

D5.4: Integrated Data Management Workflows

Author(s)	Ari Lukkarinen (CSC), Anne Fouilloux (UiO), Hamish Struthers (SNIC), Tommi Bergmann (FMI), Risto Makkonen (FMI), Jean laquinta (UiO), Helmut Neukirchen (UICE), Ernir Erlingsson (UICE), Adil Hasan (Sigma2)
Status	Final version
Version	1.0
Date	10/11/21

Document identifier:	
Deliverable lead	CSC
Related work package	WP5
Author(s)	Ari Lukkarinen (CSC), Anne Fouilloux (UiO), Hamish Struthers (SNIC), Tommi Bergman (FMI), Risto Makkonen (FMI), Jean laquinta (UiO), Stephan Oepen (UiO), Joerg Tiedemann (UH), Adil Hasan (Sigma2)
Contributor(s)	
Due date	31/08/21 (delayed to 15/11/21)
Actual submission date	
Reviewed by	Helmut Neukirchen (UICE), Anne Sofie Fink Kjeldgaard (DeIC)
Approved by	
Dissemination level	PU
Website	
Call	
Project Number	
Start date of Project	
Duration	
License	
Keywords	

Abstract:

This deliverable aims to investigate the relevant EOSC and national services that can be leveraged to support cross border data management, data-driven distributed computing and research workflows for two research communities (Climate and Natural Language Processing). Emphasis is placed on fostering data aggregation and interlinking of sharing and active data storages.



Copyright notice: This work is licensed under the Creative Commons CC-BY 4.0 licence. To view a copy of this licence, visit <https://creativecommons.org/licenses/by/4.0>.

Disclaimer: The content of the document herein is the sole responsibility of the publishers and it does not necessarily represent the views expressed by the European Commission or its services.

While the information contained in the document is believed to be accurate, the author(s) or any other participant in the EOSC-Nordic Consortium make no warranty of any kind with regard to this material including, but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Neither the EOSC-Nordic Consortium nor any of its members, their officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

Without derogating from the generality of the foregoing neither the EOSC-Nordic Consortium nor any of its members, their officers, employees or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

Table of Contents

Table of Contents	3
Table of Abbreviations	5
1. Introduction	7
2. Use Cases	7
2.1 Use Case 1: Climate	8
2.1.1 Presentation of the Nordic Climate Community	8
2.1.2 Current practices and activities	8
Description of each activity	8
Activity-1: Using Earth System Model data	8
Activity-2: Developing Earth System Models	9
Activity-3: Running Earth System Models to produce climate data	10
2.1.3 Identified Issues	11
From a researcher's point of view	11
From an infrastructure provider's point of view	11
2.1.4 Solutions and Services Deployed Within EOSC-Nordic	12
Activity-1: Role of EOSC for using Earth System Model data	12
Before EOSC-Nordic	12
Role of EOSC-Nordic	12
Shared workflows for climate model data processing: test case description	12
(I) EGI, CSC Notebooks and BinderHub	13
(II) The NIRD Service platform and its Jupyter ecosystem	15
(III) Galaxy and its Jupyter ecosystem	15
Activity-2: Role of EOSC-Nordic for facilitating the development of Earth System Models	17
Before EOSC-Nordic	17
Role of EOSC-Nordic	18
Activity-3: Role of EOSC-Nordic for running Earth System Models to produce climate data	18
Before EOSC-Nordic	18
Role of EOSC-Nordic	19
2.1.5 Pending Issues and Future Work	23
Roadmap and solutions to be tested within EOSC-Nordic	23
Beyond EOSC-Nordic	23
2.2 Use Case 2: Nordic Language Processing Laboratory (NLPL)	24
2.2.1 The NLPL Virtual Laboratory	25
2.2.2 Data Replication across HPC Systems	25

2.2.3 Management of Access Rights	27
2.2.4 Data Staging from “Off-line” Storage	28
2.2.5 Very Large-Scale Data Management	29
3. Lessons Learned	29
3.1 Climate	29
3.2 NLPL	30
4. Summary and Future Work	31

Table of Abbreviations

Abbreviation	Explanation
AI	Artificial Intelligence
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
CDO	Climate Data Operators
CESM	Community Earth System Model
CF	Climate Forecast metadata convention
CMIP	Coupled Model Intercomparison Project
CMOR	Climate Model Output Rewriter
CPU	Central Processing Unit
CSC	IT Center for Science Ltd
EGI	European Grid Infrastructure
EODC	Earth Observation Data Centre
EOSC	European Open Science Cloud
ESM	Earth System Model
ESMValTool	Earth System Model Validation Tool
EUDAT	European Collaborative Data Infrastructure
FAIR	Findable, Accessible, Interoperable, Reusable
FATES	Functionally Assembled Terrestrial Ecosystem Simulator
FTP	File Transfer Protocol
GCM	Global Climate Model
GPU	Graphical Processing Unit
GTN	Galaxy Training Network
GUI	Graphical User Interface

HPC	High Performance Computing
HPC-cloud	HPC resources in the cloud
HTC	High Throughput Computing
JSON	JavaScript Object Notation
LUMI	Large Unified Modern Infrastructure
NCAR	National Center for Atmospheric Research
NCL	NCAR Command Language
NCO	netCDF Operators
netCDF	Network Common Data Format
NIRD	National Infrastructure for Research Data, Norway
NLP	Natural Language Processing
NLPL	Nordic Language Processing Laboratory
NorESM	Norwegian Earth System Model
RCM	Regional Climate Model
RO	Research Object
STAC	Spatio Temporal Asset Catalogue
TlaaS	Training Infrastructure as a Service
TB	Terabyte
URL	Uniform Resource Locator
VLLM	Very Large Language Model
XML	Extensible Markup Language

1. Introduction

Research is driven by an ever-increasing number of datasets that may be difficult and/or expensive to reproduce, for instance observations or large simulations. These datasets may also be very diverse and complex. At the same time, researchers have access to a wide range of computing resources (laptop or personal computers, cloud computing, HPC-cloud and HPCs) opening new opportunities in terms of resource use or data re-use. However, this potential is not necessarily realized because it is difficult to re-use data without:

- the participation of the data creator(s) since they retain (both intentionally and unintentionally) information regarding the meaning of their data that is required to enable re-use by other researchers, even those working within the same research group, as well as from external partners both within the same community or for cross-disciplinary research;
- the tools used for producing and/or analyzing the data or simply for accessing metadata (for instance netCDF climate data contains metadata that are embedded in the data itself);
- the storage and computing resources necessary for handling the data.

Integrating data, tools and resources (computing and storage) is necessary but only makes sense if the datasets/tools/resources context as well as feedback on their usage are also captured: this is what we mean by “Integrated Data Management Workflow”.

In this deliverable, we demonstrate that handling separately the management of data, associated tools and the necessary computing and storage resources partially meets researchers’ expectations but can hinder Open Science even though data itself may be “declared” FAIR (as defined by FAIR principles).

We analyzed the current offering in terms of services and provided solutions to support cross borders data management and data-driven distributed computing for two use cases: climate modelling and language analysis. Lessons learned are also documented and whenever possible we highlight pending issues, in particular when not tackled by EOSC.

2. Use Cases

Two specific use cases are considered in this project:

- the Climate Modelling use case where different strategies for “decoupling” data access, tools and computing resources have been explored;
- the Language Analysis use case where centralized data storage has been set up and associated computing resources provided.

These two use cases are presented in the next section and for each use case we discuss:

- Current practices in place at the start of the EOSC-Nordic project;
- Issues identified either by service providers or researchers;
- Solutions and services deployed (or to be deployed) within EOSC-Nordic;
- Pending issues and possible future work which exceed the scope of the project.

2.1 Use Case I: Climate

2.1.1 Presentation of the Nordic Climate Community

The Nordic climate modeling community consists of research groups at universities, national meteorological institutes and research institutes, and holds demonstrable world class excellence in the field. Several of these groups contribute to the development of both global and regional climate models (GCMs and RCMs, respectively) in international projects with collaborations within Europe and the US. On the one hand, many users, including PhDs and postdocs, are developing and/or running Earth System Models (ESMs) for more fundamental research and sensitivity studies, and/or cross-disciplinary research (economy and climate, biodiversity, etc.) and, because they are not involved in international projects, they do not necessarily benefit from any of the advances, technologies or resources leveraged by these large projects/consortiums. On the other hand, the most significant scientific progress is often taking place in fundamental research projects.

2.1.2 Current practices and activities

The Earth's climate is a very complex system and understanding and/or predicting how it will change is not an easy task. It requires the involvement of a large and varied number of "actors" with different backgrounds and scientific interests: e.g. numerical modelling specialists, Earth's atmosphere researchers, oceanographers, hydrologists, atmospheric chemists, marine biologists, field ecologists, cryospheric scientists, physicists, computer scientists, etc.

The time to transfer knowledge from/to academic researchers, applied researchers, decision-makers and the public is too long. From a practical point of view this translates into a lot of time/energy wasted "reinventing the wheel", repeated simulations, lack of transparency, suboptimal use of the infrastructures and ineffective re-use of existing data, etc. In this context supporting these researchers and realizing the benefits of Open Science should be a priority for EOSC. Offering a usable and reliable modelling framework suitable for all such actors is the challenge we tried to tackle within EOSC-Nordic.

Three key climate modelling activities have been identified and for each of them, we detail the analysis of the current practices, the solution explored within EOSC-Nordic and finally pending issues that would require significant additional funding to be properly addressed:

- **Activity-1:** Using Earth System Model data;
- **Activity-2:** Developing Earth System Models;
- **Activity-3:** Running Earth System Models to produce climate data.

Description of each activity

In this section, each activity is described and a general overview of the needs is highlighted.

Activity-1: Using Earth System Model data

In this activity, users need access to existing ESM data in order to perform analyses. A wide range of applications involves analysis of climate data produced by ESMs. A few generic examples are given below:

- Analyzing and comparing existing ESM simulation outputs (model inter-comparison projects such as CMIP¹, ad-hoc simulations where a control experiment is compared to a newly tested “scenario”, etc.)
- Analyzing and comparing model outputs against observations;
- Analyzing and “re-formatting” ESM simulation outputs to be used as input for different models or studies (e.g downscaling, transport modelling/tracking, climate impacts and mitigation);
- Running/developing diagnostic and performance metric tools for routine evaluation of ESMs (such as in CMIPs).

Main requirements:

- Access to potentially large amounts of data with minimal waiting time if downloading (copy), or transfer (streaming);
- Ability to track data versions using for example persistent identifiers;
- Access to some potentially restricted data, for instance observations or ESM outputs that are not published yet;
- Availability of a wide variety of tools for processing ESM data e.g. standard climate tools such as CDO², NCO³, ESMValTool⁴ and programming environments such as Python⁵, R⁶, MATLAB⁷, Julia⁸ and IDL⁹ with all the related climate libraries/packages;
- Possibility to test new tools/packages/libraries;
- Ability to “trace” tools/computing environment used (management of the versions of all the software);
- Heterogeneous computing needs ranging from single processor to High Throughput Computing (HTC) resources, with potentially large amounts of memory;
- Need to share ad-hoc tools/scripts such as Jupyter notebooks.

For this activity, we have explored the Jupyter¹⁰ ecosystem as it seemed to provide a flexible and configurable environment for running potentially large HTC analysis.

Activity-2: Developing Earth System Models

ESMs are very complex but also very modular and can be run at various resolutions (from single column to global high-resolution) and configurations (from using a single model component to fully coupled configurations). Researchers adopt different strategies for developing climate models depending on their scientific goals and/or habits/knowledge, for instance:

- Development of a new parameterization, addition of new “species” (aerosols, land surface use, etc.) including introduction of AI to replace existing parameterizations or physical/chemical representations;

¹Eyring, V. et al. 2016. Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization, Geosci. Model Dev., 9, 1937–1958

² <https://code.mpimet.mpg.de/projects/cdo/>

³ <https://sourceforge.net/projects/nco/>

⁴ <https://www.esmvaltool.org/>

⁵ <https://www.python.org/>

⁶ <https://www.r-project.org/>

⁷ <https://www.mathworks.com/products/matlab.html>

⁸ <https://julialang.org/>

⁹ <https://www.l3harrisgeospatial.com/Software-Technology/IDL>

¹⁰ <https://jupyter.org/>

- Refine interactions within the Earth System (exchange new fields, add new components and feedbacks, including non-chemical/physically based processes like socio-economical interactions);
- Increase model component resolution, or any other grid change (nested grids) which may require substantial changes to the resolved dynamics along with commensurate changes to physical/chemical parameterizations.

Main requirements:

- Rapid turn-around when developing new parameterizations (changing the code, compiling and testing);
- Access to reference inputs and outputs for testing purposes;
- A set of benchmark tests and procedures to facilitate checking and validation;
- Software stack consistent with operational environment (same compilers and libraries);
- Ability to share model development with non-modeller experts to provide new insight, test and validate new parameterizations;
- Full control on who can access newly generated/collected data and code changes while developing;
- Need to easily share code changes when publishing scientific papers.

This activity is the most complex in terms of requirements since all the actors may not have the same level of expertise with, e.g, programming and the used software tools. Most of the tasks are “manual/bespoke” and must be constantly “monitored” by the researchers and therefore cannot be easily automated.

For this activity, we have explored the use of containers to provide a consistent software stack for development (similar to production software environment) along with the JupyterLab environment so that end-users can develop/test/debug/validate on-the-fly on a single “platform”.

Activity-3: Running Earth System Models to produce climate data

This activity requires ESMs to be ready to use in “production”, e.g. for running large configurations and long simulations in a fully reproducible manner. For instance:

- Running an ESM to simulate specific events/scenarios (changing CO₂, or more generally forcings or other model inputs, generating volcanic eruptions, experiment with geoengineering) or for systematic sensitivity studies: running the same source code with different parameters;
- Running spun up simulations and establishing reference ESM datasets for subsequent runs (to be done for each model version);
- Running an ESM in an intercomparison framework using a well defined protocol.

Main requirements:

- Access to HPC or HPC-cloud is generally needed with good interconnect and fast/large storage due to the necessary communication between processes and the large volume of data produced, but still significant flexibility in terms of optimal number of processors (depending on the configuration and time frame for delivery);
- Input data for running the ESMs must be ready to use, e.g. stored on disk and accessible from the compute platform;
- Simulations need to be fully reproducible and could be automated using Workflow Management Systems;
- Standardization of ESM outputs may be required for publishing data, such as CMORization for CMIPs, CF-conventions and/or other metadata to facilitate long-term archiving;

- Need to “assign” a unique persistent identifier for the data generated with the possibility of releasing new versions/errata;
- The outputs of the simulations have to be discoverable and accessible by other users including relevant metadata and catalog/registry, related searchable documentation for instance with a description of the variables, etc.

For this activity, we have tested the use of Galaxy¹¹ tools and composition of workflows for running Earth System Models (production mode) to be executed on heterogeneous compute platforms.

2.1.3 Identified Issues

In this section, we summarize the main obstacles we have identified for on-boarding researchers on EOSC, independent of the activity. In the next sections, we will detail how we have tackled these issues for each activity and highlight the role of EOSC.

From a researcher’s point of view

Choices may not be driven by researchers’ needs and/or current available technologies but by:

- a previous poor/good experience with a tool and/or type of resource;
- current practices and habits in a research group/lab/department;
- complexity of the tools to be used (for instance Earth System Models);
- lack of knowledge of best software practices for using/developing/sharing tools. For instance poorly written Jupyter notebooks are usually not reusable;
- lack of time for investigating/testing/learning new solutions, especially when there are no incentives and no clear benefits for the researcher(s);
- misinformation on the potential use of existing resources and services as they are often presented outside any research context;
- lack of trust in new technologies, especially when presented by “technical experts” outside their field of research;
- difficulty to understand how produced data/software could be reused in a different context, no feedback collected and reported to the data creator, and no incentive to do so. Except for CMIP experiments, most researchers produce data for a single research project;
- no clear guidance on what resources to use and how to generate/store/archive data beyond a few mandatory metadata and data format standards;
- lack of automation of research workflows. No usage of Workflow Management Systems to optimize workloads.

From an infrastructure provider’s point of view

Issues with service provision can often be traced to a lack of understanding and limited engagement with research communities. For example:

- providers collect feedback from research project managers and not from actual users of the resources;
- lack of understanding of practical issues faced by researchers;
- huge gap between too simplistic examples/test cases and real world use cases when deploying a new service or technology. For example, providing a mere “hello world” example for running

¹¹ <https://galaxyproject.org/>

11

containers on a single node whereas end-users need an example for running complex applications at scale on multiple nodes;

- limited knowledge of the appropriate services/resources to deploy, depending on the size/scale of the problem to solve. For instance, depending on the configuration/resolution, the same ESM can require tens of thousands of cores, fast interconnect and IO or be run on a laptop with a single processor;
- no global understanding of the entire workflow which, e.g, comprises data preparation, staging, running a model, post processing, visualization and publication of the results. Typically, data wrangling is ignored whereas it represents the vast majority of the time spent (and sometimes wasted) by researchers;
- the balance in trying to accommodate the needs of every single user against the time and complexity of having to install/maintain/support a very large and diverse software stack.

2.1.4 Solutions and Services Deployed Within EOSC-Nordic

Solutions have been tested within the three different identified activities, and below we detail the work done for each of these use cases.

Activity-1: Role of EOSC for using Earth System Model data

Before EOSC-Nordic

As mentioned earlier, the Jupyter ecosystem is very flexible and many researchers are familiar with it. Before the project started, some researchers were using Jupyter notebooks mainly from their laptop and mostly for prototyping or teaching with Python. Many researchers were using tools such as NCL¹², CDO, NCO, MATLAB, Python (but not with Jupyter notebooks) from HPC in batch mode, and/or from dedicated post-processing clusters.

Role of EOSC-Nordic

The main objective of this task was to explore the use of the Jupyter ecosystem as an EOSC service to solve some of the issues identified by researchers, such as a reduction of data movement and more flexible post-processing compute resources.

Several EOSC services based on the Jupyter ecosystem are already available:

- EGI notebooks (<https://marketplace.eosc-portal.eu/services/egi-notebooks>);
- EODC JupyterHub for global Copernicus data (<https://marketplace.eosc-portal.eu/services/eodc-jupyterhub-for-global-copernicus-data>);
- Galaxy interactive tools with Jupyter notebooks, offered as part of <https://live.usegalaxy.eu/>.

It is clear that many other EOSC services based on the Jupyter ecosystem will be offered very soon, deployed by other EOSC projects. Therefore, EOSC services based on the Jupyter ecosystem for Nordic climate researchers seems an attractive approach.

For this activity, we tested both existing EOSC services and other services that are currently only deployed at national level:

- I. EGI Notebooks, CSC Jupyter notebooks (<https://notebooks.csc.fi/>) and [BinderHub](#)
- II. The NIRD Service platform (<https://apps.sigma2.no/>)
- III. Galaxy Jupyter notebooks including UCloud (<https://docs.cloud.sdu.dk/Apps/jupyter-lab.html>).

¹² <https://www.ncl.ucar.edu>

All these services are very similar and generally based on Docker containers.

Shared workflows for climate model data processing: test case description

Our test case on climate data management and cross-border data processing workflows was constructed around the following assumptions:

- a team consisting of researchers from several countries should be able to process large shared datasets produced by climate models;
- the processing workflows should be reproducible, shareable and developed in collaboration;
- the data itself is well-structured in self-describing NetCDF format with model-generated metadata.

The use case can represent, e.g., a phase during an international collaborative project where climate model data produced early in the project is being utilized by several partners throughout the project implementation. For example, European research projects might provide a joint data infrastructure for project data, but shared data processing workflows are typically less prioritized. Figure 1. presents an example of such a workflow, integrating code development (for data processing and analysis) and data repositories (observations, model data).

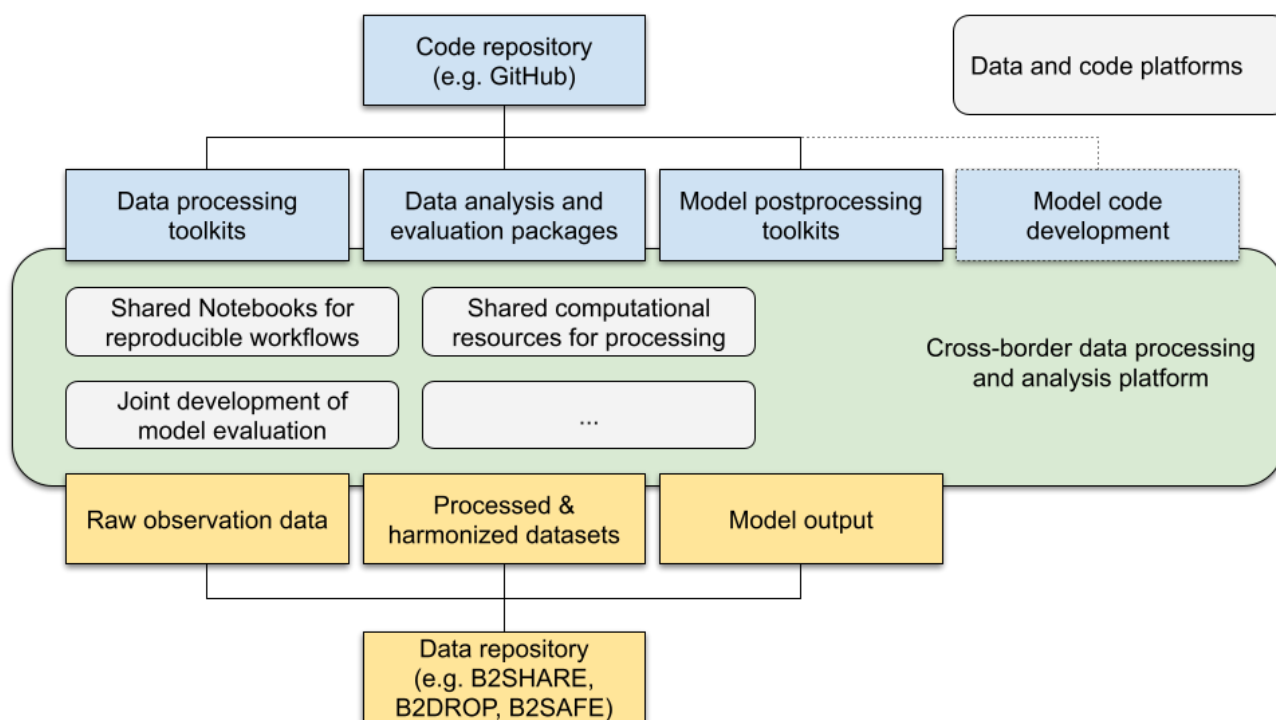


Fig. 1: Schematic example of integrating code repositories (top of figure) and data repositories (bottom of figure) in joint climate model development and analysis.

(I) EGI, CSC Notebooks and BinderHub

Our aim was to utilize existing EUDAT platforms for data storage for this use case. However, we found several bottlenecks in functionalities provided by the tools, especially for managing data sharing for a team of researchers. For example, team collaboration using the "Circles" functionality in EUDAT¹³ remains extremely limited and creates the potential for mistakes in data sharing. The services do not allow for generating an invite link to the Circle, which could be shared to the team. Instead, all members need to be

¹³ <https://eudat.eu/>

added one by one in the B2DROP¹⁴ web platform. But adding a team member by name is a significant risk: the webapp does not show any other identifier (e.g. email) than the name and there is no way to distinguish people with identical names. Searching by email is safer, but a user has potentially several accounts, which again can not be distinguished in the webapp. There are several cases where users share restricted data to *all users* by mistake. These issues have been reported to the EUDAT helpdesk. The lack of efficient and reliable team management in EUDAT's B2-services is a key limitation to, e.g., B2DROP usage in our use case.

Instead of EUDAT, our test case utilized an S3 data lake implementation at CSC¹⁵ to store the original climate model data. This specific data lake is in operational use and is integrated close to the HPC and other computational platforms, and hence provides a swift workflow for the researcher: raw data from computations can be deposited to a public (or restricted) S3 share with tools integrated to the computing center packages. Similarly, the data could be transferred, e.g., to the EUDAT B2SHARE service and shared via S3, although this might mean an extra step for the researcher. Our case does not consider deposition of output data of the processing step. It is evidently not clear for most users how to balance resources between direct use of, e.g., S3 resources or temporary duplication of original data to a local file system when processing large datasets, and utilization of cloud resources in data processing which might make this optimization even more problematic. More support for researchers is needed for development of resource efficient workflows for cloud processing.

Being focused on data infrastructures, the EUDAT services do not include code repositories (Fig. 1). Some universities and research institutes provide their own platforms for code management. In our use case, we utilized GitLab for data workflow related code management versioning.

We tested several candidates for the computational platform that could be suitable for collaborative data processing workflows. These include EGI Notebooks¹⁶ (supported by EOSC-Hub), a notebook platform at CSC, as well as BinderHub¹⁷ environments. Our tests indicated that the EGI notebook platform reached an active user limit often, indicating either inadequate resources or continued heavy usage. Furthermore, the tests revealed that while EGI supports a multitude of identity providers, certain academic/research institutes are lacking required identifiers to create a personalized notebook environment. However, the EGI Notebook platform provided a long-lasting environment that could be reopened later. The notebook environment at CSC (<https://notebooks.csc.fi>) is open to Finnish researchers at universities and research institutes. The time limits of CSC notebooks (typically a few hours) might be acceptable for, e.g., course exercises but are too restrictive for many data-processing tasks.

While our climate data use case could be realized with an isolated code-repository and computing platforms, we further investigated the use of reproducible computing environments. BinderHub provides a platform for connecting source code (from source code repositories such as GitLab or from source code archived in data repositories such as Zenodo) to an executable cloud computing environment. We tested the implementation provided at [MyBinder.org](https://mybinder.org). MyBinder provided an extremely easy way to connect our code repository into a shareable computing environment, and the provided URL could be shared to a team. Development of data processing tools in a shared GitLab repository is automatically synchronized with the MyBinder environment.

When it comes to data access and cross-border data flows, all the above notebook environments understandably suffer from lacking suitable libraries and packages. For accessing S3 data, e.g. the S3FS¹⁸ package is recommended, and efficient climate model data typically also requires NetCDF and xarray

¹⁴ <https://sp.eudat.eu/catalog/resources/b7ef259b-2816-4ed2-bc8a-374ef6ed93b6>

¹⁵ <https://research.csc.fi/-/allas>

¹⁶ <https://www.egi.eu/services/notebooks/>

¹⁷ <https://github.com/jupyterhub/binderhub>

¹⁸ <https://fs-s3fs.readthedocs.io/en/latest/>

libraries. These packages needed to be installed in the notebook environment before our code execution, which creates possibilities of problems for library versions and limits reproducibility of the data workflow. The CSC notebook environment allowed selection of notebooks configured for distinct tasks, such as data science, which somewhat alleviated this issue.

Our use case underlines the need for better integration of HPC, data and data processing platforms, especially towards shared and reproducible workflows. Common authentication pathways, team management, support services for optimizing data processing in the cloud, and customized (notebook) environments for data science will help researchers transition to emerging platforms. Having a community support group that has knowledge and experience in integrating notebooks into climate research workflows and could work with researchers/groups would further facilitate uptake within the climate community.

(II) The NIRD Service platform and its Jupyter ecosystem

The Norwegian NIRD¹⁹ service platform went operational at the end of 2018. Single user JupyterLab or JupyterHub services can be deployed by all users having access to NIRD storage. Data access is “integrated” by default, e.g. when deploying the service, users can choose which folder to share and whether it is read-only or writable. With more recent versions, one can share folders from different user projects (provided they get an explicit authorization from the project owner).

Deploying JupyterLab is often complex (single-user), even through the NIRD Service Platform, especially with multiple projects and when users need to have their own container image with a bespoke list of software packages. Therefore, we have mostly focused on the deployment of JupyterHubs and tested it in the framework of an e-Science course that is regularly given (once a year) to researchers, PhDs and postdocs.

- 2019 (16 users): to access all the data from the JupyterHub, we copied all the necessary data into a single location and kept all files with their “native” formats (mostly netCDF). Because we copied some observations with restricted access, we could not give access to the same JupyterHub to users that had not signed/agreed to the data usage (private observations). Data and compute nodes are on the same infrastructure so data access was not problematic. However, most researchers complained about the lack of computing resources: it is indeed difficult to accommodate a large number of users, limited by the computing resources of a single provider.
- 2020 (21 users): instead of copying all data to a single location, users received access to all publicly available climate data stored on NIRD (authorization was granted by two different project owners) and we only copied data with restricted access (observations). From a user’s point of view there were no differences and controlling permissions/access to observations was not much improved since there is no control of permission at the user level (but only JupyterHub level): we still had to give access to observations to everyone who had been granted access to the JupyterHub. To prevent issues with restricted data, we deployed additional JupyterHubs (with identical software stack).

This approach offered some flexibility to deploy containers with a bespoke software environment, however the major problem was data management which is not suited to restricted access because it is not possible to specify granular file permission when sharing data. Also, deployment using a single provider (in this case NIRD) put the available resources under huge stress, which impacted other users of the services for the entire duration of the course, and it was not possible to use computing resources from other national providers on the same JupyterHub.

¹⁹ https://documentation.sigma2.no/files_storage/nird.html

(III) Galaxy and its Jupyter ecosystem

Galaxy provides a framework for deploying “interactive” tools such as Jupyter notebooks. The main difference to the NIRD Service Platform is that with Galaxy, end-users can configure and deploy any existing containers. Galaxy has three interactive tools based on Jupyter Notebooks that are of interest for the Climate modelling community: Interactive Jupyter Notebook, GPU-enabled Interactive Jupyter Notebook for Machine Learning, and Interactive Climate Notebook. They are all built following the same approach and all are openly available on GitHub (for instance see <https://github.com/NordicESMhub/docker-climate-notebook> for the Galaxy Climate docker container). All have different software stacks available and users can install additional tools and packages since users have root privileges inside the container. The user interface is the same for all interactive tools and it is relatively easy to use (see Fig. 2 below).

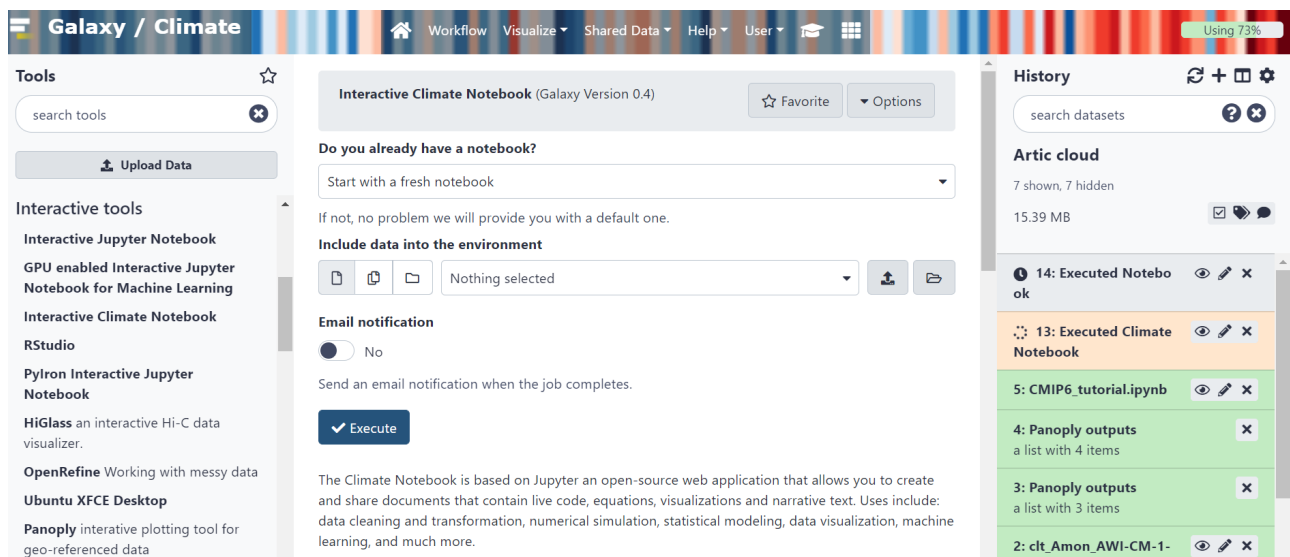


Fig. 2: Galaxy Interface (GUI) for starting a Jupyter Notebook. To start an interactive tool, the user would need to click on “Execute”.

Training material is available to help researchers to start and use JupyterLab on Galaxy (see <https://training.galaxyproject.org/training-material/topics/galaxy-interface/tutorials/jupyterlab/tutorial.html>). Our goal with the Galaxy Jupyter interactive tool is to improve the management of data. To achieve this we have investigated the following approach:

- Restricted data (observations) are uploaded by a single user (one of the course teachers) into a Galaxy history and this history is shared only with users who need access to these observations.
- Some shared and non-restricted data are copied by the Galaxy administrator into local Shared Data Libraries (<https://climate.usegalaxy.eu/libraries>) under the “Earth System community Modeling” category. However, copying all CMIP data necessary for the course (about 100 TB) is not possible, thus each end-user may have to upload the “remaining” necessary data in their Galaxy history and JupyterLab instance. Users can share their histories, but it is not straightforward to find data from published histories (https://climate.usegalaxy.eu/histories/list_published). In addition, this requires everyone to use the same Galaxy portal, for instance Galaxy Europe (<https://usegalaxy.eu/>). If using another Galaxy instance, users would need to first “Export History to File” as a tarball and then upload it to another instance (which is of course quite cumbersome).

- To facilitate data sharing, B2DROP (<https://b2drop.eudat.eu/>) has been tested (see <https://youtu.be/gVFL19l3NfU>), but has been considered superfluous by users as other, more familiar tools exist that serve the same purpose and B2DROP storage space is insufficient (20 GB);
- To ease access and facilitate post-processing, we are exploring the usage of S3-compatible object storage as our storage providers in Sweden, Finland and Norway are providing S3 access to their data. An instance of minio (<https://min.io/>) for accessing data stored on NIRD has been deployed to provide access to CMIP6 climate data (via anonymous access). An example on how to access CMIP6 data from Galaxy is available at <https://youtu.be/e9yHmF1pd3M>. End users can also directly access data via the s3fs Python package (see examples at <https://github.com/NordicESMhub/eosc-nordic-climate-demonstrator/blob/master/work/s3-tests/s3-minio-anonymous.ipynb> and https://github.com/NordicESMhub/eosc-nordic-climate-demonstrator/blob/master/work/s3-tests/nird_archive_demo.ipynb). Similar tests are currently being performed with UCloud Minio (<https://docs.cloud.sdu.dk/Apps/minio.html>) and CSC Allas Object storage (<https://docs.csc.fi/data/Allas/>). While very convenient, performance when accessing netCDF data (via streaming) is very often poor and not satisfying for users. To improve performance, we convert netCDF data into Zarr (<https://zarr.readthedocs.io/en/stable/>). The main advantage is that the user interface does not change and Zarr also provides `zarr.storage.LRUStoreCache` which can be used for implementing a local in-memory cache layer over the remote storage. This is particularly useful when “regularly” accessing the same dataset. When combined with an *intake catalog* (<https://intake.readthedocs.io/en/latest/catalog.html>), access to data becomes much more independent of its location (see for example https://github.com/NordicESMhub/eosc-nordic-climate-demonstrator/blob/master/work/owncloud/F2000climo_f19_g17_control.ipynb). Another way to define a catalog (which may be more appropriate for the Climate modeling community) is to create a STAC Catalog (<https://stacspec.org/>). In both cases, the code shared (for instance Jupyter notebooks) does not contain any relative/absolute path and is thus independent of the compute and storage environment. However, users need to share this intake/STAC catalog (a YAML or JSON file) and there is currently no registry for catalogs. ROHub (<https://www.rohub.org/>) could be used with one Research Object per catalog: this would significantly improve the FAIRness of climate data, including those with restricted access.

The main advantage of this solution, i.e. using S3 storage and cloud optimized formats (cloud optimized GeoTIFF, Zarr) is that it can be easily deployed into another Galaxy instance and/or the corresponding Docker container can be used as a standalone tool. The software stack is therefore identical and also independent of the infrastructure itself. However, different containers may need to be prepared (with the same recipe) for different architectures (x86-64 processors, ARMs, etc.).

Additional benefits are a fine granularity of permissions for file access (at a user level, group level, instance level, or public/anonymous access) and the possibility to add metadata. Several locations can be listed for the same dataset (local disks and remote locations), however access priority is defined by order of appearance in the catalog and the search stops as soon as data can be accessed, regardless of network speed. This works very well for prioritizing local access but may not be optimal when there are several replicas available remotely, and no use-closest strategy can be prescribed.

Activity-2: Role of EOSC-Nordic for facilitating the development of Earth System Models

Before EOSC-Nordic

There is no “general” pattern when it comes to practices for developing ESMs. If researchers have sufficiently powerful PCs and the ability (admin access) to install the necessary software and libraries, they start developing on their PC but quickly move to HPC where all the necessary input data and software stack are available and extensive testing at scale can be performed. Even though most HPC systems have debug/develop batch queues, many researchers would like to have more “intermediate” computing resources for running their developments and tests.

Role of EOSC-Nordic

As mentioned in the description of this activity, researchers need a very flexible framework for developing their models, with as rapid turn-around as possible but still with a software environment similar to what they would find when running ESM in “operational mode” (see Activity-3).

The work done within EOSC-Nordic has already been reported in Deliverable 5.2 (Cross-borders computing through portals). We only give a short summary in the context of data management: Interactive JupyterLab has been deployed as an interactive tool in Galaxy (same as detailed above in Activity-1). This service is currently used by climate modellers and ecologists working together on improving Earth System Models to better represent vegetation at high-latitudes (EMERALD project, <https://www.mn.uio.no/geo/english/research/projects/emerald/index.html>).

Input data required for this activity is manageable (i.e. not large) and researchers usually upload and share “standard” model inputs from the shared Galaxy data library (“reference datasets”). Additional and customized datasets, in particular those based on observations, prepared by field ecologists are uploaded in the user’s Galaxy history and only shared with a small subset of colleagues (those directly working on the same research project). These datasets tend to be private and are not publicly discoverable (the metadata tends to be more informally stored, for example information exchanged in emails, or some documentation on a web page). When data is produced within the Galaxy climate JupyterLab, researchers need to make sure they “transfer/copy” them back to their Galaxy history (via the API for the Python library BioBlend): if not copied, data is lost when the container is stopped. Once copied back to their Galaxy history, users can choose to share them with their colleagues.

Sharing histories between different Galaxy instances is currently not automatic and is cumbersome when done from the Galaxy Graphical Interface: users need to download the Galaxy history as a tarball and then upload it to another Galaxy instance. To re-run anything from the history, users may need to ask the corresponding Galaxy administrators to install the same Galaxy tool (with the same version).

This Galaxy Climate Science Workbench²⁰ service has been in use for more than one year, and researchers generally appreciate the flexibility to develop (i.e. modify source code), compile and test their changes without having to wait for large computing resources to be available. Another advantage is that permissions on datasets can be easily customized and updated. However, researchers complain about the difficulty to exchange data between the container and Galaxy histories even though they can use the Python BioBlend API: even for short simulations/tests, model outputs can be quite large. One way to improve this service would be to add a simple JupyterLab extension to facilitate data exchange with a Galaxy FTP server, EUDAT B2* services and/or S3-compatible object storage. This could be further investigated in the future.

Activity-3: Role of EOSC-Nordic for running Earth System Models to produce climate data

²⁰ <https://galaxyproject.org/use/climate-science-workbench/>

Before EOSC-Nordic

Researchers request (and get) large amounts of compute time and storage from their respective national providers. There are usually no issues for running large Earth System Model simulations, but none of their simulations are FAIR. In particular, reproducibility is not an objective (even within large projects such as CMIPs).

Researchers in Norway complained about their National providers because:

- insufficient HPC resources were provided for running their simulations (most resources were dedicated to CMIPs);
- data had to be copied from/to the post-processing infrastructure.

The same is true in Sweden. In addition, the climate modeling community is encouraged to organize resource use collectively, but in Sweden there is no integrated, national data management and archiving service for research data.

We were contacted at the beginning of the EOSC-Nordic project by the EMERALD project (<https://www.uib.no/en/rg/EECRG/125308/emerald>) because they had difficulties to accommodate the needs from field ecologists whose technical expertise is very different from Earth System Modelers using an HPC command line environment: they asked for a Graphical User Interface for running their Earth System Models, in particular the single processor Functionally Assembled Terrestrial Ecosystem Simulator (FATES).

Role of EOSC-Nordic

The goal of EOSC-Nordic was to mitigate some of the issues raised by researchers, in particular:

- develop more robust and easy to use data management procedures for all researchers;
- make it easier to manage CMIP or any other large ensemble simulations;
- offer a “simplified” user interface including a Graphical User Interface to accommodate a wider range of users and increase the user-base of ESMs.
- Explore the possibility to share/use different computing resources such as cloud computing, HPC-cloud in order to offer a more “elastic” computing infrastructure to researchers (have more resources when CMIP experiments are running and “resume” to normal otherwise).

To deploy ESMs as a service, we had to:

- Package Earth System Models with a Package manager (conda): NorESM (Norwegian Earth System Model) is available on bioconda at <https://anaconda.org/bioconda/noresm> and CESM (Community Earth System Model) at <https://anaconda.org/bioconda/cesm>; when a new version is available, the maintainers need to update the conda recipe (pull request is done automatically but it needs to be merged by a bioconda maintainer);
- Containerize ESMs and develop an XML wrapper where additional metadata are added to improve the FAIRness of our tools. The containerization is done automatically thanks to the Galaxy tool framework (<https://github.com/NordicESMhub/galaxy-tools> for Climate tools and <https://github.com/NordicESMhub/galaxy-tools/tree/master/tools/cesm> for CESM).

This work is very similar to what has been done previously with the Functionally Assembled Terrestrial Ecosystem Simulator (FATES)²¹. The main difference is that FATES is a single processor tool while CESM can run on HPC with thousands of processors for some configurations.

Below (Fig. 3) is the Galaxy Graphical User Interface that comes automatically when writing the XML wrapper: Boxes marked with a “strikethrough” eye icon are meant to be expanded by users for customizing their runs. For instance, the duration (number of simulated years) can be customized as shown on Fig. 4.

²¹ <https://training.galaxyproject.org/training-material/topics/climate/tutorials/fates/tutorial.html>

CESM Community Earth System Model (Galaxy Version 2.1.3+galaxy0)
 Favorite
Options

Provide inputdata as a tarball

Tarball

inputdata for running CESM (tarball)

1: inputdata_cesm_2.1.3_B1850_f19_g17.tar

Name of your case

b1850_f19_g17

Model compset

B1850

Model resolution

f19_g17

Save workdir as a tarball when successful

☒ Yes

[Customize the model run period](#)

[Customize the model restart period](#)

[Land namelist user customization](#)

[River-runoff namelist user customization](#)

[Atmosphere namelist user customization](#)

[Land-ice namelist user customization](#)

[Sea-ice namelist user customization](#)

[Ocean namelist user customization](#)

user-modified source code

No

Email notification

☐ No

Send an email notification when the job completes.


The Community Earth System Model (CESM)

This tool creates and runs CESM experiments from CESM supported release. CESM is a fully-coupled, community, global climate model that provides state-of-the-art computer simulations of the Earth's past, present, and future climate states.

Available component configurations and grids:

- Grid Resolutions
- Component Sets
- Component Configuration Settings

Citations: 📖

- Danabasoglu, G., Lamarque, J.-F., Bacmeister, J., Bailey, D. A., DuVivier, A. K., Edwards, J., ... Strand, W. G. (2020). The Community Earth System Model Version 2 (CESM2). *Journal of Advances in Modeling Earth Systems*, 12(2), e2019MS001916.
<https://doi.org/https://doi.org/10.1029/2019MS001916> 

Requirements: ?

- cesm (Version 2.1.3)
- tar (Version 1.32)
- binutils (Version 2.35)
- python (Version 3)

Fig. 3: CESM Galaxy Tool wrapper.

Customize the model run period

Determines the model run initialization type.

startup

Run start date (yyyy-mm-dd). Only used for startup or hybrid runs.

Provides a numerical count for STOP_OPTION.

1

Sets the run length along with STOP_N and STOP_DATE

nmonths

Sets driver snapshot history file frequency

unset

Fig. 4: Galaxy GUI for CESM with expanded options for customizing the model run period.

Then, users can compose Galaxy workflows using the Galaxy workflow editor or by uploading an existing and shared Galaxy workflow such as shown in Fig. 5.

The screenshot shows the Galaxy Europe interface with a workflow titled "Workflow constructed from history 'CESM B1850 f19_g17'". The workflow consists of several steps: "inputdata_cesm_213_B1850_f19_g17", "CSM", "Extract Dataset", "NetCDF", and "NetCDF sammy map plotting". The right sidebar is open, showing options for selecting the tabular file to summarize, choosing variables to plot (TREFHT), and selecting coordinates (lat, lon). The sidebar also includes options for "Datetime selection" (No), "longitudes values 'lonW,lonE' for limited geographical area", "latitudes values 'latS,latN' for limited geographical area", "Shift longitudes [0,360] --> [-180,180]", "Range of values for plotting", and "Do not plot values below this threshold".

Fig. 5: Workflow for running a CESM B1850 f19_g17 experiment on Galaxy. This example workflow sets-up the case, compiles the code, runs one month and generates plots (surface temperature).

As indicated earlier, it is important to provide a GUI for users that may not be very familiar with command line, for instance:

- economists running ESM for evaluating the impact of climate change on industrial activity and society;
- ecologists running in production mode;
- researchers simulating large volcanic eruptions in an ESM to advance the knowledge and understanding of major volcanic eruptions' effects on climate, environment, and society in Scandinavia and Europe at the dawn of the Middle Ages ([VIKINGS](#) project).

This service has been successfully tested: for instance, we ran a fully coupled (all components of the ESMs were "active") simulation with CESM and the shared history (Research Object) is available at <https://climate.usegalaxy.eu/u/annefou/h/cesm-b1850-f19g17>. Another experiment with one component only (atmosphere) has been run and will be used for developing a teaching material for Master students: the shared Galaxy history is available at <https://climate.usegalaxy.eu/u/annefou/h/cesm-f2000>.

The plan for the remaining period is to develop more complex workflows and share them in WorkflowHub (<https://workflowhub.eu/>) and share Galaxy histories for instance in RoHUB (<https://www.rohub.org/>) to make ESM simulations more FAIR. Finally, in order to deliver this service to researchers, HPC-cloud computing resources are necessary to run longer experiments. Therefore the workflows and tests done within EOSC-Nordic should be considered only a proof of concept.

In addition, to be able to run the developed tools as Galaxy tools and workflows, one can directly run the generated containers, for instance as Singularity containers on an HPC system. This approach is currently being tested as part of the NICEST2 project (<https://neic.no/nicest2/>), WP4 "ESM workflows to efficiently run NorESM and EC-Earth on EuroHPC²²"

(<https://nordicesmhub.github.io/nicest2/2020/05/04/plan.html#wp4-esm-workflows-to-efficiently-run-nor-esm-and-ec-earth-on-eurohpc>).

The main advantage of such a framework (Galaxy tools and Galaxy workflows) is that:

- we can accommodate a wide range of users from novices (through the Galaxy GUI) to experts (via the command line using the BioBlend Python package to interact with Galaxy tools);
- it has a large user-base (in Europe and worldwide);
- it has a governance (<https://galaxyproject.org/community/governance/>) and a large number of core developers to ensure a successful research software infrastructure project;
- it has an active community offering support with Gitter channels (<https://gitter.im/galaxyproject>, <https://gitter.im/usegalaxy-eu>, <https://gitter.im/Galaxy-Training-Network>, etc.), a help forum (<https://help.galaxyproject.org/>), the Galaxy Training Network (GTN, <https://training.galaxyproject.org/>) developing training material (following best practices), organizing events (<https://training.galaxyproject.org/>) and Training Infrastructure as a Service (TlaaS);
- The Intergalactic Utilities Commission (IUC) is a community-driven group charged with establishing and maintaining best practices and gold-standard tool wrappers for the Galaxy ToolShed. The ToolShed now contains over 7500 tool definitions, created by over 600 unique contributors.
- Metadata can be easily added in the tool to improve its FAIRness when running the tool, e.g. when producing model outputs;

²² <https://eurohpc-ju.europa.eu/>

However this approach requires:

- that sufficient resources are available in Galaxy. Adding Pulsar nodes is not technically difficult, but adding HPC-cloud resources requires researchers to have access to such resources and not only HPC;
- that data needs to be imported to Galaxy (input data) because ESMs cannot currently use S3-compatible object storage;
- adding tools such as ESMs to Galaxy. But this is still seen as an obstacle by the climate community (too few people having the required expertise). Once added, it makes the ESM experiment “fully” reproducible and it will be easier to produce FAIR climate data;
- cataloging all the experiments done, for instance by “exporting” the corresponding Galaxy histories as Research Objects in ROHub (this may be done in collaboration with the [RELIANCE](#) project).

2.1.5 Pending Issues and Future Work

In this section, we summarize the pending issues/tasks that we have identified regarding integrated data management workflows for climate research and the risks to continuation of this work when the EOSC-Nordic project finishes.

Roadmap and solutions to be tested within EOSC-Nordic

We have already tested a number of solutions and researchers have given valuable feedback on the current services. The following tasks have been identified as priorities to be worked on in the near future:

- extend the EDAM ontology (<http://edamontology.org/page>) to increase the FAIRness of Galaxy Climate tools. The EDAM ontology is mostly used to add metadata to Galaxy tools and inform about Data, Format and Operation. Both Data and Operation need to be extended, e.g. adding terms related to climate modelling and analysis. For instance, Operation has “Modeling and simulation” but any further description is not relevant for the Climate community (it does not contain “climate modelling”).
- update Galaxy Climate tools to allow “direct” access through S3-compatible object storage;
- comply with existing best practices for increasing the FAIRness of workflows (add a license, authors, etc.);
- systematically register workflows in WorkflowHub;
- add one or two ESM workflows as part of Galaxy’s IWC (Intergalactic Workflow Commission) for testing on a regular basis the validity of these workflows;
- develop tools to automatically add metadata to research experiments when performing a simulation and ease their publication (S3-compatible object storage or long-term archive);
- add a tool to get a persistent identifier to every ESM outputs (using ROHub);
- add a tool to automatically generate a list of “handles”/persistent identifiers when using data from ESM experiments;
- “publish” data provenance (metadata) to facilitate collection of feedback information, thereby providing more visibility and traceability.

Beyond EOSC-Nordic

We have confidence that the accessibility and availability of the services deployed and tested within EOSC-Nordic are assured however, after the end of the project:

- there is no commitment regarding updates and evolution to, e.g., newer version of ESMs, new packages, new compilers, new architectures, etc.;

- the current user base is still fragile and consolidation requires continuous support/training which will not be offered anymore;
- there will be no more on-boarding of new users (PhDs, postdocs, etc.) and the user base is likely going to decline;
- needs that emerge from the usage of these services in the context of EOSC (registries with workflows, sharing data and research objects, etc.) cannot be addressed anymore within EOSC-Nordic once the project ended;
- users will see no “return on investment” whereas they made significant effort to support EOSC-Nordic while services were being developed and then deployed;
- information about share and reuse of data/tools/workflows is still not collected, hence creators will not get any feedback which could be used to improve their research;
- without follow-up, and in particular measurable impact (as for papers), and due to a limited number of adopters, there will be no incentive for users to “do things the right way” and no possibility to differentiate between “bad” and “good” Open Science practitioners (who will gain no visibility or additional funding).

This list represents a risk to the progress achieved in the EOSC-Nordic climate use case and needs to be managed in concert with our collaboration projects: NICEST2, EOSC-Life, RELIANCE, etc.

2.2 Use Case 2: Nordic Language Processing Laboratory (NLPL)

The Nordic Language Processing Laboratory (NLPL) is a collaboration of research groups in (currently) Denmark, Finland, Sweden, and Norway with a specialization in Natural Language Processing (NLP, which is often used synonymously with Human Language Technologies). NLPL researchers develop methods and tools for various aspects of enabling computers to “make sense” of human language, for example automated machine translation from one language into another, or what is called sentiment analysis (or opinion mining), i.e. detecting subjective, evaluative statements in, e.g., product reviews or social media data.

Despite different specializations, the NLPL researcher community builds on a common theoretical inventory as well as shared techniques and frameworks, in recent years predominantly through applications of deep learning. Therefore, NLP research to date is experimental in nature, where a typical experiment cycle consists of repeated sequences of designing or refining model architecture, supervised training on available data, and evaluation on held-out test data. Model complexity and training data volume determine computational needs, where typical experiments can range from a few GPU hours to several GPU years. Software support for distribution and parallelization across multiple GPUs (on multiple nodes), thus, is an essential technological requirement, as is maximizing usage efficiency of available resources. NLPL researchers are expert users (software developers, to some degree) who work from the command line and navigate batch computation in an HPC environment comfortably.

NLP research groups in different countries often apply similar techniques to different languages, with e.g. Finnish and Norwegian as central research targets in Finland and Norway, respectively. Nevertheless, much contemporary NLP research is cross- or multi-lingual in nature, i.e. involving more than one human language in a typical study. For these reasons, widely used collections of training and evaluation data comprise a range of different languages and are used by researchers from different NLPL member sites and countries. Furthermore, pre-training and transfer learning are basic building blocks in NLP research today, where what are called very large language models (VLLMs) play a central role. The BERT (Bidirectional Encoder

Representations from Transformers) family of VLLMs, originally defined and pre-trained by Google, arguably is the best known example here. These models often provide the starting point for architecture extensions “on top” of BERT-like models, where training comprises fine-tuning the parameters of the original BERT layers, and estimating parameters for the additional architecture components. Augmenting the original work from Google, NLPL researchers have trained BERT-like models for many of the Nordic languages, and jointly with the original models pre-trained by Google, these form data prerequisites to much of the current NLP experimentation.

2.2.1 The NLPL Virtual Laboratory

As part of the NLPL use case in EOSC-Nordic, the researcher community is engaged in creating a shared *virtual laboratory*, i.e. a uniform software and data environment on multiple Nordic HPC systems that

- lowers the bar to entry for NLP researchers;
- reduces duplicative effort, e.g. software and data provisioning and maintenance;
- facilitates community “self help”, i.e. contributes to knowledge sharing and raising technological competencies;
- increases replicability and reproducibility of results, both across different HPC environments and over time.

In mid 2021, the NLPL virtual laboratory was instantiated on the national Puhti and Saga superclusters in Finland and Norway, respectively, with additional instantiations underway on the national Betzy system in Norway and a smaller cluster at the University of Oslo. The NLPL consortium hopes to extend the same basic approach to instantiating its virtual laboratory on the EuroHPC LUMI²³ system.

The NLPL approach to fully automated software provisioning across different systems, including local compilation with system-specific optimizations, has been described in EOSC-Nordic Deliverable D5.3. While software ultimately needs to be custom-configured and compiled for each target CPU and GPU architecture (and, in principle at least, each type of interconnect), data provisioning in the NLPL virtual laboratory is primarily a management and replication challenge.

The NLPL researcher communities in Finland and Norway have been selected for the forthcoming LUMI GPU pilot, where the plan is to use the “burn-in” phase of the new system to pre-train a large number of very large language models. This initiative will require data preparation in the current instances of the NLPL virtual laboratory (in Finland and Norway), combined with the instantiation part of the laboratory on the LUMI system, so that large-scale training can be executed there.

2.2.2 Data Replication across HPC Systems

Pre-installed, shared data resources in the NLPL virtual laboratory are predominantly natural language data and pre-trained models. The former of these are (to date) exclusively comprised of textual data (as opposed to spoken language) and can be further subdivided as to whether the “raw” language data is paired with what are called annotations, i.e. representations of associated linguistic or extra-linguistic information, for example word classes or grammatical structure at the level of individual sentences, or topics or degrees of positive or negative sentiment at the level of paragraphs or documents. A collection of textual data is commonly called a corpus in NLP, where the NLPL data repository includes corpora ranging from thousands of tokens, typically paired with detailed linguistic annotations, to hundreds of billions of tokens of “raw”,

²³ <https://www.lumi-supercomputer.eu/>

unannotated text. Textual data further lends itself well to on-disk compression, and storage sizes, correspondingly, are measured in megabytes in small numbers of terabytes.

More than the overall storage footprint, a challenge presented by the NLPL data resources is their internal organization into collections with relatively large numbers of smallish files. Optimising data management and file structures is not easy as there is a large variety in data files and text corpora and how they are organised into documents and subsets. Both types of resources exist: large collections of smallish documents (on the order of millions) organised into complex file structures as well as big text files that contain large bodies of text with some internal structure. The latter is typically easier for HPC clusters and shared file systems to cope with. Following are some key statistics for the data sub-directories of the NLPL community directory, as of mid-October 2021:

Type of Data	Number of Files	Disk Space Used
Text Corpora	21,059	1.1 Tbyte
Extrinsic Evaluation	6,057,885	1.2 Tbyte
Machine Translation	2,303	141 Tbyte
Parallel Corpora	957,527	11 Tbyte
Syntactic and Semantic Parsing	649,968	150 Gbyte
Large Language Models	457	717 Gbyte

The challenges above make it difficult to define general replication and data staging routines that would be applicable to any NLPL resource we would like to integrate. Staging data from external storage and unpacking large file collections becomes time-consuming and causes unnecessary data replication for individual users of NLPL data sets. Therefore, our strategy is to keep the most important resources on live disks shared among NLPL users to make them immediately accessible from the native file system. We created our NLPL-specific disk partition with a logical file structure documented in a resource catalogue in our NLPL webspace. Resources have a master copy on one of the HPC centers and regular cron²⁴ jobs are used to replicate them on other NLPL instances. At present, these jobs run nightly on the various superclusters where the NLPL virtual laboratory is instantiated, to

- retrieve a common set of configuration files and scripts from a central source repository;
- run rsync²⁵ on selected subdirectories with corresponding custom lists of target systems;
- email log excerpts and status updates to an NLPL-internal team of infrastructure coordinators.

This custom infrastructure runs entirely in user space (in the account of the NLPL technical coordinator) on the cluster login nodes, with the minor inconvenience that cron jobs need to be re-enabled every time the corresponding login node is re-installed.

Storage capacities are different in the connected HPC facilities for NLPL and, therefore, the replication is only partial, covering selected resources that we can mirror. In principle, large parts of the NLPL data space do not require especially high-performance access: These files are often read sequentially once upon job

²⁴ <https://linux.die.net/man/8/cron>

²⁵ <https://linux.die.net/man/1/rsync>

²⁶

startup, for example to initialize the parameters of a neural network. We have started to discuss informally with Uninett Sigma2 and CSC candidate solution that could make these parts of the NLPL data repository available for read-only access across HPC systems, for example using a lightweight filesystem like CernVM-FS²⁶, possibly combined with Stratum 1 replica servers²⁷ as required.

Resources that cannot be replicated are available from external storage based on ObjectStorage solutions. Staging from those resources is restricted to users with appropriate access permissions and, therefore, we primarily use open HTTPS connections to provide data in case they have permissive licenses. Cross-border solutions for a seamless integration of national storage services have not yet been identified and we currently investigate solutions based on Reva (<https://reva.link/>) and data services developed by CS3Mesh4EOSC (<https://cs3mesh4eosc.eu/>). More about the NLPL use of data staging and external storage solutions is described below.

2.2.3 Management of Access Rights

The challenges that have yet to receive adequate technical solutions relate to access rights and filesystem-level permission management. First, with multiple users maintaining a shared collection of directories and files (i.e. the community directory, the shared software and data installations that make up the core of the virtual laboratory), we at times run into permission challenges that can be tedious to resolve or require administrator privileges (which the NLPL volunteer maintainers do not have on the target HPC systems). This challenge can be exacerbated by fluctuation in the NLPL volunteer community, where for example some data sub-directory can be maintained by a postdoctoral fellow at one of the partner sites for some years, but eventually ownership and maintainer responsibility need to be transferred to another volunteer. One could imagine an interface where something like a cron job regularly reads a specification of permissions and such from a file (maintained by the NLPL infrastructure team), quality-controls the contents, and then executes corresponding actions, e.g. using a file that would contain for each directory a shell command like:

```
software/modules/etc/nlpl/tmp  rm -r
software/modules/              chgrp -R nlpl
software/modules/              chmod g+rwX,o+rX
```

A related but potentially more intricate challenge is inclusion of non-public data in the NLPL community directory. To prepare for its participation in the 2022 LUMI GPU pilot, for example, the Oslo and Turku teams have partnered with the national libraries of Norway and Finland, respectively. These institutions possess very large collections of digital natural language data, of which they are in principle prepared to make available select parts for select user groups and applications. Storage and processing of somewhat restricted (albeit not sensitive) data outside of the data centers of the national libraries requires certain assurances, for example that data copied onto a shared national HPC system be adequately limited in access to a well-defined group of users. Such restrictions can (with some administrator support) be implemented on a token HPC system through custom file groups and enforcement of adequate permissions at the filesystem layer. However, these mechanisms present challenges to the replication strategy sketched above, seeing as (a) replication runs in user space, such that all data to be replicated needs to be visible to the dedicated user operating the cron jobs and (b) account and file group management to date is limited to either one local HPC system, or one national ecosystem. Unified cross-border user management would seem like a prerequisite to making progress on these challenges.

²⁶ <https://cernvm.cern.ch/fs/>

²⁷ <https://cvmfs.readthedocs.io/en/stable/cpt-replica.html#>

27

2.2.4 Data Staging from “Off-line” Storage

NLPL uses external storage solutions and data staging for data management using services provided by the national HPC providers. ObjectStorage based data services are offered by CSC (under the name Allas, <https://docs.csc.fi/data/Allas/>) and NLPL is a heavy user of that service. Currently, the largest resource in NLPL is OPUS, a diverse collection of so-called parallel corpora consisting of textual documents that are aligned to translations in many languages. The data compilation and management routines are divided into various stages:

- data compilation and preprocessing is done on local scratch space on HPC clusters using tools and procedures that are developed and stored in Git repositories;
- development files are pushed to Allas into subset-specific containers;
- data releases are created as compressed data archives to the shared NLPL space on live disks on the HPC cluster;
- data releases are uploaded to Allas within the NLPL project space;
- data releases are replicated to the Finish Fairdata server using IDA²⁸ services at CSC (<https://www.fairdata.fi/>).

Staging from and to Allas is done using Swift and S3 protocols and command-line clients such as swift, s3cmd or the Allas-specific frontends developed by CSC (https://docs.csc.fi/data/Allas/using_allas/a_commands/). Special care has been taken to address issues with ObjectStorage limitations. In particular, we package data sets into subsets that do not exceed the file limit of containers (a maximum of one million files per container) and we use the segmentation approach of the Swift client (static large objects) that splits large files into smaller segments to address the size limit of objects in storage containers (5 GB size limit).

Data on ObjectStorage is not backed up and, therefore, we also push released data to Fairdata using IDA services. Currently, OPUS occupies 12 TB of storage space using compressed and packaged data files. This is likely to grow and does not completely fit on the allocated storage space on live disks anymore. Hence, staging will become more important in the future and, within the EOSC-Nordic project, we are seeking solutions to enable seamless integration of external storage locations and staging routines. There are some obstacles that make this process difficult:

1. Access permissions: data in ObjectStorage containers can only be accessed with appropriate access rights. However, access permission configuration on services like Allas is limited and providing access can have disastrous consequences. Entire projects can be wiped out with a single command and data is gone without the possibility to recover them.
2. Container limits may become problematic for some data sets where the restrictions cannot be met.
3. Cross-border use of national services like Allas are not possible at this stage as billing and access permission management is not coordinated.

For public data (like OPUS releases), the latter is not such a problematic issue as download can be assured with world-readable objects on services like Allas. For other essential resources that require more fine-grained access permissions, this becomes an issue.

In order to address the first obstacle, we are investigating the development of a transparent NLPL staging tool. The motivation is to avoid giving permissions to a wider range of users that might compromise data safety in our storage containers, we envision a command-line tool that supports staging as a backend service instead of directly accessing the containers. This would also allow a transparent integration of

²⁸ <https://research.csc.fi/-/fairdata-ida>

different storage services from various providers without changing the interface for the end user. The general idea would be to implement

- a command line tool for requesting a data resource from the NLPL catalogue to be staged on shared disk spaces in local NLPL instances;
- a service (daemon) that collects requests and checks availability of the resource and storage capacity on the local disk, frees up space if necessary by removing unused resources, and finally stages the data from external storage with appropriate access permissions on local disk.

The NLPL staging tool would serve as a kind of long-term cache in NLPL disk space to provide the most useful resources from the collection in our catalogue. Solutions based on Reva (<https://reva.link/>) and CS3Mesh4EOSC (<https://cs3mesh4eosc.eu/>) might be possible but also fresh new implementations tailored towards the NLPL use case could be possible. Dedicated cron jobs and NLPL catalogue services based on Kaivos (<https://docs.csc.fi/data/kaivos/overview/>) are solutions we consider for the latter. Our goal is to find and implement a solution within the final period of EOSC-Nordic.

2.2.5 Very Large-Scale Data Management

To prepare for the massive increase in GPU availability on the LUMI system, the NLPL partners will need to increase availability of very large scale natural language data. Jointly with the University of Edinburgh, Charles University in Prague, and three national HPC service providers, the NLPL partners have submitted a Horizon Europe²⁹ proposal to prepare a European data space for High Performance Language Technologies (HPLT). The HPLT initiative seeks to mirror seven petabytes of web data from the US-based Internet Archive (<https://archive.org>) on national storage facilities in the Czech Republic, Norway, and the UK, in order to extract textual data for a broad variety of languages, and further prepare the derived training data for staging to LUMI. Transfer of such data volume from the US to multiple European storage sites, as well as data management of this 'bulk' data collection goes beyond the NLPL experience to date. And while much of the data preparation can be distributed across multiple storage sites and associated local computing facilities, there are some preparatory steps that require synchronization across the full collection, for example deduplication (identification and removal of repeated copies of the same content). To pilot this initiative, the partners have sketched a smaller-scale experiment with CSC and Sigma2, where about half a terabyte of data shall be transferred from an existing, partial mirror in the UK to storage in Finland and Norway, for data preprocessing on the national instances of the NLPL virtual laboratory in these countries. This pilot is expected to launch in late 2021 and will have its focus on Finnish and Norwegian language data, to augment the pool of available training data for these languages for the LUMI GPU pilot in early 2022.

3. Lessons Learned

3.1 Climate

Several issues clearly came out during our investigations and deserve attention, these are mostly related to data access (regardless of the infrastructure), data permissions (in particular the case of private datasets has

²⁹

https://ec.europa.eu/info/research-and-innovation/funding/funding-opportunities/funding-programmes-and-open-calls/horizon-europe_en

²⁹

to be urgently addressed), data transfer (streaming could be one solution), unique/persistent identification of the data and tools, catalog (with registries of registries), the ability to trace/monitor the re-use of any data or tool and obtain some usage statistics, which would eventually contribute to recognizing the work done by their authors.

Onboarding should not be underestimated: it is essential to inform new users about the services available, to train users on how to best use and/or contribute to existing services and make these evolve, to ask users for feedback (which obviously means to implement or discuss potential improvements). Two major issues in EOSC are i) the lack of funding for proper maintenance and sustainability of the services deployed, and ii) the fact that the responsibility of services is delegated to each provider without any commitment about the content of the service and whether it still meets user expectations.

As to the FAIRification, this should not be seen as an afterthought but as an integrated process in which metadata is collected during the entire research work life cycle (data, tool and workflows ought to be linked together and aggregated as a research object) in order to provide comprehensive and really useful contextual information. Above all interoperability between services is paramount for the success of Open Science, and this requires Open Infrastructures and not only open data, open tools and open workflows.

3.2 NLPL

The NLPL use case is quite different from other areas addressed by EOSC services. Natural language processing is very much about system development as well as data-intensive experimentation. The field currently emphasizes new algorithms, methods, and machine learning architectures as the main publishable research output. With this, it becomes very difficult to serve HPC needs with ready-made solutions and packages that fulfill all requirements of the researcher. Things that are essential for the typical NLPL user are cutting-edge software libraries that are available for further modification and testing to work on the latest models and algorithms. The development can be fast-paced and rapidly changes, and NLP research is at times based on niche software implementations and buggy development code.

The NLPL use case emphasizes the need of a basic software development kit that is common for most tasks and the availability of standard data sets that reassure compatibility and replicability of on-going research. Most of the tools operate on command-line interfaces and research runs as systematic batch jobs with direct access to data on the file system. Data replication on HPC clusters, therefore, becomes an important prerequisite to provide the basic environment that is needed.

In EOSC-Nordic, we learned that encapsulation of batch jobs that travel with the data is not easy, especially in cases where data sets grow large and have complex structures. Solutions like Unicore (<https://www.unicore.eu/>) for distributed computing with attached data resources do not seem to be mature enough and are not preferred by HPC providers because of their limited developer base. Many ideas are further hampered by missing coordination and harmonization in cross-border access control and resource allocation management. NLPL, therefore, continues to focus on data management with national storage services with regular synchronisation of essential data sets using regular cron jobs. In the future, the shared resources on the new LUMI environment may provide an interesting solution where hardware solutions and services are natively shared across various countries, which would make collaborative HPC work possible without complex synchronisation efforts across platforms.

4. Summary and Future Work

With the procedures investigated in subtask T5.3.1, our aim was to provide helpful recommendations accompanied by practical examples on how cross border data management and integration within EOSC can be achieved. The selected use cases have explored different technical solutions to serve their respective community of users and interest in the deployed services by researchers has been encouraging. Further work will continue through to the end of the EOSC-Nordic project to further consolidate the progress achieved.

One issue faced by both use cases is sustaining the developments described here beyond the life of the EOSC-Nordic project. Sustainability of services has been flagged in other EOSC-Nordic deliverables (e.g. D5.2³⁰) and is receiving substantial attention in EOSC-Nordic WP2.

³⁰ <https://www.eosc-nordic.eu/kh-material/d5-2-cross-borders-computing-through-portals/>

³¹