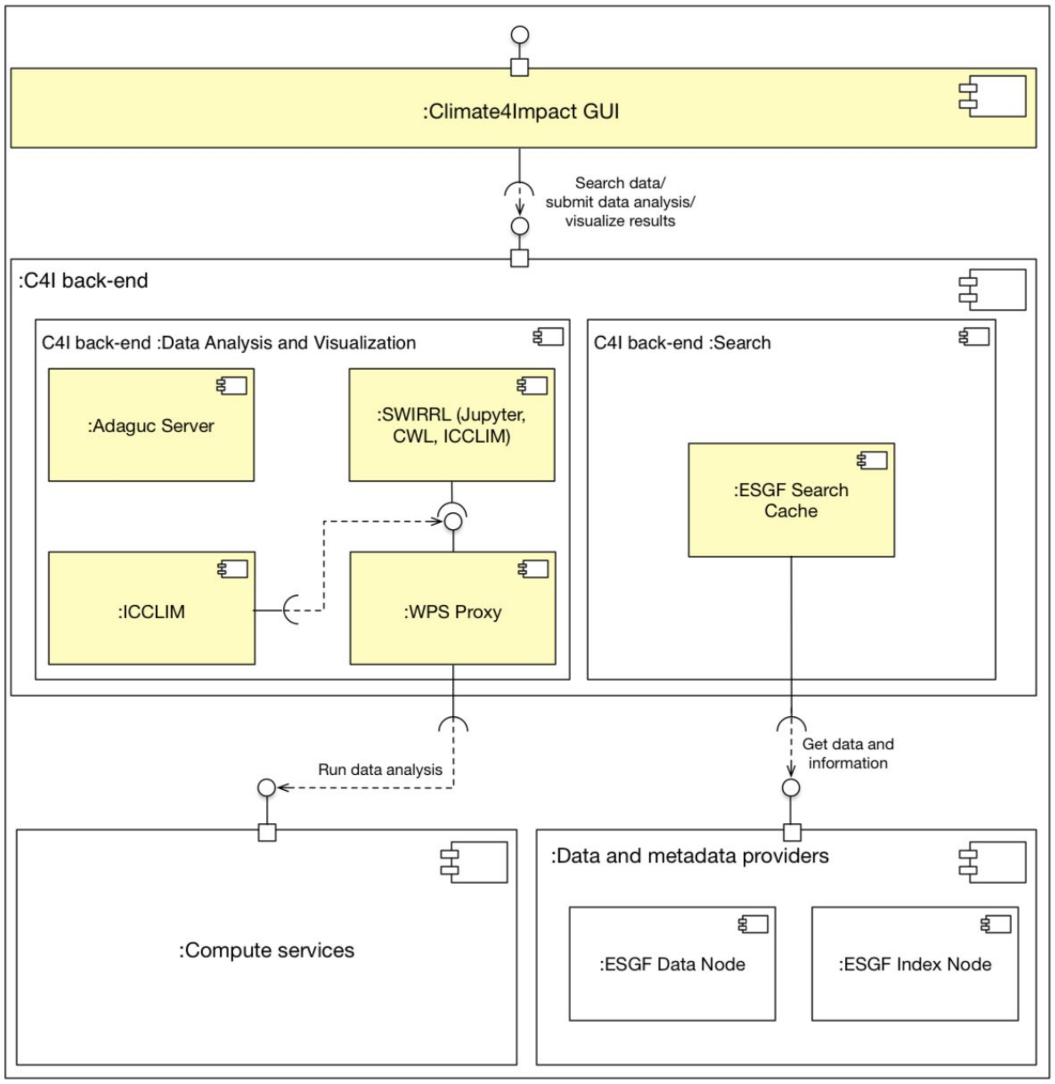

Figure 6. Updated Component Diagram of the new Climate4Impact.

[16] C4I https://climate4impact.eu
[17] Amazon Web Services. https://aws.amazon.com

As already mentioned in D10.1 [1], besides the improved quality of the production codebase, the service is undergoing major redesign efforts with a new selection of components (Figure 6), while keeping the original architectural principles and relationships with the rest of the CDI (Figure 2). This is motivated by the need to improve the usability of the discovery functionality, as well as offering users with new approaches to conduct data-analysis tasks. These should allow them more control and flexibility, while automating those mechanisms necessary for reproducibility. As means to achieve these targets, in D10.1 [1] we have explicitly included as components the ICCLIM[18] library and SWIRRL API[19] combining requirements such as [C4IFR#7] and [COMPFR#8], thereby adding more flexibility in the implementation and analysis of Climate Indices and Indicators. Further pre-existing components, such as Adaguc[20] and the connection to WPS services, will be integrated incrementally according to refined requirements collected after the evaluation of these new developments.

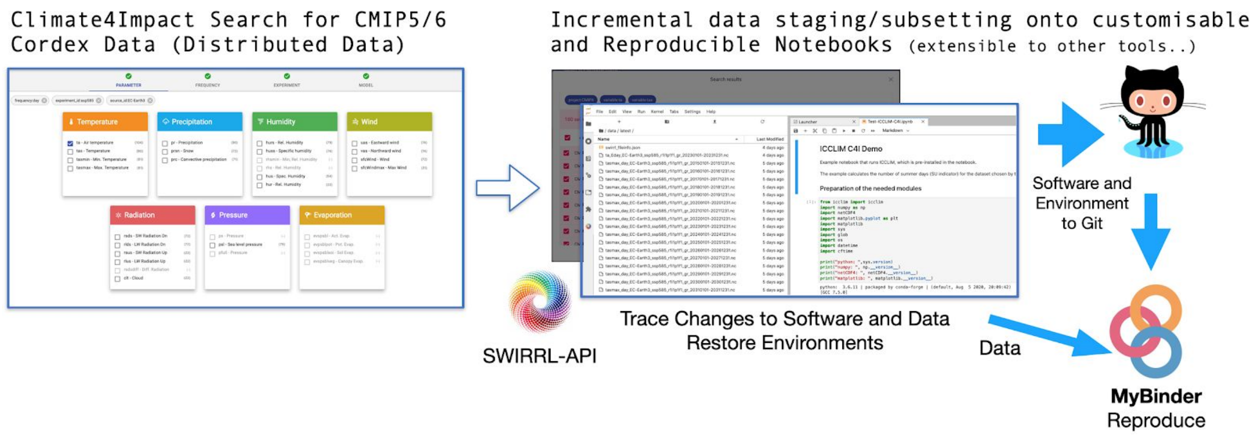
Climate4Impact Search for CMIP5/6 Cordex Data (Distributed Data)
Incremental data staging/subsetting onto customisable and Reproducible Notebooks (extensible to other tools..)
Temperature
Precipitation
Humidity
Wind
Radiation
Pressure
Evaporation
Software and Environment to Git
Trace Changes to Software and Data
Restore Environments
SWIRRL-API
Data
MyBinder
Reproduce

Figure 7. Integration of new C4I search interface with SWIRRL-API.

5.1.1 Current Status

Recent developments focussed on extending the new version of C4I with reproducible computational tools. These tools are obtained via the SWIRRL API, which manages, in a provenance-aware manner, working sessions offering notebooks and batch processes, consisting in Jupyter Notebooks [2] and processing jobs implemented as a combination of containers and workflow scripts (i.e. expressed in CWL[21]), respectively. At the moment the latter consist mainly of staging operations that move the selected data onto the cloud resources hosting the notebooks. This effort triggered new challenges in handling potentially large data-shipments. This is addressed in two ways: a) increased granularity of the data-selection process, to enable users to

[18] ICCLIM https://icclim.readthedocs.io/en/latest/
[19] SWIRRL https://zenodo.org/record/4264852#.X60BkNv_q7p
[20] Adaguc http://adaguc.knmi.nl/
[21] https://www.commonwl.org/

choose at the level of a single data-file, as opposed to the whole dataset; b) implementation of a new workflow with remote subsetting capabilities (OpenDAP[22]), which is activated and parameterised by the users by specifying time and spatial components. Computations performed in C4I/SWIRRL can be also packaged as snapshots. These are automatically stored in the user's GitHub account or in a shared repository that can be re-enacted in Binder (http://mybinder.com). (Figure 7). The new C4I has been demonstrated in official IS-ENES training sessions and will be open to *alpha* testers by early 2021. However, the current production version of C4I portal[23] is still available to users with its usual set of tools. In Table 3 we list the repositories and the versions associated with the C4I developments.

5.1.2 Relation with the other components of the ENES-CDI

As expected by M7.3, we will produce a plan for the integration of the information available in the ESMValTool portal[24] to provide easy access to pre-computed Models Evaluation data. Moreover, C4I will provide explicit links to the landing pages associated with the datasets' DOI, addressing the requirement [CITFR#3] on access to citation information, when this is available, highlighting the overall FAIRness [4] of the IS-ENES infrastructure. As shown in Figure 2 and Figure 6, Climate4Impact uses the federated search services offered by the ESGF and connects to the ESGF Identity management and access control component of the CDI (Section 6). This is needed to handle the identification of the users and their authorization to access the different data repositories, especially CORDEX.

Repository URL	Description	Relevant Versions
https://gitlab.com/is-enes-cdi-c4i	Production C4I portal available at https://climate4impact.eu	c4i-backend 0.1.0 c4i-frontend 0.2.3
https://gitlab.com/is-enes-cdi-c4i/is-enes3	*Alpha* releases of the new C4I portal	master branch
https://gitlab.com/KNMI-OSS/swirrl/swirrl-api https://gitlab.com/KNMI-OSS/swirrl/jupyterswirrlui	SWIRRL API and Jupyter-Lab extensions	master branch
https://github.com/cerfacs-globc/icclim	icclim (Index Calculation CLIMate) library	4.2.14

Table 3. Climate4Impact Software Components and Repositories.

[22] https://www.opendap.org/
[23] https://climate4impact.eu
[24] https://cmip-esmvaltool.dkrz.de/

5.1.3 Next Steps

The implementation is conducted according to the architectural directions specified in [1] and updated in Section 2 of this document. We are incrementally addressing the fundamental requirements associated with the service, including additional flexibility brought by combining notebooks and reproducible analysis. In the next phase the work will continue improving the usability and performance of the new computational tools and the search interfaces, enabling user authentication in the new version of the service. These activities will be driven by the feedback collected from the group of *alpha* testers and from the demands of scientific use cases that will be developed in the form of reproducible notebooks.

5.2 ES-DOC

The ES-DOC (Earth System Documentation) software ecosystem facilitates both the provision and the consumption of documentation of the CMIP6 workflow and, where possible, automates the various and often complex stages involved.

The toolchain comprises an interrelated set of libraries, services and storage solutions. The core user-facing asset is a website hosted with WordPress, supported by dedicated servers and databases on the former WebFaction commercial Cloud hosting (now migrating to tsoHost[25]), which consolidates the web applications and maps them onto domains. The software ecosystem and archive contained in GitHub repositories under the 'ES-DOC' organisation enables content pushed by modelling institutes to be processed into CIM documents (the canonical format for representing aspects of the modelling workflow), stored, and displayed on the Wordpress website for discovering, viewing and comparing.

The back-end software stack comprises several Python-based utility libraries to automate the creation and publication of standardised documents and controlled vocabularies, and the storage of documents in repositories on GitHub and through five databases on WebFaction. In addition there is a shell-script library to facilitate development and maintenance using these tools on the system of multiple separately version-controlled repositories.

The front-end stack consists of Python web services to manage documentation and errata stored in the WebFaction databases, for generating and publishing the model documentation, and for rewriting various URLs; and JavaScript (Vue framework) web applications that support the viewing, searching, and comparing of the published documentation, as well as serving and displaying other relevant content.

[25] https://www.tsohost.com/

These processes being provided by the ES-DOC software stack are described in the workflow diagram of Figure 38 of report [1] (D10.1 - Architectural document of the ENES CDI software stack). All elements of this generic workflow have been implemented and are fully available as services, software packages and specification documents:

- Tools for incorporating external standards
  - Specification: CIM (https://github.com/ES-DOC/esdoc-cim-v2-schema)
  - Specification: CMIP6_CVs (https://github.com/WCRP-CMIP/CMIP6_CVs)
  - Software: pyessv (https://github.com/ES-DOC/pyessv)
  - Software: pyesdoc (https://github.com/ES-DOC/esdoc-py-client)

- Tools for creating documents and populating the ES-DOC archive
  - Software: pyesdoc (https://github.com/ES-DOC/esdoc-py-client)

- Tools for user-access of the ES-DOC archive content
  - Software: pyesdoc (https://github.com/ES-DOC/esdoc-py-client)
  - Software: ES-DOC web API (https://github.com/ES-DOC/esdoc-api)
  - Service: ES-DOC explorer (https://explore.es-doc.org)
  - Service: ES-DOC comparator (https://compare.es-doc.org)

- Integration with the CMIP6 errata service and DKRZ data citation service
  - Software: pyesdoc (https://github.com/ES-DOC/esdoc-py-client)
  - Software: ES-DOC web API (https://github.com/ES-DOC/esdoc-api)
  - Service: ES-DOC-explorer (https://explore.es-doc.org)
  - Service: further_info_url

All software in the stack is available from the ES-DOC organisation GitHub repositories at https://github.com/ES-DOC.

Not all document types are available yet for creation (by the CMIP6 modelling groups) and consumption (by users of the CMIP6 outputs). The missing items are the creation and archiving of descriptions of conformance to numerical requirements; and the comparison of all document types other than model descriptions. These will be relatively straightforward to include during 2021, as the implementation has been designed to be easily applicable to new use-cases.

As well as completing the set of CMIP6 documents, a document versioning scheme needs to be implemented, and a framework for collecting document quality control information and presenting it to the data users.

The application of the ecosystem to non-CMIP6 workflows also needs to be considered. These include the CORDEX and input4mips projects. It is likely extensions may be required for these other settings. The work for documenting the CORDEX models has started and has not needed any software library extensions, rather a new application of the existing tool kit.

5.3 Institutional compute service deployments at ENES CDI sites

This section describes the first release of the ENES CDI compute layer for processing and analytics of CMIP6 and CORDEX data.

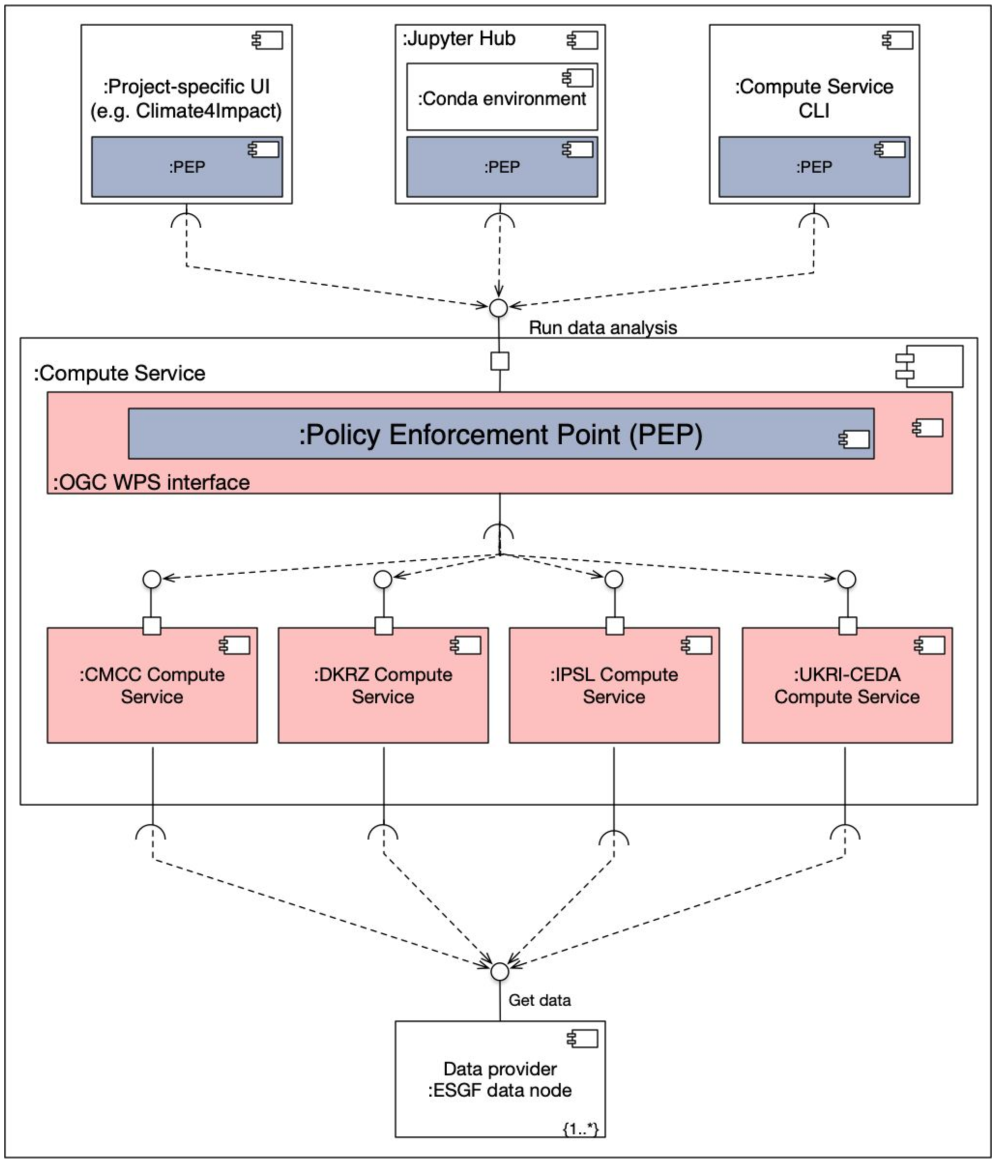

Figure 8. Compute service component diagram

Beyond the different implementations of the core analytics services developed at each site, addressing institutional and national requirements, the main goal is to move towards a sustainable and integrated data analytics and processing layer to efficiently support end-user needs. To this aim, three common aspects that each compute service should implement during the project lifetime has been defined and reported below:

- an interoperable and flexible server front-end based on the OGC-WPS interface [COMPFR#7][NFR#11][NFR#8];
- a programmatic client interface [COMPFR#6] with a Python binding;
- a security infrastructure based on the work and roadmap defined with the ESGF IdEA WG activity [COMPFR#5].

In addition to these aspects, the component diagram, in Figure 8, highlights the strong link of each institutional implementation with the data services to foster the "data near processing capabilities" paradigm. The diagram has been further detailed to also cover security aspects and ensure legitimate use of the computing resources at the host facility by external users. As described in section 2.1 of this document, some Policy Enforcement Points have been added at different levels to implement authentication and authorisation control.
In the next months, further steps will be taken in this direction, in close collaboration with the activities performed in WP5/NA4, to provide an increasingly integrated compute service for the ENES CDI.

The progress made in the different institutional deployments of the compute service are reported below.

5.3.1 Compute Service at CMCC

The compute service designed at CMCC implements an "Analytics Hub" tailored to meet the needs of multi-model climate analysis, which requires access to a large amount of data (i.e. from CMIP experiments) available through the ESGF federated data archive, besides running workflows with tens/hundreds of data analytics operators [COMPFR#4]. To serve the entire climate community needs, multiple and distributed Analytics-Hubs can be deployed, each one referring to a small set of variables.

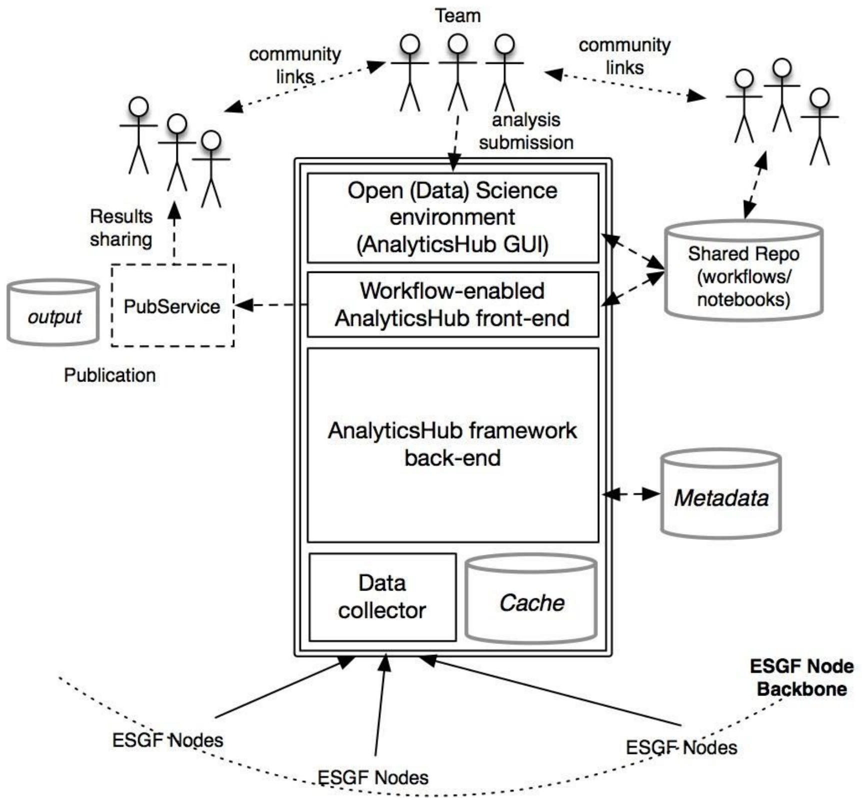

Figure 9. Analytics-Hub architecture.

5.3.1.1 General architecture

The detailed Analytics-Hub architecture is shown in Figure 9. As it can be seen, it consists of multiple components: (i) an interface/GUI providing an Open (data) Science-ready environment for Data Science applications, interactive and exploratory data analysis, visualization, etc. [COMPFR#9]; (ii) a workflow-enabled, secure, and interoperable front-end; (iii) an analytics framework back-end to perform data analysis at scale [NFR#3] and support metadata management; (iv) a data collector and its local storage [COMPFR#1] to gather the relevant datasets from ESGF and keep them synchronized with the remote repositories, as well as other auxiliary services for publication and sharing of results and code. Additional details about the concepts that inspired the design of the component and the architecture can be found in D10.1 [1].

5.3.1.2 AnalyticsHub GUI

JupyterHub provides a multi-user environment for executing multiple instances of the Jupyter Notebook service, by also managing user authentication and authorization. Jupyter Notebooks represent a web-based application to create, edit, run and share composable documents

containing live code, equations and plots[26]. It provides support for several programming languages through specific kernels for Python, R, Julia, Ruby and Scala among others. Additionally, it provides a file manager panel to handle notebooks and run terminals that resemble Linux shells [2].
In the context of the CMCC AnalyticsHub, the target programming language at the level of the Data Science environment is Python. A specific Python-based environment has been set up with the most well-known Python modules and some other modules required by the users; some examples include Cartopy, Matplotlib, NumPy, SciPy, Pandas, etc. Conda has been used to handle more effectively the creation of the environment and manage the installation of Python packages and their dependencies[27]. The conda environment has then been made available to users from the JupyterHub instance as a specific kernel, by using the IPython kernel module[28]. The IPython Kernel allows an easy integration of additional conda[29] environments or virtualenvs[30] with the set of kernels exploitable in Jupyter Notebooks. Thanks to this module, users can also extend the default conda environment or create new environments, and use them directly from the Jupyter Notebook interface. A customized JupyterHub spawner has also been implemented in order to load the proper conda environment at notebook startup.
Besides the setup of the environment, an initial set of demonstration and supporting notebooks has been created to inform users about some of the main features available in the system and how to use them in some simple, yet real world examples [COMPFR#8].

5.3.1.3 AnalyticsHub front-end and back-end

The Ophidia framework provides server-side [COMPFR#1], in-memory, parallel data analysis for multidimensional scientific data in multiple domains with a focus on the climate domain. It implements an internal storage model, leveraging the so-called datacube abstraction, and a hierarchical data organization to manage large amounts of multidimensional scientific data. Ophidia provides more than 50 metadata and (parallel) datacube operators and about 100 array-based primitives. To target interoperability [NFR#11][COMPFR#7], the Ophidia front-end server exposes several interfaces, such as a SOAP compliant with Web Services Interoperability (WS-I) Basic Profile v1.2[31], an Open Geospatial Consortium Web Processing Service (OGC-WPS)[32], and a Grid Security Infrastructure (GSI) [6] with support for Virtual Organisations (Virtual Organization Membership Service - VOMS [7]). The Ophidia Terminal, a command line interface (CLI) -like application similar to a Linux terminal, along with a Python module are provided on the client-side to run interactive or batch experiment sessions. Finally, in

[26] https://jupyter.org/
[27] https://docs.conda.io/projects/conda/en/latest/index.html
[28] https://ipython.readthedocs.io/en/stable/install/kernel_install.html
[29] https://docs.conda.io/projects/conda/en/latest/user-guide/concepts/environments.html
[30] https://virtualenv.pypa.io/en/latest/
[31] http://ws-i.org/Profiles/BasicProfile-1.2-2010-11-09.html
[32] http://docs.opengeospatial.org/is/14-065/14-065.html

order to address complex scientific use cases, Ophidia provides a native analytics workflow engine for defining processing chains and workflows with hundreds of data analytics operators.
In the context of the CMCC AnalyticsHub, Ophidia represents the main component providing complex data analytics and workflow features. It has been deployed in the environment and can be easily exploited from the JupyterHub interface through the Ophidia Python binding - PyOphidia[33] [COMPFR#6].

5.3.1.4 AnalyticsHub data collector

The Analytics-Hub is responsible for providing computing and analytics capabilities on top of a data collection layer which both (i) pre-stages and caches the data relevant to the analyses from the different ESGF data nodes and (ii) keeps the local copy of data synchronised with the remote copy available in the ESGF infrastructure.
In the current release, a first prototype of the data collection logic has been implemented, using Sproket[34] as the main tool for the download operation. It relies on JSON format for the definition of the ESGF search criteria and can be easily configured to define the target ESGF index node that data has been indexed on. To speed up the download process, a parallelization approach based on Python threading has been exploited, allowing different parts of the whole process to run concurrently. As a first step towards validating the reliability of the implemented procedure, the precipitation variable has been downloaded, with a time frequency of three hours (3hr) and the ssp245 and ssp585 experiments. In the future, according to users' requirements, other variables will be taken into consideration and included in the catalogue.

5.3.1.5 Next steps

The plan is to continue the development of the data collection layer, by strengthening the download process and implementing the logic of synchronization with the ESGF data nodes catalogues. Moreover, a science portal will be developed, to provide users with timely information about the capabilities of the CMCC Analytics-Hub, as well as documentation to allow easy access to the Jupyter environment. The conda environment will be enriched with additional libraries based on new user requirements. JupyterHub extensions (e.g. JupyterLab)[35] will also be taken into consideration in future releases to further increase the user experience. Finally, example Python notebooks and workflows will be implemented to demonstrate the capabilities of the integrated service.

[33] https://github.com/OphidiaBigData/PyOphidia
[34] https://github.com/ESGF/sproket
[35] https://jupyterlab.readthedocs.io

5.3.2 Compute Service at DKRZ

The compute service at DKRZ is centered around the access to interactive nodes with access to the HPC system and an attached high performance data pool. The overall structure and components are described in the previous deliverable [1]. The following updates and improvements were performed for this release of the ENES CDI:

- A production ready JupyterHub deployment[36] was made available replacing the initial pre-production system now supporting a predefined set of resource profiles defining the compute resource allocations for interactive sessions. These allocations vary from single CPU or single compute nodes to large allocations of resources to support parallel processing (e.g. using dask[37]) by exploiting the binding to the HPC batch system (slurm[38])
- A set of pre-defined Jupyter kernels was made available covering the basic requirements of many users
- For advanced users the possibility was added to integrate user-specific compute kernels into the DKRZ JupyterHub environment. This functionality was an often requested feature by scientists wanting to use new cutting edge data science libraries and own code bases as part of their work. This functionality e.g. was already exploited as part of IS-ENES3 to make specific ESMValTool functionality accessible as part of Jupyter notebooks running at DKRZ
- Users of the compute service need an easy to use data catalogue system. As part of the evolving Pangeo community and tool ecosystem the use of Intake catalogs is becoming more and more important. To react on this requirement, for the DKRZ CMIP data pool (together with other data collection e.g. ERA re-analysis data) a regular automatic Intake catalogue generation capability was deployed[39]. Example notebooks were developed[40] illustrating this feature for IS-ENES3 compute service users.

Future work will concentrate on developing good documentation on the efficient exploitation of the parallel processing capabilities especially in the context of xarray[41] and dask usage scenarios, which gain more and more popularity in the climate model community. Work also continues to deploy WPS processing web services supporting basic capabilities like subsetting and regridding. They are currently being prototyped and deployed to support data integration into the Copernicus climate data store. First pre-production ENES CDI WPS compute services will be deployed at DKRZ mid 2021 and made available via virtual access (with limits to the numbers of concurrent users of these services).

[36] https://jupyterhub.dkrz.de/
[37] https://dask.org/
[38] https://www.dkrz.de/up/systems/mistral/running-jobs/slurm-introduction
[39] https://gitlab.dkrz.de/mipdata/intake-esm
[40] https://portal.enes.org/data/data-metadata-service/analysis-platforms
[41] http://xarray.pydata.org/en/stable/

5.3.3 Compute Service at CNRS-IPSL

The compute service design at IPSL still mainly relies on generic remote access to dedicated login nodes (see [1], Figure 32).
Since the last report, IPSL storage capacity has been rationalized and reorganized for users with dedicated project shared-spaces. A new THREDDS service has been deployed to facilitate data sharing among project partners.
Each user can still ask for a work space with dedicated storage that will be balanced with other requests. The IPSL mesocentre provides 50 TB shared storage for data analysis (Lustre), temporary and final results, alongside of a 4Po of specific CMIP and CORDEX and observational datasets (Reanalysis, Obs4MIPs, input4MIPS, etc.) with centralized access (including the whole French climate modelling production from IPSL and CNRM).

The IPSL mesocentre improved several pre-configured Python environments that activate mutualized and useful tools for data quality check and analysis. Those environments now include the well-known Xarray and Dask libraries that provide user-friendly I/O and parallel tasking. The "Dask-jobqueue"[42] plugin is used to interface Dask with the usual IPSL PBS (Portable Batch System) manager (ciclad-web.ipsl.jussieu.fr). Individual users can reserve different compute resources through a cluster with compute nodes (2000 cores, up to 256 GB RAM/node).

Current work concentrates on:

1. providing a flexible JupyterHub environment that will interface with the HPC system.
2. containerizing the Python environments on top of a Kubernetes instance to be able to address and scale the different end-users compute service use-cases and requirements.
3. cataloging CMIP and CORDEX data using STAC-like technology (i.e., intake-esm).

5.3.4 Compute Service at UKRI (CEDA)

The compute service at UKRI is based on the JASMIN petascale storage and cloud computing infrastructure[43] for big data challenges in environmental science. JASMIN provides the UK and European climate and earth-system science communities with an efficient data analysis environment. Many datasets, particularly model data, are too big to be easily shipped around: JASMIN enables scientists to bring their processing to the data.

[42] Dask-jobqueue: https://jobqueue.dask.org/en/latest/
[43] JASMIN: https://jasmin.ac.uk/

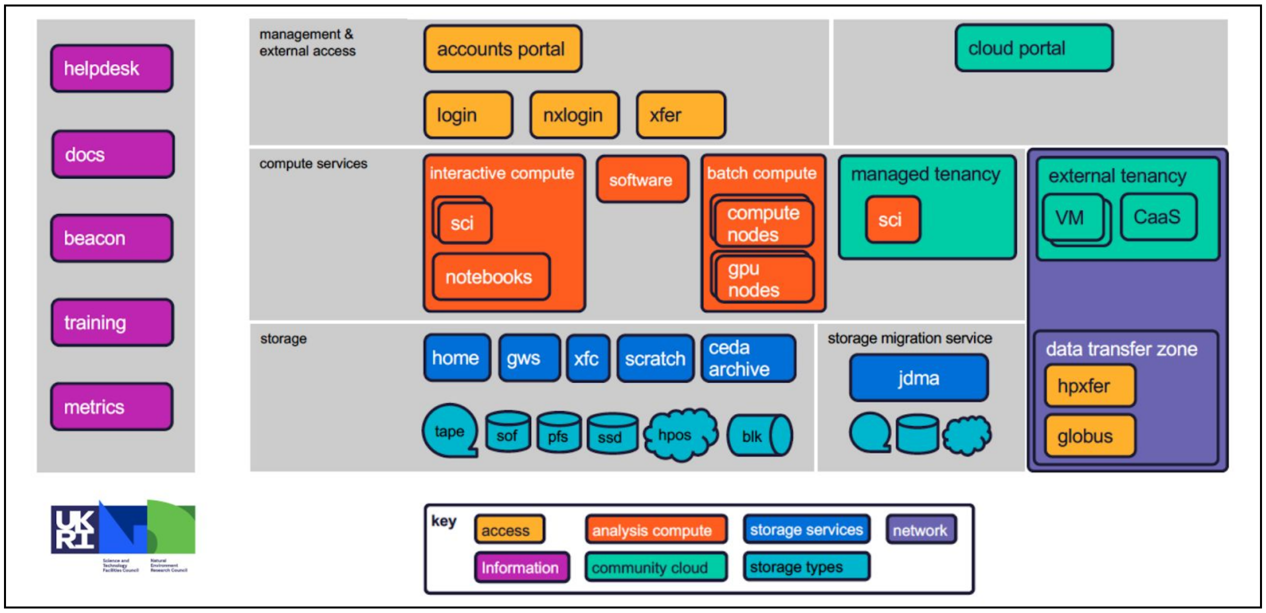

Figure 10. Overview of JASMIN services.

Figure 10 shows an overview of the services that are provided by JASMIN. The middle row shows the current compute services, which are accessible via three main interfaces:

- Interactive scientific analysis nodes: a range of server specifications (including high-memory servers) that are accessed via SSH
- Batch compute cluster (LOTUS): ~14,000 cores, including varied CPU/GPU configurations.
- Notebook Service: Jupyter Notebook environment with access to file systems/archive on JASMIN.

The data storage accessible to the compute services on JASMIN consists of:

- Archive access: ~13PB of curated data sets on disk.
- Home directories: 100GB per user.
- Group Workspaces: shared disk allocated on a per-project basis.
- Tape: used for backup and housing of infrequently used data.
- Object store: used for some archive and project-specific access.

A common set of software packages is available across the compute services listed above:

- Common Python and analysis packages: built using Conda.
- Additional packages: RPM-based (where Conda packages are not available).
- Compilers and libraries for parallel computation.
- Workflow management tools.

The JASMIN Cloud provides Cluster-as-a-Service (CaaS) which allows projects to develop their own software solutions with access to the JASMIN Object Store. The Cloud is built on OpenStack with Kubernetes employed to manage each deployment. The JASMIN Notebook Service is built on a similar Kubernetes configuration, providing automated scaling.

Current work concentrates on:

1. Development of the ESGF-specific WPS framework (built on Birdhouse) to provide CMIP (and other) data.
2. Loading of CMIP6 data into the JASMIN Object Store for access by users (via Intake catalogues):
    a. Using Xarray and Zarr[44]
    b. Using S3-netCDF[45]

The JASMIN compute services are funded through the national science programme and available to national scientists and their collaborators participating in that programme (e.g. through a funded grant award). Within IS-ENES3, access is being made available to a broader community both through the Trans-National Access scheme for access to servers and the Virtual Access scheme for access through Jupyter notebooks.

5.3.5 Compute Service repositories

ECAS service: https://ecaslab.cmcc.it/jupyter/hub/login doc: https://ecaslab.cmcc.it/web/home.html repo: https://github.com/ECAS-Lab
Ophidia doc: http://ophidia.cmcc.it/ repo: https://github.com/OphidiaBigData *PyOphidia* repo: https://github.com/OphidiaBigData/PyOphidia
Birdhouse WPS framework doc: https://birdhouse.readthedocs.io/en/latest/ repo: https://github.com/bird-house security proxy: https://github.com/bird-house/twitcher

[44] Xarray and Zarr: http://xarray.pydata.org/en/stable/generated/xarray.open_zarr.html
[45] S3-netCDF: https://github.com/cedadev/S3-netcdf-python

ESGF-specific WPS framework under development for C3S prototype repos under development at: https://github.com/roocs WPS: https://github.com/roocs/rook Example notebooks to interact with the "rook" WPS: https://rooki.readthedocs.io/en/latest/notebooks/index.html underlying library: https://github.com/pydata/xarray
CliMAF repo: https://github.com/rigoudyg/climaf doc: https://climaf.readthedocs.io/en/master/
Third party components: - *JupyterHub:* https://jupyter.org/hub - *xarray: http://xarray.pydata.org/en/stable/* - *Synda: https://github.com/Prodiguer/synda* - *Sproket: https://github.com/ESGF/sproket*

Table 4. Software repositories related to the computing services.

6 Identity Management and Access Entitlement

Under the ESGF future architecture initiative, a new implementation of the identity and access entitlement (IdEA) system has been underway since March. Significant progress has been made with individual components but none are yet at a stage where they can be integrated with the rest of the current ENES CDI as described in this document.

6.1 Current Status for Authentication and Authorisation with ENES CDI

This can be described as follows:

- Systems requiring authentication and authorization use the existing legacy ESGF system based on OpenID 2.0 and short-lived user X.509 certificates for authentication and SAML interfaces for authorisation.
- Some services, notably the Climate4Impact Portal take advantage of OAuth 2.0 for delegation of authentication.
- In some cases, where simple authentication is required, GitHub’s OAuth 2.0 service is used.

6.2 Implementation status for Future Architecture Components

We review in turn each of the components described in the architecture document D10.1. The new components for the Future Architecture [5] are all being implemented for deployment as Docker containers. Deployment may be made in one of two ways:

1. Using an Ansible Playbook to deploy a Docker image onto a single host using Docker Compose
2. Using a Helm Chart to deploy the Docker image into a Kubernetes cluster.

Option 1. involves a simpler more traditional deployment strategy, deploying containers on individual hosts. For option 2., container deployment and operation is managed by the Kubernetes orchestration system. Kubernetes manages the placement of containers on nodes in its cluster and keeps track of container health. The deployment itself is configured using Helm Charts[46].

6.2.1 Authentication, single sign-on and user delegation

The new system adopts the OAuth 2.0 framework for user delegation and OpenID Connect for single sign-on use cases. These also enable authentication using tokens passed in HTTP request headers and provide a simpler alternative to X.509 client certificate-based authentication used in the original ESGF system for command line use cases.

6.2.2 IdP Proxy and Federation Site IdP Implementations

The Identity Provider (IdP) Proxy is a special arrangement of the traditional model of Identity Provider ⇔ Rely Party pattern for single sign-on. The Proxy provides an intermediary between Relying Parties (in this case, ENES CDI services requiring authentication and authorisation) and IdPs. Supported IdPs include those sites in the federation wishing to host such services and also a number of external commercial IdPs such as Google and GitHub. Both IdP Proxy and federation site IdP are based on customisations of the open source implementation Keycloak from RedHat.

1. **IdP Proxy**: implementation complete; Docker image completed; Deployed for integration testing on JASMIN. Target deployment on AWS
2. **Federation site IdP**: close to completion. Docker image

6.2.3 Relying Party and Policy Enforcement Point Implementation

A Relying Party (RP) is a component that implements the interactions necessary with an IdP to secure a given service enforcing authentication with single sign-on. A Policy Enforcement Point (PEP) is a component that *enforces* authorisation access control decisions for a service. The PEP

[46] https://helm.sh/

refers to a Policy Decision Point (PDP) or authorisation service in order to make the access control *decisions* themselves. PDPs make decisions based on user attributes, access policies related to resources and in some cases, other factors related to the environment, for example access restricted to certain temporal constraints. Both PEP and RP lend themselves well to a filter architectural pattern in which they front access requests to the application to be secured and enforce the access constraints.

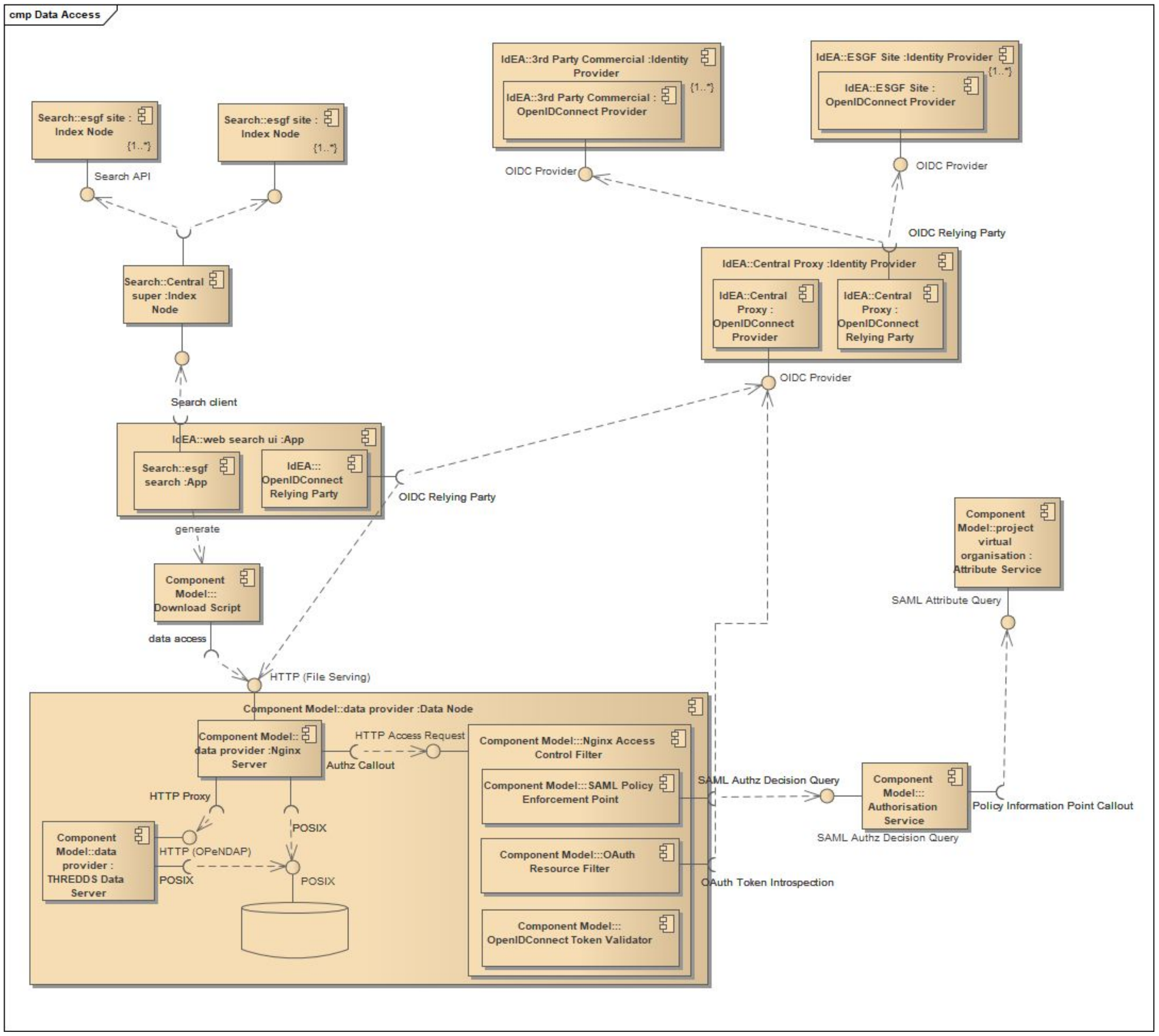
cmp Data Access
IdEA::3rd Party Commercial :Identity Provider
IdEA::3rd Party Commercial : OpenIDConnect Provider
{1..*}
IdEA::ESGF Site :Identity Provider
IdEA::ESGF Site : OpenIDConnect Provider
{1..*}
Search::esgf site : Index Node
{1..*}
Search::esgf site : Index Node
{1..*}
Search API
OIDC Provider
OIDC Provider
OIDC Relying Party
Search::Central super :Index Node
IdEA::Central Proxy :Identity Provider
IdEA::Central Proxy : OpenIDConnect Provider
IdEA::Central Proxy : OpenIDConnect Relying Party
OIDC Provider
Search client
IdEA::web search ui :App
Search::esgf search :App
IdEA::: OpenIDConnect Relying Party
OIDC Relying Party
generate
Component Model::project virtual organisation : Attribute Service
Component Model::: Download Script
SAML Attribute Query
data access
HTTP (File Serving)
Component Model::data provider :Data Node
Component Model:: data provider :Nginx Server
HTTP Access Request
Authz Callout
Component Model:::Nginx Access Control Filter
Component Model:::SAML Policy Enforcement Point
SAML Authz Decision Query
Component Model::: Authorisation Service
Policy Information Point Callout
SAML Authz Decision Query
HTTP Proxy
POSIX
Component Model::data provider : THREDDS Data Server
HTTP (OPeNDAP)
POSIX
POSIX
Component Model:::OAuth Resource Filter
OAuth Token Introspection
Component Model::: OpenIDConnect Token Validator

Figure 11. Data Node showing the integration of identity management and access entitlement components.

In the existing ESGF IdEA system, RP and PEP components are implemented on a per-application basis. In the new architecture they have been implemented as filter components integrating with the popular Nginx web application server. This recognises the fact that in nearly every case, ESGF services run over HTTP. Further, with the move to support Kubernetes, Nginx can be configured as the Kubernetes Cluster Ingress Controller.

- Nginx auth plugin: implementation based on Python Django web framework. Implementation complete; Docker image complete; Ansible-Docker deployment has been deployed and tested, Kubernetes deployment still to be tested

6.2.4 Federated Authorisation

As stated above (section 6.1), the existing legacy ESGF system uses SAML interfaces for authorisation interactions. The main actors are the PEP (See component inside Nginx Access Control Filter in Figure 11), PDP (aka. Authorisation Service, bottom right in Figure 11) and Attribute Service. In the new system, the per-application PEPs are replaced by a generic PEP deployed as part of the Kubernetes Ingress Controller (Nginx) or if not using Kubernetes, via a standalone Nginx deployment

1. PEP (implemented in Nginx auth plugin - see preceding section). Implements a SAML AuthzDecisionQuery callout to PDP to get authorisation decisions
2. Authorisation Service (PDP). Implementations in Java and Python. Supports SAML AuthzDecisionQuery interface. Includes callout to Attribute Service using SAML AttributeQuery interface. Contains user-configurable authorisation policy file
3. Attribute Service: implementation in Java and Python. Supports SAML AttributeQuery interface.

In the above, the Authorisation Service calls out to a virtual organisation Attribute Service instance in order to query for user's VO-level attributes required in order to make an authorisation decision. This is effectively a pull-model for attribute management. As a future enhancement, this model may be replaced with a push-model in which a Relying Party obtains user attributes from the IdP Proxy during the sign in process. These can then be passed via the PEP to the Authorisation Service so that the latter can make authorisation decisions.

6.2.5 Integration Status

Components for the new architecture have been integrated in a test configuration with the ESGF Data Node. Work is planned to test the Index Node's publisher API with the new access control. With these core ESGF components integrated, work must be undertaken to integrate the components in the ENES CDI with the new access control system. In summary, the IdEA system is in a state of active development at the time of writing. The existing legacy system is in place operationally but over the coming months as development is completed, the new system will be fully integrated into the ENES CDI.

IdP Proxy service (integration testing): https://auth-test.ceda.ac.uk/auth/realms/esgf/login-actions/authenticate?execution=ee6edfb0-a79e-4cd4-839c-fead8885e62c&client_id=security-admin-console&tab_id=LTGDdNd0cts doc: https://github.com/watucker/esgf.github.io/blob/idp-user-accounts/idp_user_accounts.md and https://github.com/watucker/esgf.github.io/blob/idp-user-accounts/idp-registration.md repo: https://github.com/ESGF/esgf-idp-proxy-theme
ESGF Site Identity Provider doc: repo: https://github.com/ESGF/esgf-idp-theme-example
Nginx Auth Plugin (RP and PEP implementation) doc: repo: https://github.com/cedadev/django-auth-service
Authorisation Service (PDP) doc: repo: https://github.com/ESGF/esgf-security (Java); https://github.com/cedadev/ndg_xacml (Python)
Attribute Service doc: repo: https://github.com/ESGF/esgf-security (Java) https://github.com/cedadev/ndg_saml (Python)
Third party components: - *ESGF Docker repo: https://github.com/ESGF/esgf-docker* - *Keycloak: https://www.keycloak.org/* - *Nginx: http://www.nginx.com/* - *Docker: https://www.docker.com/* - *Kubernetes: https://kubernetes.io* - *Helm: https://helm.sh/*

Table 5. Software Repositories related to Identity and Access Entitlement Services.

7 Conclusions and main targets of the next release

The main achievements of the first release of the ENES CDI are the increased quality of all of the pre-existing software components, a first step towards a fully interconnected infrastructure and the refined view of the next steps to be pursued. Some software has been consolidated in official release versions, while new services are being designed and developed. Adoption of containerisation, Cloud Computing and hosting, the latter provided sometimes by commercial parties, are also gaining attention (ie. for JASMIN, Identity Management Services, Climate4Impact). Specification of metadata and conventions underwent major improvements, with the associated discussion boards delivering documentation and repositories, where progress can be browsed and traced. This resulted in producing a first collection of compliant software tools, for validation and management of the metadata.

Important targets for the next release will address the interconnections between the different components, their capability to scale-up accommodating larger data-streams and computations, with better usability. These include the improved curation of the ESGF PID collection, especially concerning consistency and publication of the PID metadata through better tooling and automation of the attribution. Interactive services such as Climate4Impact and the Analytics-Hub will undergo major improvements in terms of their user-facing interfaces, search and computational capabilities, especially addressing the integration of more flexible development environments based on notebooks (JupyterLab), These can be delivered as part of advanced reproducible workspaces (SWIRRL), allowing execution of workflows, with automated provenance recordings and versioning of the users' methods and computational contexts [3].

We will also consider further updates to the architecture, which will take into account progress made with the integration of components produced in other WPs. For instance, we will pursue the connection of Climate4Impact to the model evaluation system based on the ESMValTool[47] (M7.3), to provide access to relevant metrics in support of the selection of model data. This is in cooperation with WP9/JRA2. Data Statistics service and Data Request schema will engage closely with their communities to achieve, respectively, better representation of the information, through more specific views, and increased flexibility in accommodating different high-level requests (M10.2 [9]). Finally, the coming release will see also the improvement of the Metadata for Climate Indices. These will be refined and further developed in the upcoming workshop (2021). Progress will be reported in the second (D10.3) and the third release (D10.5) of the ENES-CDI.

[47] https://cmip-esmvaltool.dkrz.de/

8 References

[1] S. Fiore, et al. *D10.1 - Architectural document of the ENES CDI software stack,* https://zenodo.org/record/4309892#.X9CSbS2ZMn1

[2] Rule, A., et al. (2018). Ten Simple Rules for Reproducible Research in Jupyter Notebooks. http://arxiv.org/abs/1810.08055

[3] Goble, Carole, et al. "FAIR computational workflows." Data Intelligence 2.1-2 (2020): 108-121. https://doi.org/10.1162/dint_a_00033

[4] Wilkinson, Mark D., et al. "The FAIR Guiding Principles for scientific data management and stewardship." *Scientific data* 3.1 (2016): 1-9.

[5] Kershaw, Philip, Abdulla, Ghaleb, Ames, Sasha, & Evans, Ben. (2020, July 2). ESGF Future Architecture Report (Version 1.1). Zenodo. http://doi.org/10.5281/zenodo.3928223

[6] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, “A security architecture for computational grids,” in Proceedings of the 5th ACM Conference on Computer and Communications Security, ser. CCS ’98. New York, NY, USA: ACM, 1998, pp. 83–92.

[7] R. Alfieri, et al, “Voms, an authorization system for virtual organizations,” in Grid Computing, First European Across Grids Conference, Santiago de Compostela, Spain, February 13-14, 2003, Revised Papers, ser. Lecture Notes in Computer Science, vol. 2970. Springer, 2004, pp. 33–40

[8] Lin, D., Crabtree, J., Dillo, I. *et al.* The TRUST Principles for digital repositories. *Sci Data* **7,** 144 (2020). https://doi.org/10.1038/s41597-020-0486-7

[9] Juckes, M. IS-ENES3 Milestone M10.2 CMIP Data Request Schema 2.0. https://is.enes.org/documents/milestones/m10-2-cmip-data-request-schema-2.0/view