H2020-EINFRA-2017

EINFRA-21-2017 - Platform-driven e-infrastructure innovation

DARE [777413] "Delivering Agile Research Excellence on European e-Infrastructures"

D7.4 Pilot Tools and Services, Execution and Evaluation Report II

Project Reference No	777413 — DARE — H2020-EINFRA-2017 / EINFRA-21-2017
Deliverable	D7.4 Pilot Tools and Services, Execution and Evaluation Report II
Work package	WP7: IS-ENES/Climate4Impact Use Case
Tasks involved	T7.3 IS-ENES Pilot Development; T7.4 Evaluation
Type	R: Document, report
Dissemination Level	PU = Public
Due Date	31/12/2020
Submission Date	30/12/2020
Status	Draft
Editor(s)	Christian Pagé (CERFACS)
Contributor(s)	Christian Pagé (CERFACS)
Reviewer(s)	Iraklis Klampanos (NCSRD)
Document description	A report detailing the set of DARE components activated for the use case, their configuration parameters and possible customization steps

	(if needed). It will also report on pilot execution benchmarks to assess the usability and performance of the pilot. At early stage the pilot prototype will be evaluated and results will feed in the next stages.

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
1	03/12/2020	First version	CERFACS
2	10/12/2020	Internal review version	CERFACS
3	23/12/2020	Final version	CERFACS

Executive Summary

This deliverable is an updated version of D7.3, with significant changes because the new climate Use Case that has been designed, implemented and evaluated. The objectives of this deliverable are to evaluate the Climate Science Domain pilot prototype. The evaluation is done by targeted users of the DARE platform, and the results feed into the dissemination and adoption toward the climate scientific community, through the IS-ENES channels. The targeted users are software developers of the Climate Research Infrastructure, and the evaluation session has been organized virtually as a webinar, hosted by CERFACS, on October 16th, 2020. 24 persons registered to the webinar. The results show a positive view about the DARE platform. Regarding a future adoption of the DARE Platform by the community, results and discussions show that, overall, the adoption should take place as long as the DARE platform technology is picked up by the H2020 IS-ENES3 project. It will be crucial that dissemination within IS-ENES3 continues in the upcoming year, and that the DARE platform gets further developed in future European projects.

Table of Contents

List of Terms and Abbreviations

Abbreviation	Definition
C4I	climate4impact
CMC	Canadian Meteorological Centre
ECCC	Environnement et Changement Climatique Canada
ENES	European Network for Earth System modelling
ENES CDI	European Network for Earth System modelling Climate Data Infrastructure
EOSC	European Open Science Cloud
ESGF	Earth System Grid Federation
EUDAT CDI	EUDAT Common Data Infrastructure
NetCDF	Network Common Data Form
UQAM	Université du Québec à Montréal

1. Introduction

1.1 Purpose and Scope

The deliverable objectives are to analyze and report about the evaluation of the DARE developments related to the Climate Domain Pilot.

This deliverable consists of a report detailing the architecture and the schema of the climate use cases. In this updated version of the report, the focus will be on the second use case (readers should refer to D7.3 for a detailed description and report of the first use case). There will be an emphasis on the set of DARE components used. It will also report on the training and the feedback of the targeted users that attended the second webinar training. The evaluation and results will be used to steer and strengthen the future adoption of the DARE Platform by the climate community. It will be very important and crucial that dissemination continues strongly after the end of the DARE project, especially in the upcoming year.

1.2 Approach and relation to other Work Packages and Deliverables

Evaluation has also been done in WP6 for the Seismology Domain, and reported in D6.4. Both Work Packages have adopted the same methodology for the training and similar structure in the deliverables. The targeted users are not identical for those two domains - WP7 evaluation focuses more on developers - but nonetheless the approach is similar.

1.3 Methodology and Structure of the Deliverable

The structure of this deliverable is as follows. First, a summary of the second Climate Domain Use Case will be presented, then a summary of the technical implementation will be presented. Evaluation results will be shown and discussed, followed by lessons learned that will be used to steer the dissemination toward the climate research developers community.

First, it is important to state how the recommendations and comments regarding WP7 from the previous review were taken into account to steer the work for the second reporting period. Below are listed the three recommendations that were specifically addressing WP7 work, along with how it was taken into account.

R2. The consortium should develop a robust and realistic strategy to engage developer / user communities, including a more pro-active exploitation of the existing EPOS and IS-ENES communities in order to achieve a critical mass and ensure a sustainable business model. Approach to other potential developer / user communities should be assessed in terms of realistic effort-vs-outcome estimates. Only the most impactful engagements should be pursued and focused upon for the most efficient use of the remaining time and resources.

In order to accomplish the Reviewers' requirements, as it was done in WP6 for EPOS, a new additional use case has been developed in order to target a broader audience for the ENES and climate community (related to R4 below). The new cyclone use case was prioritised balancing the very limited efforts left available for the remaining time of the project. This should guarantee an effective exploitation of the

DARE platform also after the end of the project, as well as ensure platform sustainability since the engaged communities should be interested in deploying instances of the platform.
Training events were organised in order to increase the number of potential future users, with appropriate dissemination.

R4. The consortium should modify IS-ENES / Climate4Impact use case for more detailed user scenarios, with several sub-cases ranging from fundamental to operational/applied research, considering EPOS use case as an example of good practice and following a similar format.

A complete new use case was developed to fulfill those requirements. The use case is leveraging a complex analysis tool that is not available to end users and that is quite useful for the analysis of climate change scenarios: a cyclone tracking software that can track both extra-tropical and tropical cyclones. It was not possible to develop several sub-cases given the efforts left for the second half of the project in WP7, but the new use case is also acting as a template to develop further similar applications using the DARE platform within the ENES community. Interfaces and components that were developed can be used to support other complex analysis tools that exist in the climate community.

R5. Terms and means of collaboration with Copernicus should be clarified.

The use cases and the technology developed for the ENES community are targeting different categories of users than the Copernicus C3S. While the C3S is targeting end users that need to rely on climate services, in a more operational sense (the C3S is currently providing a subset of CMIP5 experiments and CORDEX-CMIP5), while the use cases developed within the DARE project for the ENES community is targeting researchers, because possible datasets are including all CMIP5 experiments (including CORDEX) as well as all CMIP6 experiments and any other experiments, as soon as they are made available on the ESGF data nodes. Researchers can also use those use cases to perform analysis on datasets that are accessible outside the ESGF infrastructure, as the only technical requirement is the OpenDAP protocol and basic CF compliance. Nevertheless, within the ENES community several institutes that are part of the project H2020-IS-ENES3 are taking part in the development of the C3S, ensuring a good information exchange and collaboration, as well as complementarity with the C3S and other Copernicus relevant initiatives.

2. Climate Cyclone Tracking Use Case

This is a new Use Case that was not presented in the DARE Description of Work. It has been designed following the DARE mid-term review report. The detailed description will be presented and discussed.

2.1 Description and Motivation

Objectives

- Enable end users to access complex climate data analysis tools.
- Enable access to large input climate data processing (multi-model multi-scenario) through remote execution on an instance of the DARE platform using cloud resources.
- Streamline and ease the whole data lifecycle.
- Definitely move away from a download-then-analyze type of workflow.

The Use Case also considers the following:

- Interoperability with the ESGF infrastructure.
- Interoperability with the EUDAT CDI by using B2 services.
- Coordination with the IS-ENES Data Task Force.

Cyclone Tracking Use Case Scientific Objective

- Main Objective: Analyse how climate change will impact the tropical and extratropical cyclone track densities and intensities.

Cyclone Tracking Methodology

This new Use Case is implementing the extra-tropical and tropical cyclone tracking (it can also track high pressures as well as mid-altitude weather systems), based on a Fortran implementation of the Sinclair (2004)[1] algorithm and methodology.

Why Cyclone Tracking? It can be used, for example, for those applications:

- Operational Forecasting
- Trends in Storm Tracking
- Numerical Model Verification (NWP and Climate Models)
- Storm Impact Studies
- Storm Impact Forecasting

This tool has been a group effort over a long period of time, from contributions of the Université du Québec à Montréal (UQAM) Earth and Atmospheric Science division, Ouranos (Montréal), the Canadian Meteorological Centre (CMC, Environnement et Changement Climatique Canada, ECCC), and the High impact Lab Québec Region (ECCC), and finally CERFACS. Of course the original code was written by Sinclair himself:

- Initial Tracking code from Sinclair (Fortran-77)
- Imported at UQAM and adapted to ECCC "Standard" file format
- Adapted for Climate Simulations at Ouranos (Hot & Cold Start)
- Tropical Tracking from CMC "merged"
- Realtime diagnostics on UQAM's meteocentre.com server

[1] Sinclair, M. R., 2004: *Extratropical Transition of Southwest Pacific Tropical Cyclones. Part II: Midlatitude Circulation Characteristics, Mon. Wea. Rev., 132, p. 2149.*

- Operational Version implemented at CMC
- Adaptation to the NetCDF file format at CERFACS

The algorithm itself has the following characteristics, and is based on the algorithm developed by Sinclair (1997)[2]

- Extra-Tropical Cyclones Tracking
    - Mean Sea-Level Pressure Minima
    - 1000 hPa Gradient Wind Vorticity Maxima
    - Lasting at least 24h
- Tropical Cyclones Tracking
    - Same as Extra-Tropical Cyclones Tracking, with additional variables
    - 850 hPa Relative Vorticity
    - 250-850 hPa Thickness
    - 10 m Surface Wind
    - Lower atmosphere baroclinicity

Analysis plots can be generated from the output of this algorithm, and those are very useful for different types of applications. In the current Use Case workflow here, it has been chosen to initially produce the density plot for climate change scenarios (see Figure 1). It represents the storm track densities for a given number of climate scenarios, and are used to assess the impact of climate change on those tracks, as it is important for storm impacts and precipitation patterns.

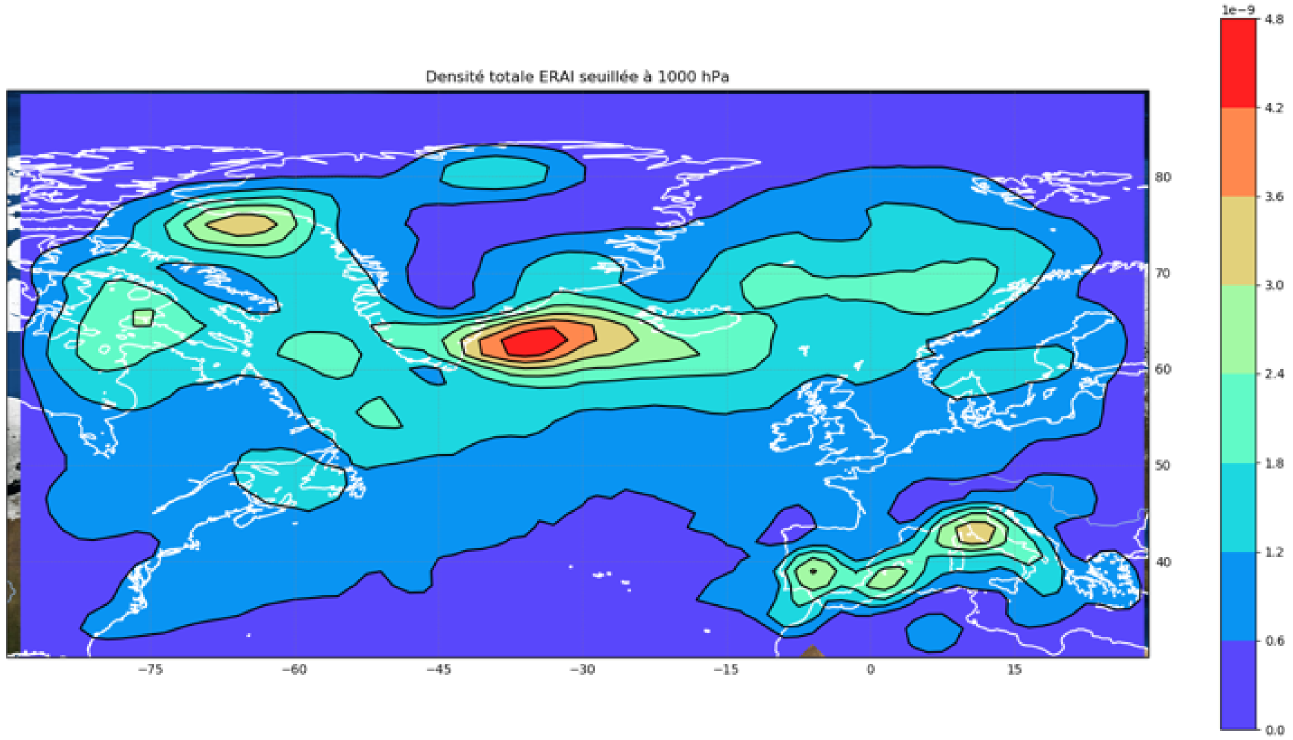


Figure 1: *Cyclone Use Case specific end-user plot example: storm track densities. This has been incorporated in the current implementation of the workflow.*

Expected Outcomes

[2] Sinclair, M. R., 1997: Objective identification of cyclones and their circulation intensity, and climatology. Wea. Forecasting, 12 , 595–612.

Demonstrate an end-to-end solution based on the DARE platform for the heterogeneous base of the climate-change impact community end-users, dealing properly with the large amount of data needed to perform their research and applications.

This Use Case is generic enough in the sense that all interfaces and components that will have to be developed for this use case will also be useful for most of the data analysis workflows currently needed in the climate science domain. It shows in a generic way how complex analysis tools can be leveraged for end users, in a relatively easy way by research developers.

The final sketch of the Cyclone Use Case is shown in Figure 2.

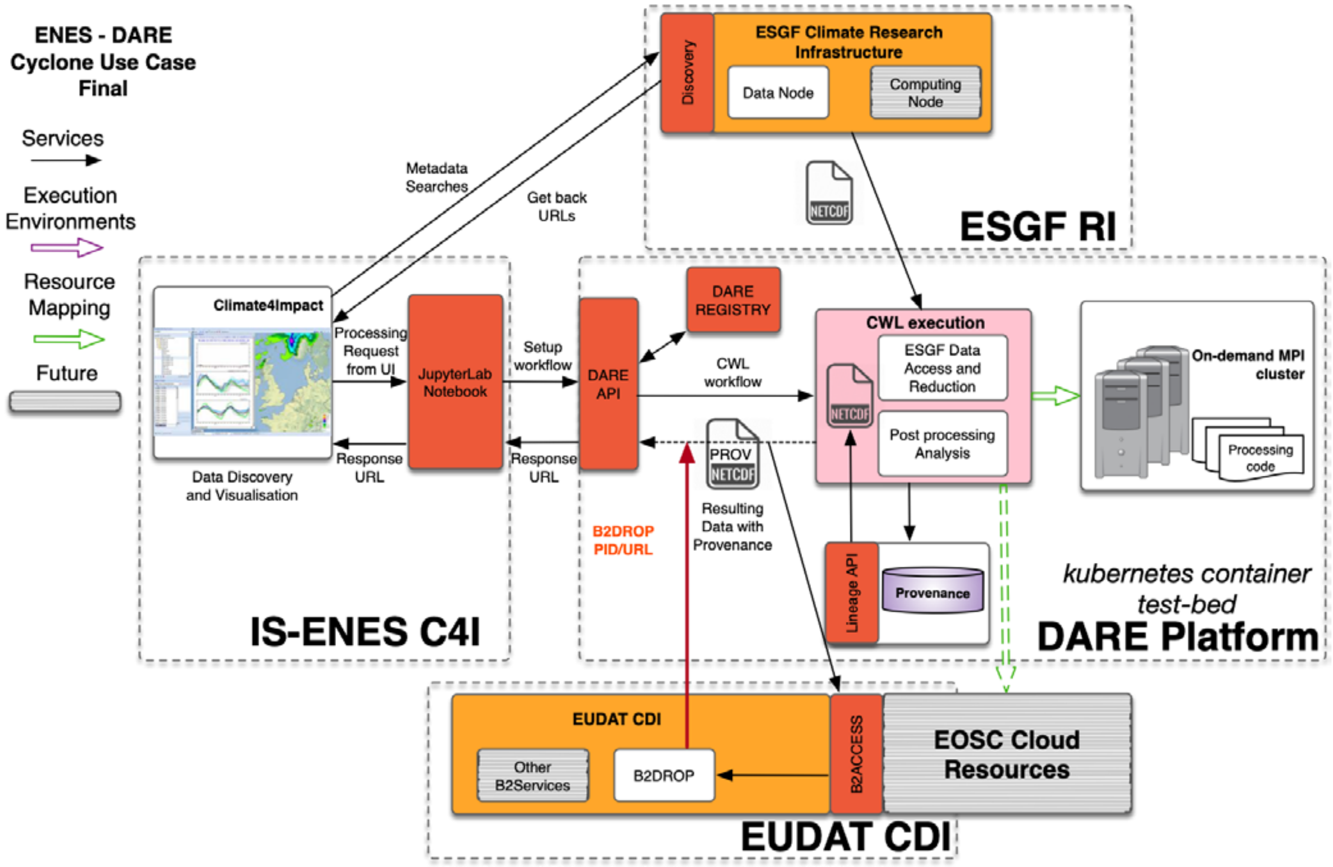


Figure 2: *Final version of the DARE Climate Cyclone Use Case Sketch.*

The idea of this Use Case is to delegate calculations related to data analysis triggered by end users on the IS-ENES Climate4Impact (C4I) portal. Currently, the calculations take place on the portal front-end server. This approach is not scalable and has a negative impact on the performance of the front-end.

In this use case, the DARE components take care of the processing as well as input and output, seamlessly and transparently to the end users, also adding provenance and lineage information. The C4I platform queries the Climate Research Infrastructure ESGF and retrieves URLs of data files using metadata queries according to facets. Those URLs are then forwarded with the processing request to the DARE Platform, using a Jupyter Notebook. The deployment of those DARE components must be easy. It is intended to hide the complexity of underlying e-infrastructures and technologies to the software developer of platforms like C4I or data processing workflows.

The major advantage of using the ESGF RI instead of the Copernicus CDS (C3S) is that the number of datasets available is much greater, as all CMIP experiments are available. It also gives access to newer datasets much sooner, for example to CMIP6 datasets at the moment, as they are not available yet on the C3S. This is because the targeted end-users are not the same as the ones targeted by the C3S. The C3S is a climate service, operational, targeting stakeholders and practitioners, in order to produce climate change impact studies. As stated on their website: "We provide authoritative information about the past, present and future climate, as well as tools to enable climate change mitigation and adaptation strategies by policy makers and businesses." While the ESGF RI and the targeted end-users of the Use Case here are those working in scientific research domains, using climate data and scenarios, or those that need specialized datasets or the latest climate scenarios. Those end users are very different, and they need different solutions. This also applies to the first Generic Climate Use Case that was presented in the DARE Description of Work and in D7.3.

The workflow and inner container is described in more details in Figure 3.

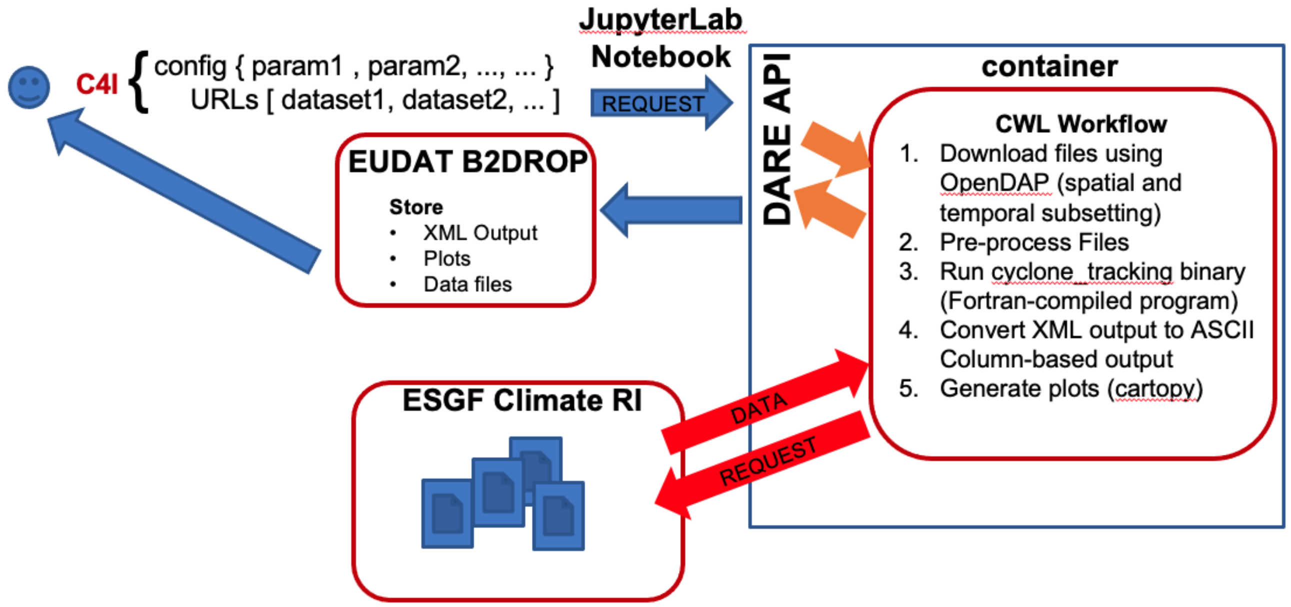

JupyterLab Notebook
C4I
config { param1 , param2, ..., ... }
URLs [dataset1, dataset2, ...]
REQUEST
DARE API
container
CWL Workflow
1. Download files using OpenDAP (spatial and temporal subsetting)
2. Pre-process Files
3. Run cyclone_tracking binary (Fortran-compiled program)
4. Convert XML output to ASCII Column-based output
5. Generate plots (cartopy)
EUDAT B2DROP
Store
• XML Output
• Plots
• Data files
ESGF Climate RI
DATA
REQUEST

Figure 3: *Final version of the DARE Climate Cyclone Use Case inner workflow container.*

This Use Case benefits from all the developments that took place on the first generic Use Case that was presented in D7.3 and described in D7.1 and in the Description of Work. During the development and implementation phase of this Use Case, an agile approach was used together with the developments taking place in the DARE platform to enhance and update the CWL workflows supported by all components.

2.2 Overview of External Components Integration

In D7.1, a list of possible components of external architecture that can be used within this use case context, is defined:

- ESGF Climate Research Infrastructure (RI)
- EUDAT CDI Services: B2NOTE, B2SAFE, B2SHARE, B2DROP, B2HANDLE, B2FIND, B2STAGE, B2ACCESS, GEF
- IS-ENES CDI C4I
- EGI FedCloud

For authorization and authentication systems:

- EUDAT B2ACCESS
- ESGF OpenID
- EGI Certificates
- EOSC OpenID Connect

In the schematic presented in the previous section, we can see that the implementation is using the following components:

- IS-ENES CDI C4I
- EUDAT CDI Service B2DROP
- EUDAT B2ACCESS tokens are used to access a given user's B2DROP account.
- ESGF RI Data Nodes (CMIP6 data do not require authentication)
- And of course DARE Components: API, Registry, CWL workflow support, lineage/provenance system

2.3 Summary of DARE API Integration

The DARE platform has been deployed by Fraunhofer: https://platform.dare.scai.fraunhofer.de including also JupyterLab on https://jupyter.dare.scai.fraunhofer.de . Those were used for the final tests and developments that were done before the Webinar training. It also has been used for the live demo presented at the training, and to provide interested participants an access to a DARE platform instance deployment.

The Cyclone Use Case implementation is using the DARE API to launch the execution of the workflow. This has been possible by the addition of CWL workflows support within the DARE API and underlying components. Deliverable 2.2 has more details on the architecture of the DARE platform and its implementation, so it will not be reproduced here. The focus here is rather to explain how the Cyclone Use Case implementation is using the DARE components and API.

The CWL workflow itself is prepared beforehand by the ENES climate community inside a docker container. The docker container is accessible in this repository: https://gitlab.com/project-dare/dare-platform/-/tree/master/containers/exec-context-cyclone/cwl
This container image has all the required environments installed to execute the steps of the CWL workflow of the Cyclone Tracking. The CWL workflow itself is retrieved when building the docker image, and is stored in the following repository: https://gitlab.com/project-dare/wp7_cyclone-tracking/-/tree/cwl
The original ENES community cyclone tracking workflow was implemented recently as a master bash script file launching a few python scripts as well as the Fortran binary of the cyclone tracking algorithm. Prior, all steps were performed by hand, manually. This master bash script has been separated into specific generic components written and converted as separate python scripts, and executed by a master bash script. As a final step, all those components as well as the master script were encapsulated in CWL components.

One of the benefits of the DARE platform is to run automatically in parallel many components of the workflow, using large datasets. In the current implementation, this has been accomplished using the Scatter Feature of CWL, which will run in parallel the cyclone tracking algorithm for several climate scenarios, and combine those results together to produce the required plots, without any modification to the workflow itself. Only the input information will differ, containing several datasets in a vector.

The ENES research developers, to leverage such complex tools, will only need to create or convert existing workflows running those analysis tools into CWL. The resulting workflows will also have the advantage of being shareable, and also benefit from all the advantages of the DARE platform, such as automated provenance (conforming to the PROV-O standard), as well as parallel execution on heterogeneous hardware and architectures.

The detailed process of running the Cyclone Tracking Use Case using the DARE Platform and its DARE API integration is shown in Annex A1.

2.4 Integration with the Climate Research Infrastructure

The integration with the Climate Research Infrastructure (RI) is now quite robust. Significant developments were done since the release of D7.3. At that time the integration with the Climate RI was not developed and designed yet. The current CWL implementation of the Cyclone Tracking Use Case is directly accessing climate model datasets on the ESGF Data Nodes, taking advantage of the time and spatial subsetting of the OpenDAP protocol using the very efficient and optimized xarray and dask python packages. The locations of the datasets are provided using the URLs generated by the C4I 2.0 novel user interface. This integration is really an important component of the current Use Case, but also future Use Cases within the climate community. It must also be emphasized that the source code was used for the developments that are being done to develop C4I v2.0 in the H2020 IS-ENES3 project. This is a great success of co-developments between two projects.

On the other hand, the ESGF Computing Nodes were not mature enough to be included in the current implementation, but this can be a way to drive adoption within the ENES Climate community using the IS-ENES3 H2020 project workforce, by planning the integration of ESGF Computing Nodes in future versions of the workflows using the DARE Platform. With the current Cyclone Tracking workflow and also the Generic Use Case from D7.3.

Regarding the evaluation, it is targeting the software developers of the climate RI and especially C4I, and not directly end users. Consequently the focus is on the integration and interfacing between the needed infrastructures to support the Use Case. Those infrastructures are the DARE platform itself, ESGF, EUDAT CDI and IS-ENES C4I.

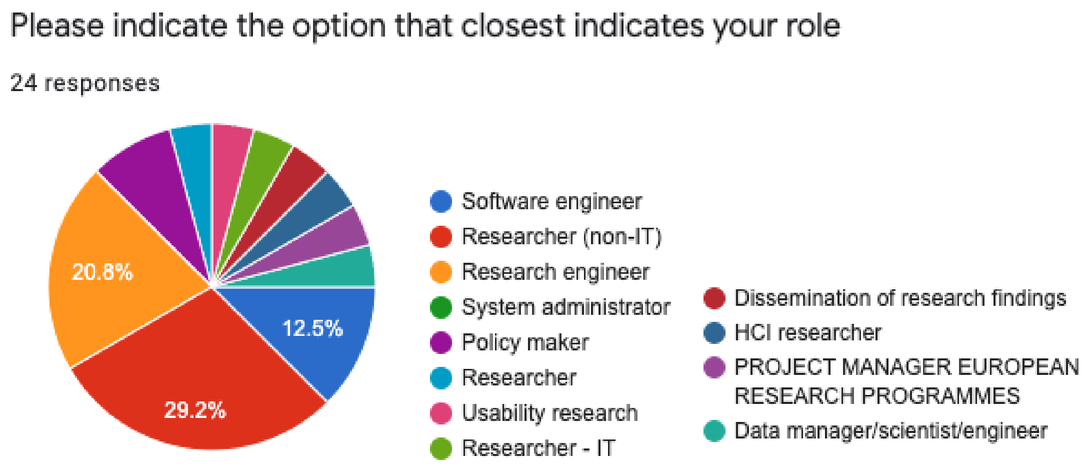
3. Training Event and Evaluation Results

The evaluation took place online and was organized by CERFACS, on October 16th, 2020. The reader is encouraged to read D8.5 for further details on the organization and agenda of the training. Here, the focus will be on the results.

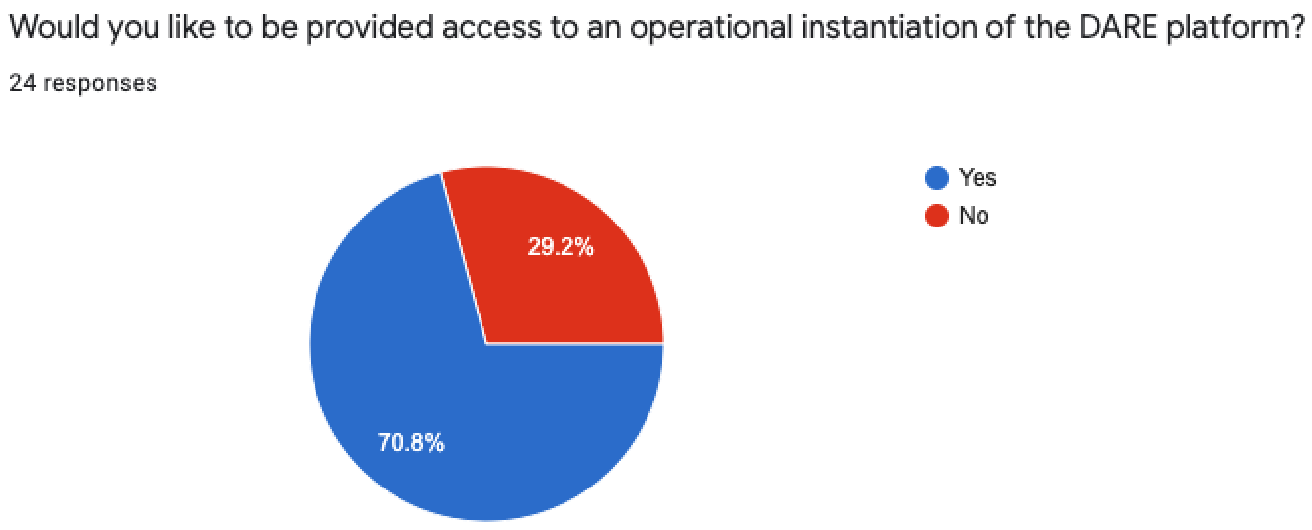
The event was advertised among the climate research developers community, mainly through the IS-ENES and ENES channels. Some people that are not directly working in the DARE project but from institutes involved in the DARE project, were also present. The total number of registered people was 24.

3.1 Training Attendees

The registration form is accessible here: https://docs.google.com/forms/d/1_dc3Ji1NBOYJ_dZV9BwLxLkF-UY8bmqFS0TwhwzSFgE/edit
24 people filled in the form.



The results of the attendees role poll is what was expected and in line with the objectives, with a large majority being involved as engineers and IT-related positions.

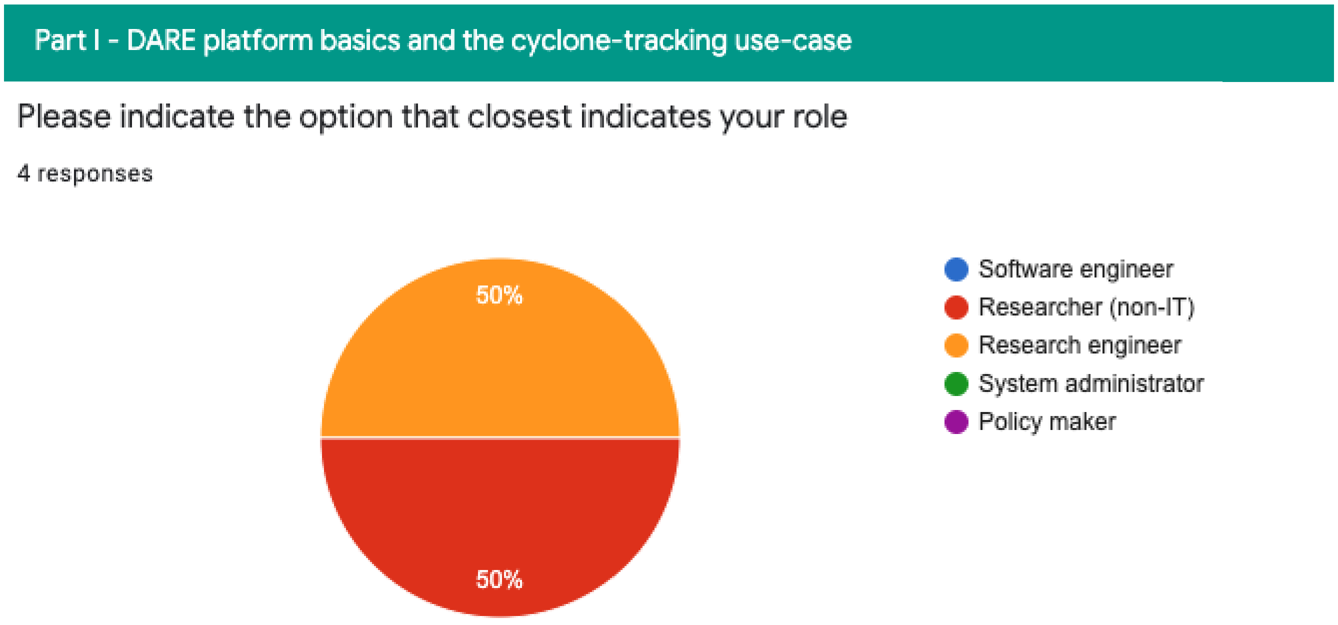


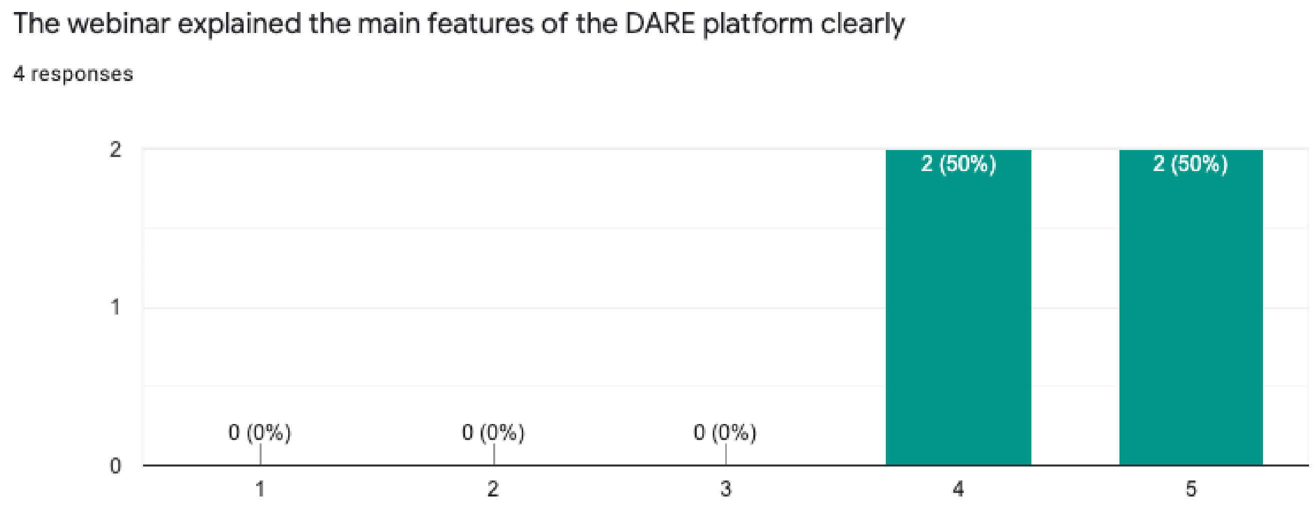
There was surprisingly a high percentage of people interested to try the DARE Platform. This means that dissemination prior to the webinar was effective in building an interest in the platform.

3.2 Evaluation Feedbacks

Participants to the training webinar were invited to participate in a survey to give their feedback about the training. This was done using a questionnaire at the following address: https://forms.gle/fvkKRf8M2MP7r4tA7

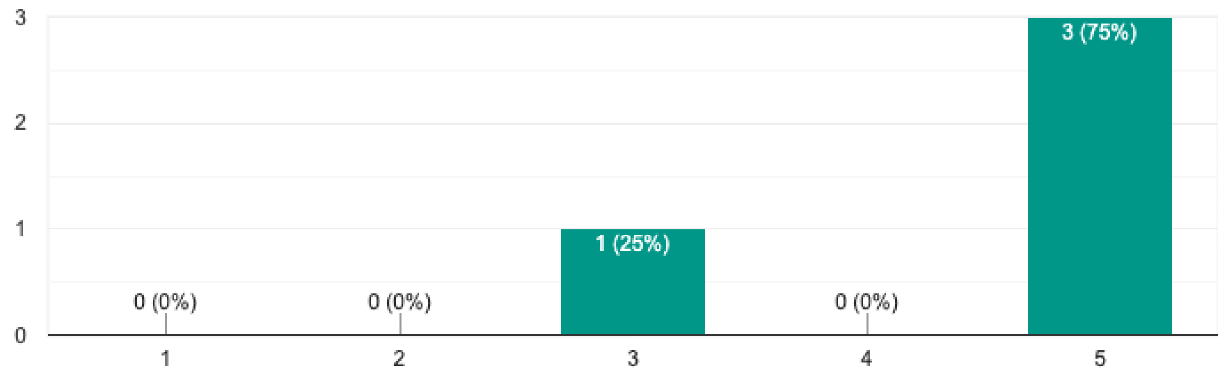
The participants were reminded a few times, but overall only a few took time to give their feedback. It is not possible to draw robust conclusions given the number of responses, but nonetheless those feedback can be informative. Also, combined with the questions and comments that the participants had during the training, and from the responses indicated in the feedback form (see below for details on the answers), in general most of the answers were neutral or positive about the training and the DARE Platform.





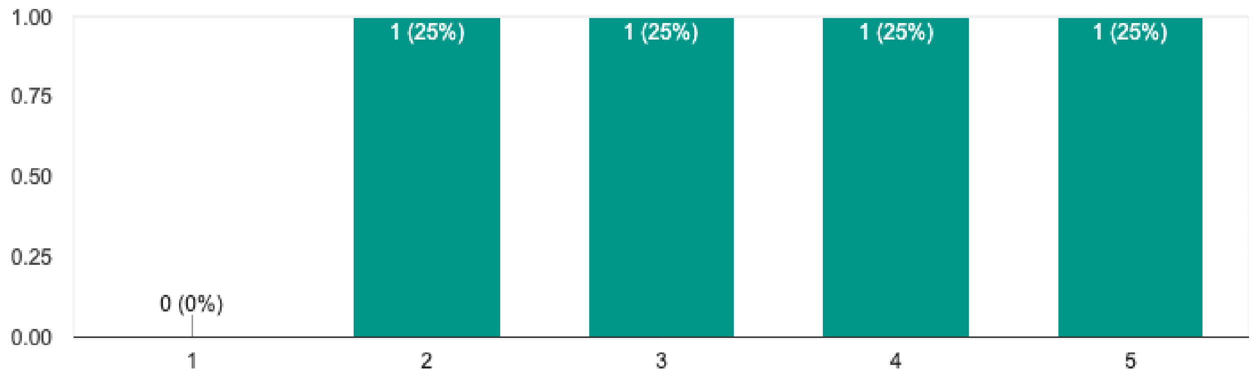
The webinar provided enough pointers to additional information and documentation to get me started as a user

4 responses



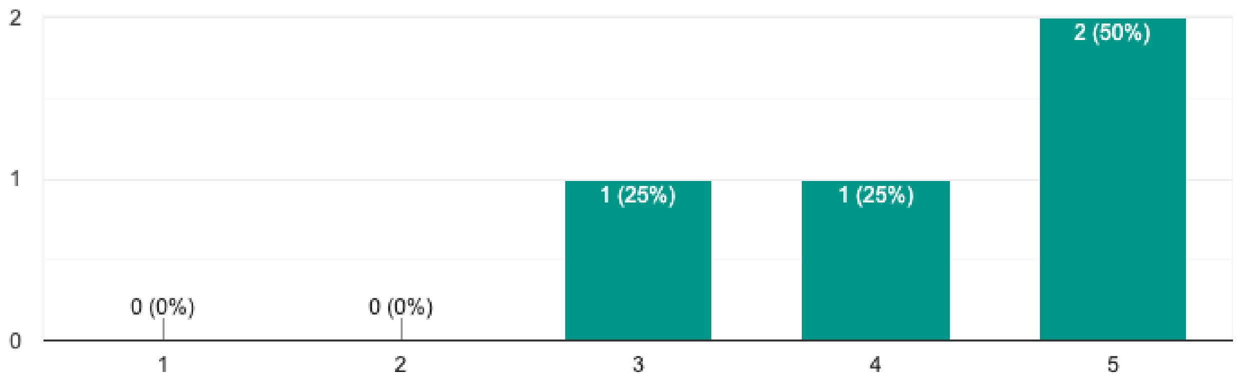
I could see myself using this platform for my daily work

4 responses



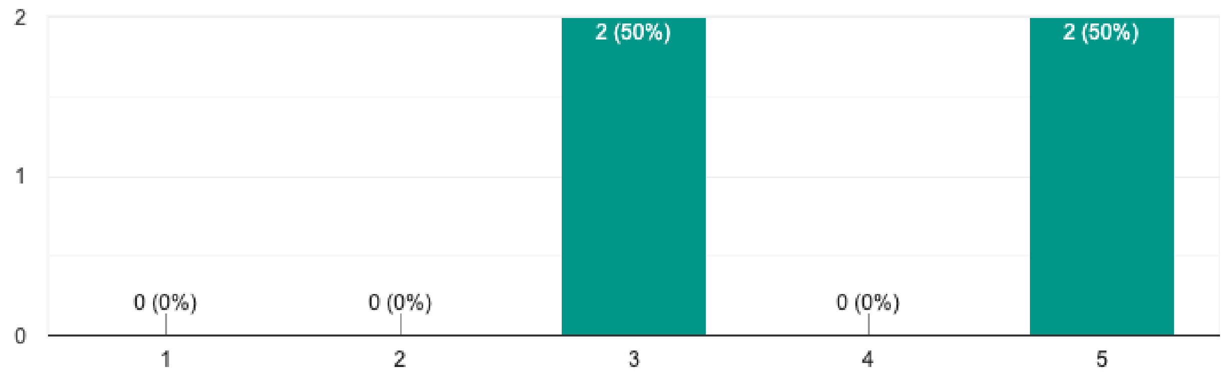
I think the various components of the platform (WaaS, provenance, registries, jupyter environment, etc.) are well integrated

4 responses



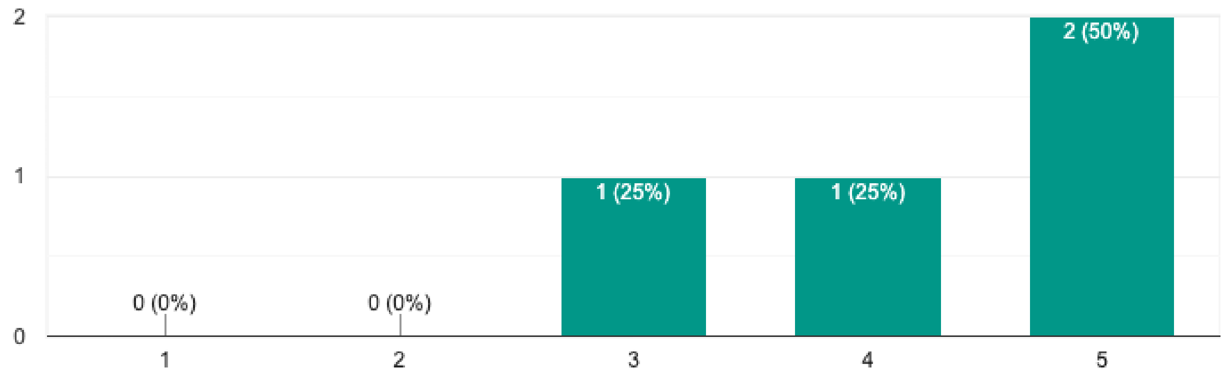
I would imagine that research developers and engineers would learn how to use this platform quickly

4 responses



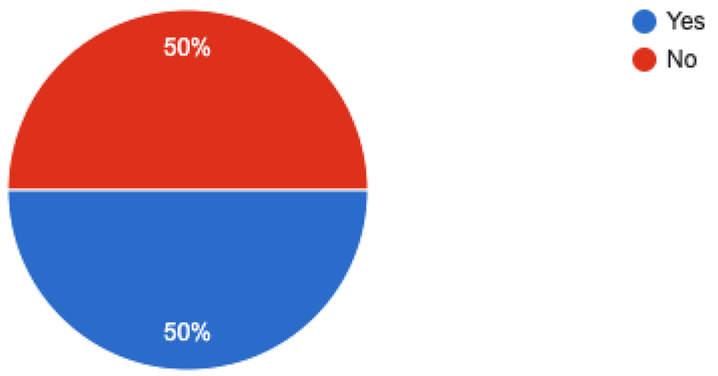
The cyclone-tracking use-case presented helped me to understand how the DARE platform can meet my requirements

4 responses



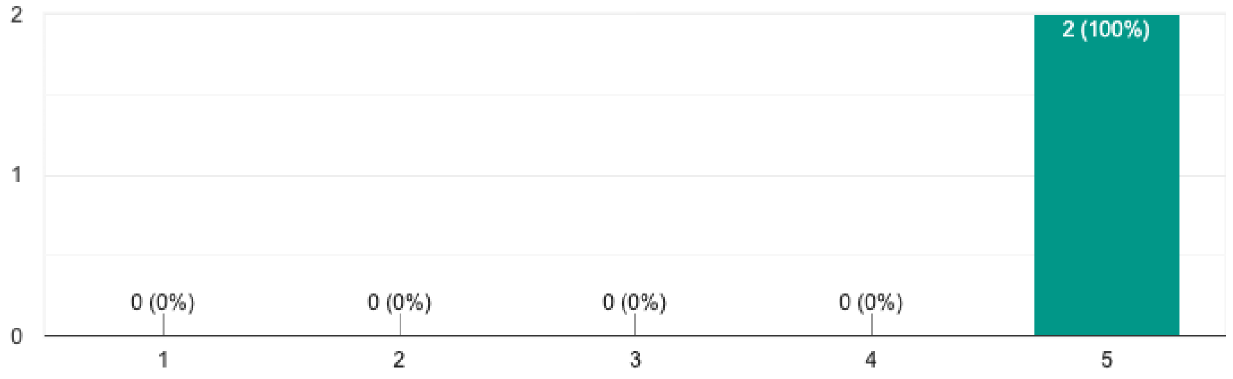
Did you participate in Part 2, on deployment and the hands-on session?

4 responses



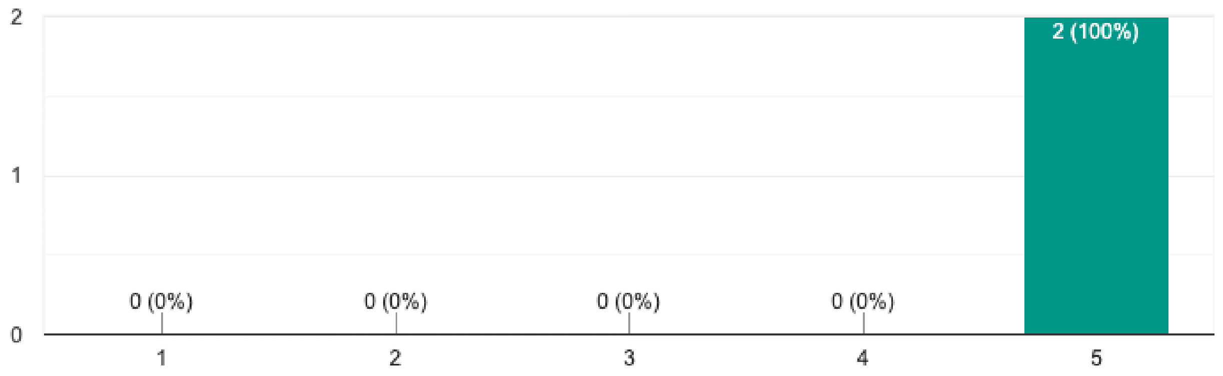
The hands-on experience was informative of the platform's capabilities

2 responses



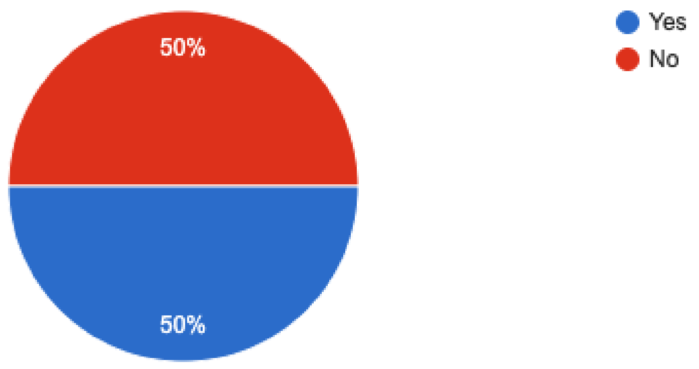
The webinar provided enough information to acquire and deploy the DARE platform

2 responses



Would you like to participate in a follow-up interview, to help us assess the usability of the DARE platform?

2 responses



3.3 Evaluation Interviews

The evaluation interviews aimed to assess the usability of the DARE platform in general terms without focusing on a particular use case. The data analysis thereby provided an overall feedback which applies to both climate and seismological (Deliverable D6.4) scientific applications due to the level of participation and the characteristics of the participants. Thus, this section is also reported in Deliverable D6.4 (§3.2.2.2) for completeness, and in Deliverable D2.2 (§5.3.2). The readers could skip to §4.

Aims

The interview-study aims were to evaluate:

1. ease of use
2. user satisfaction
3. utility of the DARE platform
4. DARE impact on the speed of the engineers' responses to research requirements
5. impact of the platform on researchers' productivity
6. impact of the platform on the innovation in research community

In addition, the study aimed to collect:

1. usability and functionality issues
2. suggestions to improve the platform

Participants

We have recruited 6 participants, from various backgrounds. Two of them helped us to pilot the interview questions. All of them have been involved in the development of the platform to varying extents, and they all have programming skills.

Procedure

We conducted semi-structured interviews to obtain insights arising from participants' perspectives and experiences. The interview questions were developed to probe several areas of exploration, such as the ease of use of the DARE platform, its learnability, its potential for integration with other services, and for automation of research methods. They asked about data-use policies.

Initially, we piloted the interviews with two DARE team members. This tested our interview questions and interview data-collection procedure.

Data Collection and Analysis

The data has been collected online, using the Zoom platform for video-conferencing. The sessions were recorded for analysis purposes. Similar to the previous evaluation (§5.3.1), a thematic analysis top-down approach was employed. Codes were given to the participants (from P1 to P6) to maintain anonymity in reports and publications.

Results and Discussion

Here is a summary of main findings of the interviews, clustered thematically. Readers are referred to [Constantin & Andries, 2020] for a complete report.

Successful experience: overall, all the participants found that the experience of using DARE led to success. Discussing what contributed to their success, they mentioned that developing targets for use cases was accomplished (N=2)[3], and the structure of the platform in itself was deemed to be versatile and functional, because it "*hides some of the complexity of the execution*" (P3). Its components have also been mentioned as contributing to a successful experience with the platform, primarily the workflows that are already set up (N=4), or the fact that DARE sets up the environment with all the libraries that the user may need (P6).

Ease of use: the participants generally found the platform easy to use (N=5). We employ caution when presenting this result as all interviewees were in some way involved in its development. They were able to identify potential areas for improvement, and suggestions for additional features and training to improve the use of the platform.

Less successful platform features: the participants discussed these in terms of features that might be missing and could be added, referring to the lack of maturity of the platform, rather than unsuccessful features. Examples included the lack of a user interface, for finding files and logs (P4). Instead they had to use the DARE API for that purpose (P4). The difficulty of installation (P5), the potential assumption that the users should have some cloud computing knowledge to use the platform (P6), as well as computational and programming skills (P3). There was confusion around the provenance and the API documentation regarding types of data (P2).

Training needed for using the platform: all the participants agreed that training would be beneficial for the users and developers. More specifically, an initial training stage was described as essential as the platform has plenty of functionality to offer, and the opportunity and support to explore that in depth should be provided (N=2). Perhaps some training, more specifically aimed at individuals who may not have a background in computer science (N=2), to provide support with the development of workflows and helper functions would be useful. Providing examples which can be sorted by functions,

[3] Indicates number of interviewees that gave this response, 2 in this case.

objectives, etc. was also mentioned (P5)). Videos to explain how to register an application, and videos to introduce each aspect of the platform's functionality (P6).

Responsibility for knowledge transfer: all the participants agree that the developers have the initial responsibility for providing support (e.g. to organise webinars). A collaboration between research engineers and domain-specific engineers was also mentioned (N=2). Lastly, one of the participants suggested a community forum could be developed over time, similar to Stack Overflow, provided that enough members would actively engage.

Productivity and innovation in communities: all the participants agreed that by using the platform, the productivity of the users would increase. This was motivated by explaining that the platform is aimed at reducing the engineering time by hiding the technical details (N=3), allowing the users to spend more time developing their specific applications and, consequently, on research. More specifically, the platform can be used via an API call, not needing to worry about the workflows or the infrastructure (N=2). Most of the participants thus considered that, by reducing the time previously spent on infrastructure complexities, the platform can accelerate innovation in the users' communities, by enabling them to focus more on research.

Integration with external and local services: all the participants considered that the platform integrated very well with both external and local services, given the services that were tested so far. Examples of good integration included the European seismic archives which could be downloaded for use (P3). DARE was described as modular and independent thanks to Kubernetes (P4) and well connected to the climate computational resources infrastructure (P5). DARE was also described as one of the first platforms to allow the communities to transfer into a cloud, which should be the future of operations (P6).

Automation of research methods and development practices: the participants knew that automation was one of the goals of DARE (to hide some technical details) and they all believe that this was achieved to some extent. More specifically, all resources are shareable between users, by enabling the use and sharing of workflow systems, and large-scale parallelisation without the users' input.

Additional automation and functionality can be added to the platform, depending on a community's needs (N=2), e.g., graphical interfaces could help beginners (P2), the addition of a desktop version with the same libraries, for local deployment (P6), as well as a visual representation of the user's repository (P6) may be helpful.

Summary and caveat

We could have recruited more responders and conducted more interviews had face-to-face events been possible. The participants supported the view that the platform is usable, and that it should be easy to use (depending on the users' background and skills). The platform supports automation of practices, and it allows additional functionality to be added. Training should be provided by developers in the early stages and extended by the communities themselves, to facilitate understanding of the different opportunities enabled by the platform. Some of the replies indicate limitations in the training, e.g., regarding difficulty in finding files and logs when the data catalogue and P4 have these facilities. However, readers are warned that all the interviewees were or had been members of the DARE project, so these results should be treated with caution.

4. Next Steps

At the beginning of the DARE project, the levels of development of the Use Case implementation were defined as (Level 1 in green, Level 2 in yellow, Level 3 in red):

Interface	Details	Themes
DARE (dispel4py) and ESGF Data Nodes	Enable the DARE Platform to download data from ESGF Data Nodes, with proper authentication	Workflow Authentication
DARE (dispel4py) and ENES CDI C4I	Enable C4I to trigger the execution of workflows using the DARE API, and retrieve the results along with provenance/lineage.	Workflow Provenance Authentication
PE Mapping Functions	Mapping of dispel4py PEs to calculation/processing functions (e.g. icclim python package). Add more provenance custom information.	Workflow Provenance
C4I GUI Wizard	Develop and Implement a Wizard on the C4I front-end to design and execute workflows	Workflow
DARE API and ESGF Computing Nodes	Delegate calculations to the ESGF Computing Nodes before accessing data on the Data Nodes	Workflow Provenance Authentication

With this evaluation feedback and suggestions, the following modified plan was proposed, with Level 1 (green) being already completed, for the phase 2 of developments, before the mid-term review.

DARE API and B2DROP	Enable the DARE Platform to read input data or store results into EUDAT B2DROP Service	Workflow Authentication
DARE API and ENES CDI C4I	Enable C4I to trigger the execution of workflows using the DARE API, and retrieve the results along with provenance/lineage.	Workflow Provenance Authentication

PE Mapping Functions	Mapping of dispel4py PEs to calculation/processing functions (e.g. icclim python package). Add more provenance custom information.	Workflow Provenance
DARE API and ESGF Data Nodes	Enable the DARE Platform to download data from ESGF Data Nodes, with proper authentication	Workflow Authentication
DARE API and C4I	Properly describe more complex workflows	Workflow
C4I icclim processing backend	Develop and implement a more efficient parallel processing (xarray & dask)	Workflow
Climate Domain Use Case	Make Use Case totally generic. Add more custom provenance and lineage.	Workflow Provenance
DARE API and ESGF Computing Nodes	Delegate calculations to the ESGF Computing Nodes before accessing data on the Data Nodes	Workflow Provenance Authentication
C4I GUI Wizard	Develop and Implement a Wizard on the C4I front-end to design and execute workflows	Workflow
C4I and DARE API	Implement proper workflow cancellation, restart and error tracking management	Workflow

After the mid-term review, the focus has been put into a novel Use Case, the Cyclone Tracking Use Case. So this plan has been completely changed to accommodate what was needed to implement for the new Use Case. The completed plan that has been accomplished is as follows. The two last items still in red will be pursued within the IS-ENES3 H2020 project and the ENES climate community, enhancing adoption of the DARE Platform and technology by the climate community.

DARE API and B2DROP	Enable the DARE Platform to store results into EUDAT B2DROP Service	Workflow Authentication
DARE API and ENES CDI C4I	Enable C4I (v2.0) to trigger the	Workflow

	execution of workflows using the DARE API, through the new Jupyter Notebook approach, using a container	Provenance Authentication
Direct interface to ESGF Data Nodes	Access input datasets directly through the ESGF Data Nodes, using OpenDAP and on-demand time and spatial subsetting	Workflow Provenance
CWL representation of the workflow	Adapt the existing Cyclone Tracking workflow to conform to CWL	Workflow Provenance
Parallel execution of selected CWL steps	Modify the existing Cyclone Workflow in CWL to enable parallel execution of specific steps	Workflow Provenance
Send configuration and parameters from C4I to the workflow using the DARE API	Implement a solution to send configuration and parameters of execution through the DARE API and into the CWL workflow	Workflow
C4I and DARE API	Implement proper workflow cancellation, restart and error tracking management.	Workflow
DARE API and ESGF Computing Nodes	Delegate calculations to the ESGF Computing Nodes before accessing data on the Data Nodes. Computing Nodes are not mature enough yet.	Workflow Provenance Authentication
C4I GUI Wizard	Develop and Implement a Wizard on the C4I front-end to design and execute workflows C4I is being completely redesigned (v2.0). A Jupyter Notebook interface is available.	Workflow

5. General Conclusions

The Climate Science Domain Pilot first prototype was evaluated in a special training session that took place in Utrecht on June 17th, 2019, while the new Cyclone Tracking Use Case was evaluated in an online webinar training session that took place October 16th, 2020. The targeted users are software developers and related roles of the Climate Research Infrastructure. A large majority of the participants had this expertise. Even if the event was only virtual, it was a success to be able to gather a significant amount of people within the targeted expertises.

The overall feedback is positive, and the evaluation helped to identify better what are the next actions with respect to sustainability and adoption within the climate community. One of the conclusions is that the interest is high as DARE is really filling a needed gap in the infrastructures' services, even if the landscape is quite busy with several technological options.

It is now a critical period for the adoption of DARE within the climate research community. With the DARE project ending on 31 Dec 2020, it is an opportunity to benefit from the fact that the H2020 IS-ENES3 project is still running for another 2 years. Moreover, two institutions (CERFACS and KNMI) are partners in both projects, and the same people are working together in both projects. This will enhance the adoption of the DARE technology into the climate community, especially because of the deep knowledge of the platform as well as the continuing contacts within the larger DARE community.

A1. Annex: Jupyter Notebook for the Cyclone Tracking Use Case

Jupyter Notebook repository: https://gitlab.com/project-dare/dare-examples/-/blob/master/wp7/tutorial/DARE%20platform%20tutorial.ipynb

DARE Platform tutorial

Setup & user authentication

First of all, in order to interact with the DARE platform, we provide a DARE platform client (helper_function.py), which needs to be downloaded.

- Using APIs interactively can be tedious

The client provides utility functions on the various DARE services.

In the following steps, we will use some of those functions to login to the platform, register or execute workflows, list files etc. Each time that you use a function from the *helper_manager.py*, take a moment to check the implementation in the respective python script, so as to have a better understanding of what the code does and how to interact with the platform.

Moreover, in the following code, we will download a second script named *cyclone_helper.py*, which does not interact with the DARE platform, but it will help us to collect all the necessary files for the cyclone use case in the second part of the tutorial.

```
# Imports

import json

from os import getcwd

from os.path import join, exists

import requests

# Download the DARE platform client - helper function library

hf_scripts = requests.get("https://gitlab.com/project-dare/exec-api/-/raw/master/client/helper_manager.py")

if hf_scripts.status_code == 200:

    with open("helper_manager.py", "w") as f:

    f.write(hf_scripts.text)

from helper_manager import DareManager

# download the helper script which will collect all the docker and CWL files for the Cyclone Use-case

cyclone_script = requests.get("https://gitlab.com/project-dare/exec-api/-
/raw/master/client/cyclone_helper.py")

if cyclone_script.status_code == 200:
        with open("cyclone_helper.py", "w") as f:

        f.write(cyclone_script.text)

import cyclone_helper as ch

credentials_file = "credentials.yaml"
```

Notice that in the above code block, we used a variable named *credentials_file*. This file contains your DARE & B2DROP credentials. If you already have downloaded and modified the credentials file, you can skip this step.

After executing the following code, update your workspace on the left and open the credentials.yaml file. Update the username and password fields with your DARE credentials and leave the issuer field as it is. If you have credentials for B2DROP update the *b2drop_username* and *b2drop_password* fields. You will need the B2DROP credentials at the end of the tutorial, if you want to upload the results of your workflow in B2DROP.

```
if not exists(credentials_file):

        credentials_yaml = requests.get("https://gitlab.com/project-dare/exec-api/-
/raw/master/client/example_credentials.yaml")

        if credentials_yaml.status_code == 200:

        with open(credentials_file, "w") as f:

        f.write(credentials_yaml.text)
```

We will now initialize the base URLs to the various services of the DARE platform.

Constants - Base URLs

```
BASE_URL = "https://platform.dare.scai.fraunhofer.de/"

LOGIN_HOSTNAME = BASE_URL + "dare-login"

EXEC_API_HOSTNAME = BASE_URL + "exec-api"

D4P_REGISTRY_HOSTNAME = BASE_URL + "d4p-registry"

WORKFLOW_REGISTRY = BASE_URL + "workflow-registry"

SPROV = BASE_URL + 'sprov'
```

Each one of the above constants, initializes a base URL to the DARE services, which are:

- The DARE login service in order to authenticate yourself and acquire a session token
- The Execution API, which can be used in order to execute dispel4py & CWL workflows, as well as to list files and folders and to download or upload files
- Dispel4py Workflow Registry, where you can register your dispel4py workflows.
- CWL Workflow Registry, where CWL workflows and respective docker images are stored.

In the following steps, we will demonstrate the use of each one of the aforementioned services.

Login to the DARE platform

In order to authenticate yourself on the platform and obtain a session token, you need to implement the following steps using the DARE platform client (helper_manager.py) which was downloaded. First, you need to configure the DareManager by providing the URLs to the DARE services, which were instantiated above and the name of the configuration file (credentials.yaml),

```
dm = DareManager(login_url=LOGIN_HOSTNAME,

        d4p_registry_url=D4P_REGISTRY_HOSTNAME,

        workflow_registry_url=WORKFLOW_REGISTRY,

        exec_api_url=EXEC_API_HOSTNAME,

        config_file=credentials_file)

token = dm.login()["access_token"]

print('Acquired the following token:')

print(token)
```

```
Acquired the following token:

eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJCWldyaGNjTnkwR3VwNk80NlA3OFdOM3ZjcUtFVTRLcVFqc
DBJWUNUX3BnIn0.eyJleHAiOjE2MDI4NTE3OTMsImlhdCI6MTYwMjgzNzM5MywianRpIjoiYzk5NWE0ZmQtYmZi
OS00ZWEyLTlmZmUtMDc0OWQ5MzFmOTM4IiwiaXNzIjoiaHR0cHM6Ly9rZXljbG9hay5kYXJlLnNjYWkuZnJhdW5
ob2Zlci5kZS9hdXRoL3JlYWxtcy9kYXJlIiwiYXVkIjoiYWNjb3VudCIsInN1YiI6IjRkMWRkMDk3LWJkNjYtNGE4Zi1hNG
U3LWRjN2RmYWExYzU1ZiIsInR5cCI6IkJlYXJlciIsImF6cCI6ImRhcmUtbG9naW4iLCJzZXNzaW9uX3N0YXRlIjoiYjg4Z
GRjZDQtMDY1Zi00M2Q4LTk2NjUtOTczN2QzZjA5NGE4IiwiYWNyIjoiMSIsImFsbG93ZWQtb3JpZ2lucyI6WyJodHR
wczovL3BsYXRmb3JtLmRhcmUuc2NhaS5mcmF1bmhvZmVyLmRlL2RhcmUtbG9naW4iXSwicmVhbG1fYWNjZXNz
Ijp7InJvbGVzIjpbIm9mZmxpbmVfYWNjZXNzIiwidW1hX2F1dGhvcml6YXRpb24iXX0sInJlc291cmNlX2FjY2VzcyI6ey
JhY2NvdW50Ijp7InJvbGVzIjpbIm1hbmFnZS1hY2NvdW50IiwibWFuYWdlLWFjY291bnQtbGlua3MiLCJ2aWV3LXBy
b2ZpbGUiXX19LCJzY29wZSI6ImVtYWlsIHByb2ZpbGUiLCJlbWFpbF92ZXJpZmllZCI6dHJ1ZSwibmFtZSI6IkNocmlzdG
lhbiBQYWdlIiwicHJlZmVycmVkX3VzZXJuYW1lIjoiY3BhZ2UiLCJnaXZlbl9uYW1lIjoiQ2hyaXN0aWFuIiwiZmFtaWx5X
25hbWUiOiJQYWdlIiwiZW1haWwiOiJjaHJpc3RpYW4ucGFnZUBjZXJmYWNzLmZyIn0.fPvYijT_bK4m2OOXYx0H1I7
ThhZtSvfnkDmnoTP3HEUPUiPg0dkaedepUf1jla-
lfo_4raIbtakV7n0Anq889h6lGAphIzzff8RJO5irTaZwtXe1RE7cZiBzZlR2HNwec22upOzjna4nPbd_cTRF941S3SFyn0
CiOxohk4VVowAoByuL5N9cwRXvhje0VuFriFHTGGlQJpUa9wyhYC-
D3i7mb2zPdPAWeYSaqaXlEzjHLEwPtx2boiCSUbuQ5AMgyBhRHq4iI5DFRAf9PowWj4Hiuomh4xKJIlAFyHQSEXbV
BEHNJA3QPiiJ2nP0-ZgGRvFC4fEWWxZz8BBuh3bJPA
```

The DareManager will remember the URLs, the configuration file and your token for this Jupyter session. If you refresh your workspace, you will need to execute this step again.

CWL workflow: Cyclone use case

In this part of the tutorial, we will show you how to execute CWL workflows in the DARE platform. The steps are quite similar to those of a dispel4py execution. However, for the CWL workflows, we use a different workflow registry with a slightly different logic. For the dispel4py execution, the DARE platform includes a generic docker image which is used for any use case. As we mentioned in the execution step, you can obtain a more personalized execution environment by providing a requirements file.

Note that, similarly to the dispel4py registration, your username will be used in some names, versions or tags to avoid duplicate entries in the DB. In a real-life use case, you could re-use an existing docker or workflow and reference them by name/tag and name/version respectively.

Docker and CWL registration

For the CWL execution, the workflow registry allows you to register docker containers and associate them with CWL workflows. Therefore, the steps that should be followed are:

1) A user creates a Dockerfile and python/bash scripts and registers them in the platform.

2) An admin, downloads and checks the files, builds the image and update the aforementioned entry with a public URL of the docker image (e.g. in DockerHub, GitLab Registry etc)

3) A user registers a CWL workflow and associates it with the docker.

We will execute a simpler case now: since we have already the files for the docker and the workflow as well as a public Docker image of the cyclone use case, we will download them from the GitLab repository and we will use only the *register_cwl()* function to register both the docker and the workflow.

However, in a real use-case scenario, you should do the following steps to register the docker and afterwards use the *register_cwl()* to register the workflow and set the *register_docker* parameter to false.

```
from helper_manager import DareManager

dm = DareManager() # add the parameters as shown in the beginning

# complete the parameters

docker_params = {

        "docker_name" :"",

        "docker_tag": "",

        "script_names": "",

        "path": ""

}

# register the docker files

dm.register_docker(docker_params)

# ask an admin to check the files, build the image and update with a public URL

# using the following code

docker_params = {

        "docker_name" :"",

        "docker_tag": "",

        "url": ""
}

dm.provide_docker_image_url(docker_params)
```

You can use additional functions to download ,update, delete etc your registrations using the CwlManager instead of the DareManager.

Example:

```
from helper_manager import CwlManager

cm = CwlManager(cwl_url, username, token)

cm.download_docker(docker_name, docker_tag, local_path)
```

Check the CwlManager class of the *helper_manager.py* script for more information.

Since the docker and CWL files are already available, we will download them locally. For this case, we have prepared a use-case helper script (cyclone_helper.py), which we will use for the downloading. The script contains the two following functions to retrieve the docker and CWL files from the respective GitLab repositories. Below, we list the links to these repositories:

- *GitLab Repository with the Docker files*
- *GitLab Repository with the CWL workflow of the Cyclone Use Case*

```
# Download the use-case files

ch.download_docker_files()

ch.download_cwl_files()

# This is the docker name & tag that you will have to use!

docker_name = "cyclone_{}".format(dm.username)

docker_tag = "v1.0"

docker_url = "registry.gitlab.com/project-dare/dare-platform/cyclone:v1.0"

docker_folder = join(getcwd(), "docker_files")

script_names = ["entrypoint.sh", "input_files.txt", "input_files2.txt", "input_files3.txt",
"cyclone_config_CMIP6.json", "config_cmip6.txt"]
```

Note that in the above code, you are provided with the name and tag of the docker that you need to use. You are also provided with a URL with a public image of the cyclone docker. During this tutorial we will not build the image, but we will use the existing one in the GitLab Registry. The available public images for cyclone are listed *here*.

Note that if the docker is already registered in the DARE platform, you need only the docker_name and docker_tag for the CWL workflow registration and there is no need to re-register the docker. To do so, in the function *register_cwl()* use the *register_docker* parameter and set it as **False**.

We now have the necessary files and we can register our docker environment!

After the docker registration, you can still update your docker using one of the following functions:

- Fist configure the cwl_manager: *python from helper_manager import CwlManager cm = CwlManager(cwl_url, username, token)*
- Then, you can use one of the following options:
  - the *update_docker()* function in order to update the name or the tag of the docker, to update the Dockerfile etc
  - the *add_script_to_existing_docker()* function to add a new script in your docker
  - the *edit_script_in_existing_docker()* function to edit a script of your docker and finally,
  - the *delete_script_in_docker()* function to delete a script that is not necessary any more

Check the *CwlManager* class in the *helper_manager.py* script for more details.

```
# This is the CWL parameters that you need for the registration

cwl_folder = join(getcwd(), "cwl_files")

workflow_name = "tracking_master.cwl"

workflow_version = "v1.0_{}".format(dm.username)

spec_name = "spec.yaml"

workflow_part_data = [

        {"name": "env_preparation.cwl"},

        {"name": "env_preparation.sh"},

        {"name": "processfiles.cwl"},

        {"name": "processfiles.py"},

        {"name": "processfiles.sh"},

        {"name": "transferfiles.cwl"},

        {"name": "transferfiles.py"},

        {"name": "transferfiles.sh"},

        {"name": "extractnc.cwl"},

        {"name": "extractnc.py"},

        {"name": "extractnc.sh"},

        {"name": "make_tracks.cwl"},

        {"name": "make_tracks.sh"},

        {"name": "xml2ascii.cwl"},

        {"name": "xml2ascii.sh"},
        {"name": "postprocess.cwl"},

        {"name": "postprocess.sh"},

        {"name": "plots.cwl"},

        {"name": "plots.py"},

        {"name": "plots.sh"}

]
```

The CWL workflows can be of two different classes, i.e. Workflow and CommandLineTool. The workflow class contains the steps of the workflow. A CommandLineTool CWL is just a step of a workflow. In the CWL Workflow Registry, we will associate our CWL of class Workflow with its steps. The list *workflow_part_data* contains all the CommandLineTool CWLs and the python and bash scripts that they use.

In the previous code block, we have provided you the CWL parameters you will need, along with the docker ones, so as to register your workflow. We will use the *register_cwl()* function as we mentioned above. Note that in the workflow_part_data list we use dictionaries. The reason is that CWLs could have their own spec yaml file instead of using only one (as we do in this case). Also, you can define the version of each CWL step, but for this use case we will use the same version as the parent workflow. To sum up, in a dictionary in the *workflow_part_data* list you can use the following keys: name, version, spec_name. Here, we use only the name key.

We will provide the necessary *docker parameters* (the variables we created in the above step) and the *cwl parameters* as **dictionaries**. There is an additional parameter, the *registered_docker*, which is True by default, and informs the DARE client that a docker registration should be performed. We will not specify it here, since we will use the default value.

The docker parameters should contain:

- the docker name

- the docker tag

- the docker URL

- the list of the script names

- the path to the scripts

The cwl parameters should contain:

- the workflow name

- the workflow version

- the name of the spec.yaml

- the path to the folder where we saved the workflow files

- the workflow_part_data dict

After the registration, you can refer to the docker by using only the name and the tag! Similarly, you can refer to your workflow by using only the name and the version!

Let's create fist the dictionaries for the parameters!

```
docker_params = {
        "docker_name": docker_name,
        "docker_tag": docker_tag,
        "url": docker_url,
        "script_names": script_names,
        "path": docker_folder
}
cwl_params = {
        "workflow_name": workflow_name,
        "workflow_version": workflow_version,
        "spec_name": spec_name,
        "path_to_cwls": cwl_folder,
        "workflow_part_data": workflow_part_data
}
# TODO register your workflow in the platform. Replace the None with the correct code
workflow = dm.register_cwl(cwl_params=cwl_params, docker_params=docker_params)
print("Request status: {}".format(workflow[0]))
workflow = json.loads(workflow[1])
print("Workflow id: {}".format(workflow["id"]))
print("Workflow name {} and version {}".format(workflow["name"], workflow["version"]))
```

```
Request status: 200
Workflow id: 28
Workflow name tracking_master.cwl and version v1.0_cpage
```

Now, the Cyclone CWL use case is registered in the DARE platform! To execute a CWL workflow, you should use the *exec_cwl()* function. The parameters here are optional. If you have registered your workflow in this Jupyter session, the DareManager will remember the name and version of the workflow that you used. Otherwise, the only mandatory parameters that you should provide are the *workflow_name* and *workflow_version*.

The remaining optional parameters are:

- input_data: provide a dictionary with input parameters. You can access them in your workflow as environmental variables. For example, in python:

```
import os
```

```
input_data = os.environ["INPUT_DATA"]
```

- nodes: provide an integer with the nodes that you want to be created to execute a job. Note that your application should support MPI to use this option. By default the nodes parameter is set to 1.

Let's execute & monitor our workflow!

```
# remember that in case you have already registered your workflow in a previous Jupyter session you need to add

# workflow_name = "tracking_master.cwl"

# workflow_version = "v1.0_{}".format(dm.username)

# as parameters of the following function

dm.exec_cwl()
```

```
(200, '{"run_dir": "cpage_20201016083957_9a78a21566d44336a7cccfd5fa9ea2f3", "run_id": "9a78a21566d44336a7cccfd5fa9ea2f3", "job_name": "cwl-fbed777067"}')
```

```
dm.monitor_job()
```

```
Running containers...
```

Provenance for CWL

Although with less interactive and customisation options than dispel4py, CWL workflows can also produce lineage. In DARE we support the interactive exploration of CWL provenance. For instance, in the image below we show the lineage of Cyclone Tracking in the *S-ProvFlow viewer*. Here the workflow processes, their outputs and dependencies are described adopting the metadata vocabularies supported by CWL.

provenance

provenance

Alternatively, the provenance can be downloaded as a data output and imported within compatible systems (Eg. openprovenance.org). The link below shows the provenance produced by the CWL Cyclone Tracking workflow that has been manually imported to the openprovenance.org.

https://openprovenance.org/store/documents/3239

Openprovenance.org can be used to visualise and share all the provenance produced in DARE. This can be achieved by clicking the Get W3C-PROV from the top menu bar of the S-ProvFlow viewer. The lineage of the workflow will be exported to rdf and can then be uploaded to openprovenance.org. Although openprovenance.org offers generic storage and visualisation of provenance documents, it can result overwhelming for large traces, thereby less practical for a in depth exploration.

Check the logs & output files

As we did in the dispel4py example, we will list our files to check if the workflow was successful. The *exec_cwl()* function returned the execution directory that you need to use. If you have just executed your job and the Jupyter Kernel is not cleared, you can go directly to the *list_exec_folder()* function without even providing the *run_dir* parameter!

In case you want to check your entire workspace (uploads and runs), use the following function to list the content of the *runs* and *uploads* directories.

```
dm.list_workspace()
```

```
Uploaded files......

Files generated from runs......

Api Local path: /home/mpiuser/sfs/cpage/runs/cpage_20201016083957_9a78a21566d44336a7cccfd5fa9ea2f3

Execution path:
/home/mpiuser/sfs/d4p/cpage/runs/cpage_20201016083957_9a78a21566d44336a7cccfd5fa9ea2f3
```

As mentioned, if you are running in a new Jupyter session, you need to copy the run directory you want to use from the output of the previous function. The run directory name follows the below pattern: <username>_<timestamp>_<run_id> *Once you found your directory and copied its name, use it in the* list_exec_folder() *function like this:*

```
my_run_dir=""

dm.list_exec_folder(run_dir=my_run_dir)
```

Let's list the files inside your run directory!

```
# Notice that if you have executed the CWL job in the same Jupyter session

# the DARE Manager remember the run_dir

# in any other case you need to provide it

run_dir = ""

dm.list_exec_folder()
```

```
Listing files......

Api Local path: va_day_INM-CM4-8_ssp585_r1i1p1f1_gr1_20500101-20541231.nc

Api Local path: tracks.xml

Api Local path: sftlf_fx_CanESM5_ssp585_r1i1p1f1_gn.nc

Api Local path: psl_day_INM-CM4-8_ssp585_r1i1p1f1_gr1_20150101-20641231.nc

Api Local path: psl_day_CNRM-CM6-1_ssp585_r1i1p1f2_gr_20150101-21001231.nc

Api Local path: images.json

Api Local path: tracks_20500101-20501231.txt

Api Local path: va_day_CanESM5_ssp585_r1i1p1f1_gn_20410101-20501231.nc

Api Local path: INM-CM4-8_20500101-20501231.nc
Api Local path: input_tracks.txt

Api Local path: ua_day_INM-CM4-8_ssp585_r1i1p1f1_gr1_20500101-20541231.nc

Api Local path: orog_fx_CanESM5_ssp585_r1i1p1f1_gn.nc

Api Local path: zg_day_CanESM5_ssp585_r1i1p1f1_gn_20410101-20501231.nc

Api Local path: BMNG_hiver.jpg

Api Local path: zg_day_INM-CM4-8_ssp585_r1i1p1f1_gr1_20500101-20541231.nc

Api Local path: CNRM-CM6-1_20500101-20501231.nc

Api Local path: zg_day_CNRM-CM6-1_ssp585_r1i1p1f2_gr_20400101-20641231.nc

Api Local path: total_density_1000.png

Api Local path: CanESM5_20500101-20501231.nc

Api Local path: PROVENANCE.zip

Api Local path: ua_day_CNRM-CM6-1_ssp585_r1i1p1f2_gr_20350101-20541231.nc

Api Local path: orog_fx_INM-CM4-8_historical_r1i1p1f1_gr1.nc

Api Local path: logs.txt

Api Local path: sftlf_fx_CNRM-CM6-1_amip_r1i1p1f2_gr.nc

Api Local path: tracks.nc

Api Local path: orog_fx_CNRM-CM6-1_amip_r1i1p1f2_gr.nc

Api Local path: psl_day_CanESM5_ssp585_r1i1p1f1_gn_20150101-21001231.nc

Api Local path: warmstart.txt

Api Local path: va_day_CNRM-CM6-1_ssp585_r1i1p1f2_gr_20350101-20541231.nc

Api Local path: ua_day_CanESM5_ssp585_r1i1p1f1_gn_20410101-20501231.nc

Api Local path: sftlf_fx_INM-CM4-8_historical_r1i1p1f1_gr1.nc
```

Finally, use the following code to download any output or logs file. Remember to provide the *directory* parameter if you do not want to use the *run_dir* in DareManager's cache or if you want to download files from another execution directory.

Additionally, as we mentioned in the dispel4py part, you can download files from the uploads folder. Use the kind parameter and set it to upload. If the file is directly stored under the uploads folder do not pass the dictionary parameter, otherwise provide the name of the uploads subfolder.

Finally, you can also specify a different path to store the file other than your current working directory by using the *local_path* parameter.

```
# provide also the directory parameter if you are running with a new session and therefore

# with a new DareManager
# the directory parameter should be set with the run_dir value defined in the previous code block

dm.download_file(filename="total_density_1000.png")

File is downloaded to path: /home/jovyan/dare-examples/wp7/tutorial/total_density_1000.png
```

Share files from DARE platform to B2DROP

For this step, we will use the B2DROP credentials that you updated in the first part of the tutorial.

In the *helper_manager.py* script, use the *b2drop_share()* function. The necessary parameters are:

- the kind of file you want to upload, i.e. directory or file

- the dare_path_kind, i.e. run or upload

- the remote directory, i.e. in b2drop, where you want to upload your files. Use only the relative path to your directory since "/remote.php/webdav/" prefix is added by the Execution API

- Optionally, if you want a different directory from the one stored in the session or if the session is empty, add the dare_directory parameter with the name of the directory (only the name, without the full path)

```
# possible values for kind: file or directory

# select if you want to upload a directory (which will be zipped) or a single file

kind = "file"

dare_path_kind = "run"

filename="total_density_1000.png"

remote_dir_name = "cyclone_tracking"

dm.b2drop_share(kind=kind, dare_path_kind=dare_path_kind, filename=filename,
remote_dir=remote_dir_name)
```

Clean up your uploads and/or runs

In case you want to cleanup your uploads and run directories, you should use the *cleanup_workspace()* function from the *helper_manager.py* script. Set up uploads and/or runs parameters to True/False to clean them up or not respectively. Then, list your workspace to check that it's cleaned.

```
# if you need to cleanup your runs use runs=True

# same for uploads. You can cleanup both, so just set both to True

response = dm.cleanup_workspace(uploads=True, runs=True)

print(response)

dm.list_workspace()
```