Co-ordinated by

MAchinE Learning for Scalable meTeoROlogy and climate

MAELSTROM

Final version of workflow tools published

Oliver Kindler, Kristian Ehlert, Fabian Emmerich,

Saleh Ashkboos, Thomas Nipen

www.maelstrom-eurohpc.eu

MAELSTROM

D2.5
Final version of workflow tools published

Author(s):	Oliver Kindler (4cast), Kristian Ehlert (4cast), Fabian Emmerich (4cast), Saleh Ashkboos(ETH), Thomas Nipen (MetNorway)
Dissemination Level:	Public
Date:	28/03/2024
Version:	1.0
Contractual Delivery Date:	31/03/2024
Work Package/ Task:	WP2/ T2.2 T2.3 T2.4 T2.5 T2.6
Document Owner:	4cast
Contributors:	4cast, ECMWF, ETH, MetNorway
Status:	Final

MAELSTROM

Machine Learning for Scalable Meteorology and Climate

Research and Innovation Action (RIA)
H2020-JTI-EuroHPC-2019-1: Towards Extreme Scale Technologies and Applications

Project Coordinator:	Dr Peter Dueben (ECMWF)
Project Start Date:	01/04/2021
Project Duration:	36 months

Published by the MAELSTROM Consortium

Contact:
ECMWF, Shinfield Park, Reading, RG2 9AX, United Kingdom
Peter.Dueben@ecmwf.int

The MAELSTROM project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955513. The JU receives support from the European Union's Horizon 2020 research and innovation programme and United Kingdom, Germany, Italy, Luxembourg, Switzerland, Norway

Contents

Figures

Tables

1 Executive Summary

This document presents a comprehensive overview of the advancements made in the development of the Mantik workflow platform, serving as the primary outcome of Work Package 2. The expectations on the platform are outlined in Deliverable 2.1 by delineating a weather and climate workflow and formulating requirements. Naturally, this report also provides information on the latest developments after Deliverable 2.4 in section 3.

Users benefit from the platform's ability to store applications, and to facilitate reproducibility, flexibility, and extensibility of Machine Learning (ML) solutions while fostering collaboration. Additionally, the Mantik Web interface allows for convenient benchmarking of MAELSTROM applications as well as integrated experiment tracking and versioning features via incorporating the open source library MLflow. Users can effortlessly replicate earlier stages of application development or adjust specific training parameters, significantly reducing overhead. Furthermore, comprehensive documentation of Mantik's functionalities is readily available online.
By project completion, the platform grants access to JSC and CSCS cluster nodes via the web interface, adhering to the highest security standards.

2 Introduction

2.1 About MAELSTROM

To develop Europe's computer architecture of the future, MAELSTROM will co-design bespoke compute system designs for optimal application performance and energy efficiency, a software framework to optimise usability and training efficiency for machine learning (ML) at scale, and large-scale ML applications for the domain of weather and climate science.

The MAELSTROM compute system designs will benchmark the applications across a range of computing systems regarding energy consumption, time-to-solution, numerical precision and solution accuracy. Customised compute systems will be designed that are optimised for application needs to strengthen Europe's high-performance computing portfolio and to pull recent hardware developments, driven by general ML applications, toward needs of weather and climate applications.

The MAELSTROM software framework will enable scientists to apply and compare ML tools and libraries efficiently across a wide range of computer systems. A user interface will link application developers with compute system designers, and automated benchmarking and error detection of ML solutions will be performed during the development phase. Tools will be published as open source.

The MAELSTROM ML applications will cover all important components of the workflow of weather and climate predictions including the processing of observations, the assimilation of observations to generate initial and reference conditions, model simulations, as well as post-processing of model data and the development of forecast products. For each application, benchmark datasets with up to 10 terabytes of data will be published online for training and ML tool-developments at the scale of the fastest supercomputers in the world. MAELSTROM ML solutions will serve as a blueprint for a wide range of ML applications on supercomputers in the future.

2.2 Scope of this Deliverable

2.2.1 Objectives of this Deliverable

The primary objective of this Deliverable is the description of the final public version of the Mantik workflow platform[1,2] (Task 2.2) and Graphical User Interface (Task 2.4). Beside enabling the reproduction, versioning and tracking of ML experiments, the platform shall also introduce the possibility to share solutions and experiences regarding benchmarking results, fostering communication and recommendations between application developers of WP1 (Task 2.3).
Data preprocessing tools and the entire data loading pipeline instrumentalise CliMetLab Plugins for the MAELSTROM applications to ensure fast access to the data sets (Task 2.5), leading to significant improvements in data processing as the first step of the workflow.
During the project, Mantik was reinvented as a universal interface to HPC infrastructure ,simplifying usability, while maintaining a necessarily high security standard in HPC handling (Task 2.6). After a solution is deemed final, subsequent users shall be enabled to adapt it easily.

[1] https://cloud.mantik.ai
[2] https://gitlab.com/mantik-ai

2.2.2 Work performed in this Deliverable

As a concluding report, this document will serve as a summary of the developments of Mantik during the MAELSTROM project. For more detailed information, see references provided in the text.

Through the integration of the HPC interface softwares UNICORE[3] and FirecREST[4] (maintained by the Jülich Supercomputing Centre (JSC) and the Swiss National Supercomputing Centre (CSCS), respectively), the Mantik web-platform enables users to execute ML applications on the JSC and CSCS HPC clusters. To achieve this, 4cast has been in extensive collaboration with the responsible developers of UNICORE and FirecREST.

MAELSTROM has been represented at several conferences, where 4cast presented the Mantik workflow platform (PASC22 [1], HiPEAC 2023 [2], EuroHPC Summit 2023 [3]). Following an invitation of the FirecREST team at CSCS, 4cast has contributed a talk to the SOS26 conference, where we demonstrated how FirecREST was incorporated into the Mantik platform to enable access to the CSCS cluster via Mantik. [4]

Since Deliverable 2.4 new features on user collaboration have been implemented, enabling role-based access control (RBAC). This allows users to form *Groups* and *Organizations* and assign specific roles to members, leading to different levels of access to the functionalities of Mantik projects.

A more detailed description of the updated architecture of Mantik is provided in section 4.

2.2.3 Deviations and countermeasures

The project aimed to provide a comprehensive ML development platform developed from scratch. However, it was discovered that MLflow already offers essential features for this purpose. As a result, Mantik integrated these functionalities, enabling users to store, reproduce, share, and benchmark ML solutions, along with its core feature as a universal interface for HPC infrastructure.

Initially, MAELSTROM aimed to automate benchmarking cycles quarterly, but experience taught that the necessity of executing a benchmark is best judged by the scientists themselves. To address this, an automated run submission feature was implemented in Mantik, allowing users to schedule benchmarking runs ahead and flexible on HPC clusters as needed. This approach offers a less resource-intensive alternative while still accommodating additional use cases beyond benchmarking.

For more details about the deviations related to the benchmarking infrastructure, the reader is referred to Deliverable D2.3 focusing on performance benchmarking.

[3] https://www.unicore.eu
[4] https://products.cscs.ch/firecrest

3 Progress by Work Package Tasks

This section provides a detailed update on the recent progress separated by Work Package Tasks.
It focuses on the improvements made since the last Deliverable 2.4. For a comprehensive view on the Mantik platform as a whole, the reader is referred to section 4.
General developments in the MAELSTROM workflow will be described from the perspective of the workflow platform.

3.1 Workflow Platform (Tasks 2.2)

The development of the web-based platform Mantik by the MAELSTROM project takes into account the requirements provided by weather and climate researchers. By integrating these with a standard ML workflow protocol and adjusting platform expectations throughout the project, several software objectives were identified to facilitate the production cycle. These have been described in prior Deliverables as well and are mentioned here again for the convenience of the reader:

- Utilising HPC clusters typically demands a deep understanding of specialised software environments managed by cluster administrators, imposing significant overhead on researchers. Mantik seeks to alleviate this burden by addressing the RestAPI of the cluster interface software. This primarily targets the JSC cluster, where numerous large datasets crucial to MAELSTROM's weather and climate research are hosted. In addition, we plan to integrate additional HPC infrastructure relevant to the project, such as cluster nodes of E4 or the Swiss cluster CSCS through their interface FirecREST.

- To ensure the reproducibility of our ML solutions, every aspect of the workflow must integrate smoothly with the platform. This includes fundamental data management, application code, and the various experiments conducted, each representing different model variations and their corresponding output metrics. By hosting these assets on the platform, users can effectively interact with their applications on HPC infrastructure, facilitating tasks such as model training, experiment tracking and the ability to replicate results at any given moment in the future.

- We aim to foster collaboration and the establishment of interest groups within the research community. Platform users should have the capability to share their ML applications with selected researchers or the public, promoting knowledge exchange and enhancing ML solutions.

- To facilitate the ongoing improvement of solutions, the software framework should encourage the exchange of well-performing ML applications among users. Hosting a diverse range of applications, properly categorised, sets the stage for easier access to cutting-edge ML solutions for both newcomers and seasoned researchers alike.

More details on the architecture of Mantik are given in Section 4.
To address the requirements listed above, the following functionalities have been implemented:

- The open-source ML framework MLflow[5] [2] is used for tracking user-chosen training parameters and for model versioning. It provides a large range of functionalities to support the workflow of ML developers.

- MLflow does not provide any security measures when hosting its features publicly i.e. providing the possibility of restricting access. Therefore, we integrated features that facilitate user management and limit access to authorised individuals, ensuring secure access to the AWS cloud instance hosting our MLflow platform.
  Personal credentials to HPC infrastructure can be added in user settings, so a run can be scheduled via the GUI. Furthermore, RBACis implemented, allowing users to form *Groups* and *Organizations.*

- To offer users access to HPC resources, we created the Mantik Compute Backend. It provides a REST API for directly submitting ML applications to the computational sites, and its usage is also provided in a more convenient way through the frontend of the Mantik platform. In order to use the MLflow functions, applications must adhere to the structure of MLprojects, a format structure given by MLflow. MLprojects ensure that ML applications execution is handled flawlessly on all platforms.

- The JSC cluster is crucial for the MAELSTROM project, since most data is hosted there, and their hardware systems are used by all MAELSTROM Applications to train the ML models. Their interface software UNICORE was therefore integrated by Mantik developers to allow interacting with their systems through the Mantik platform. Additionally, we integrated the FirecREST interface of CSCS. The workflow platform is able to send requests and schedule runs via a GUI through UNICORE and FirecREST.

[5] https://mlflow.org

- A Mantik Python package was developed[6]. It provides users with a Command line interface (CLI), offering authentication, data transfer, and run submission:
    - To grant MLflow users access to the cloud instance, the package can authenticate users on the platform and provide them with access to the secure MLflow services from Python applications.
    - Data transfer is possible to S3 buckets as well as HPC sites that allow access via UNICORE. Files can be transferred between the cluster and local machines. Moreover, it allows to perform all common file system operations on the cluster: viewing directory contents, and creating, copying, or deleting directories or files.
    - The package offers Pythonic access to the Compute Backend API, allowing users to directly run and oversee their ML applications on the preferred HPC system from any location.

  The usage of the package is documented in the Mantik Documentation.[7]

- Presently, Mantik is a fully operational web platform, encompassing all the outlined requirements, not previously mentioned recent updates include:
    - Tutorials have been added to improve the overall user experience.
    - The development progress is documented in detail in the Changelogs[8].

[6] https://pypi.org/project/mantik
[7] https://mantik-ai.gitlab.io/mantik/cli.html
[8] https://mantik-ai.gitlab.io/mantik/changelog.html#added

3.2 Benchmarking (Task 2.3)

For benchmarking the applications inside MAELSTROM, we widely use Deep500 [5], developed by ETH Zurich. Deep500 is a modular cross-platform benchmarking infrastructure that helps to evaluate the performance (and accuracy) of deep learning programs. Deep500 is designed in a way that breaks down deep learning into different levels (operators, network processing, training, and distributed training) and measures each level separately. The code is open-source and can be found on GitHub[9].

Within MAELSTROM, ETH developed an interface for Deep500 to measure weather and climate applications. This is developed mainly for supporting the applications in PyTorch and TensorFlow with an easy-to-use interface. We performed two software benchmarks for evaluating all applications within MAELSTROM and reported the results in deliverables 1.3 and 2.6. This improvement focused mainly on making it easier to mark and note important parts of an application for timing reasons, such as I/O and backpropagation, while also keeping track of the total time it takes to run. We created a straightforward script to test all applications. The code and tutorials are available on GitHub[10].

As of now, we used Deep500 to measure different aspects of MAELSTROM applications. These aspects include batch computation, epoch computation, forward and backward pass, and I/O. Using such a benchmarking tool, we unified all the applications benchmarking (regardless of their framework) and show that Deep500 is a fully functional tool that is instrumentalized by all MAELSTROM applications for benchmarking.

All application developers additionally used the command line tool JUBE[11] to orchestrate their execution of benchmark runs. JUBE was developed by JSC to simplify the process of benchmarking on various machines. It automatizes HPC job submissions for user-specified parameters relevant to the benchmarking of applications. In addition users get a convenient overview of their results.

[9] https://github.com/deep500/deep500
[10] https://github.com/sashkboos/Deep500-for-MAELSTROM
[11]https://www.fz-juelich.de/en/ias/jsc/services/user-support/software-tools/jube?expand=translations,fzjsettings,nearest-institut

3.2. User Interface (Task 2.4)

In this subsection, we detail recent progress on the User Interface, as presented in Deliverable 2.4, pages 12-24, subsection “3.2. User Interface (Task 2.4)”. The text is formulated in an analogous style of writing to support comprehensibility.

Here, the RBAC will be presented as well as Run submission details. All user settings, currently available and planned will be described. An outlook on development plans after the MAELSTROM project is provided.

3.2.1. Role-based access control (RBAC) and user management

To foster collaboration and shareability of ML solutions, the Mantik workflow platform provides integrated features of user management. It is a crucial component of any web-based platform, encompassing various core functions to ensure secure and efficient interactions between users and the system. It involves enabling users to register accounts by providing necessary information and credentials. Once registered, users can manage their profiles by updating personal details and preferences. Password management allows users to reset their passwords. Naturally, the user email address can be updated at any time.

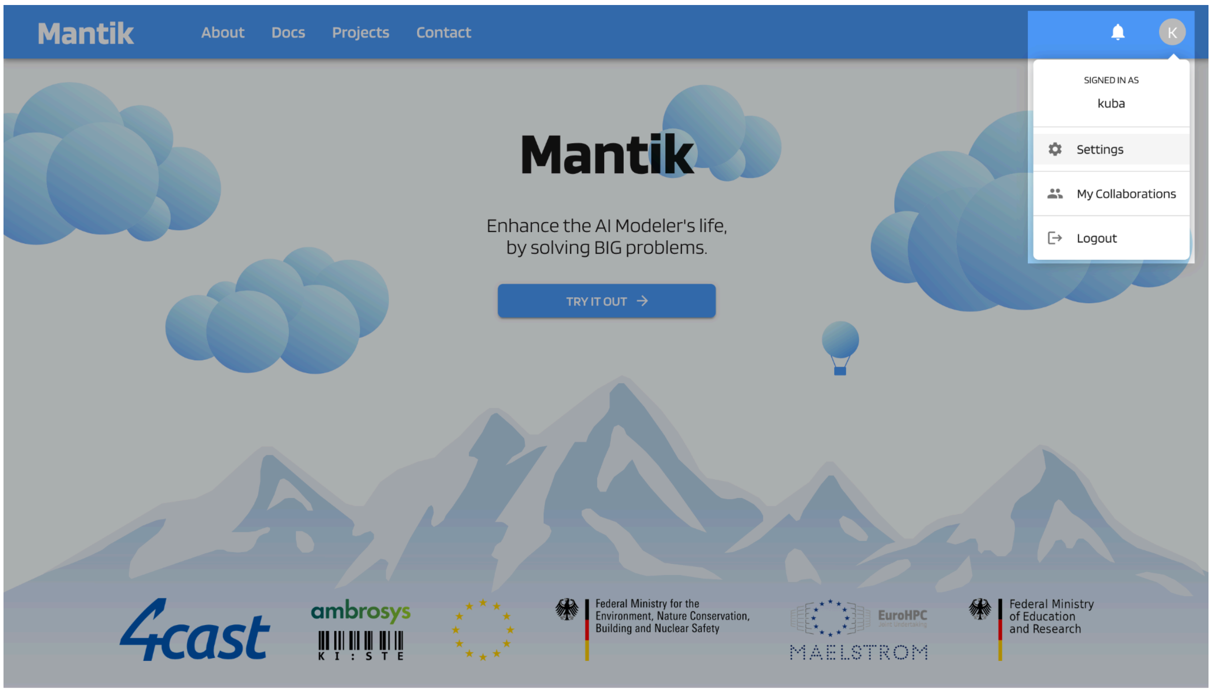
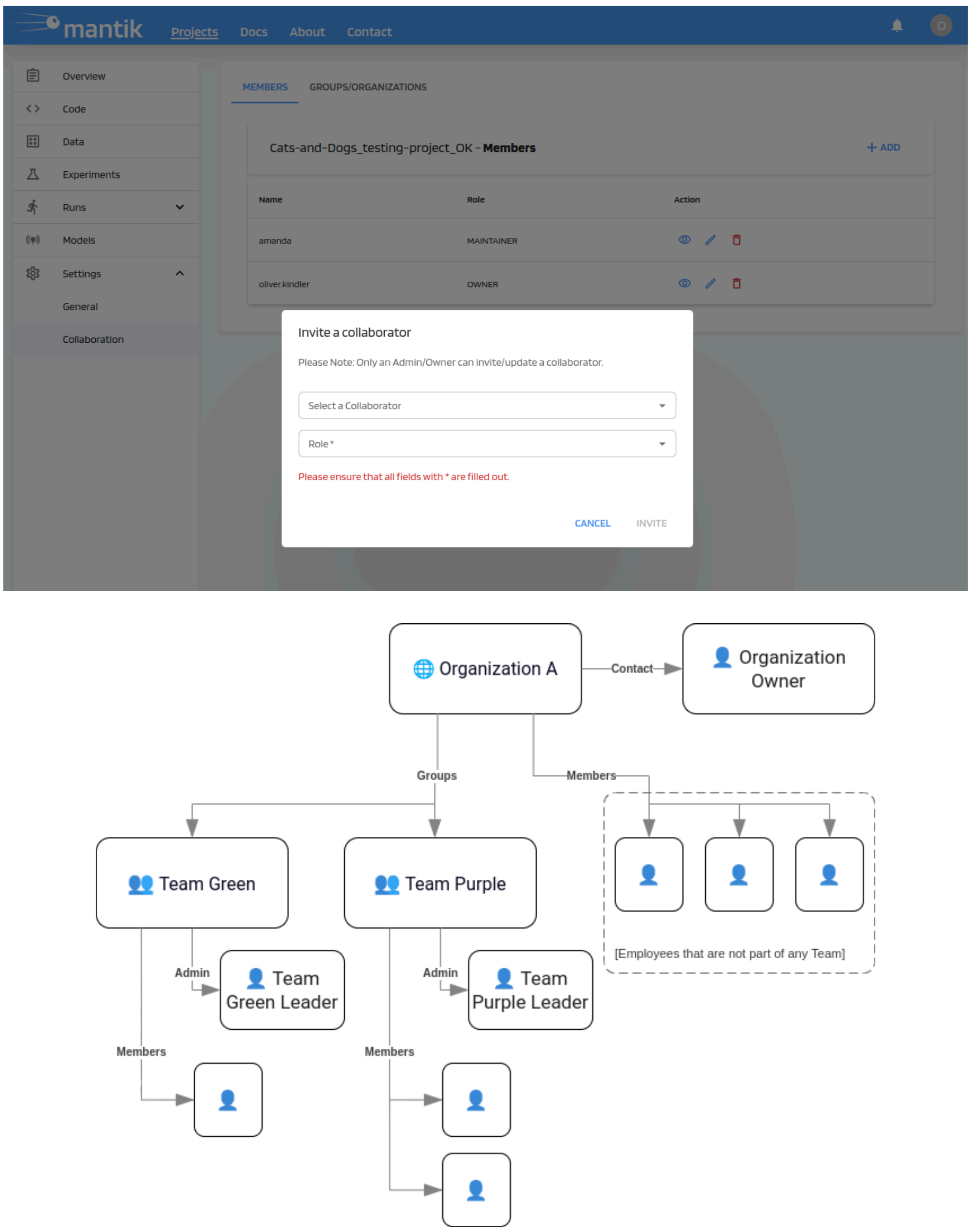

Mantik
About
Docs
Projects
Contact
K
SIGNED IN AS
kuba
Settings
My Collaborations
Logout
Mantik
Enhance the AI Modeler's life, by solving BIG problems.
TRY IT OUT
4cast
ambrosys
KI:STE
Federal Ministry for the Environment, Nature Conservation, Building and Nuclear Safety
EuroHPC
MAELSTROM
Federal Ministry of Education and Research

***Figure 1**: Collaboration menu in Mantik.*

In order to collaborate with each other and share solutions, users can arrange themselves in two main structures, *Groups* and *Organizations.* While *Groups* will only consist of users, *Organizations*

will be able to add entire *Groups* and users alike (see Figure 2, bottom). To create a *Group* or *Organization* the user has to enter the 'My Collaborations' menu, as seen above in Figure 1.
As written in Deliverable 2.4 Mantik organises the ML workflow in *Projects.* The tab 'Collaborations' has been added recently to the settings tab of the *Project* page (Figure 2, top). Here, the owner of a *Project* can invite other users, *Groups* or entire *Organisations.*



Cats-and-Dogs_testing-project_OK - **Members**

Name	Role	Action
amanda	MAINTAINER	
oliver.kindler	OWNER	

Invite a collaborator

Please Note: Only an Admin/Owner can invite/update a collaborator.

Select a Collaborator

Role *

Please ensure that all fields with * are filled out.

Figure 2: *Project Collaboration page (top) and structure of user Groups / Organizations (bottom).*

If a user is invited to a *Project,* the invitation will show up at the Bell icon left to the user menu on the top bar. Invites to *Groups* or *Organisations* will show up for the respective administrator. Every user, *Group* or *Organization* has to be assigned one of five different roles by the *Project* owner as part of the Invitation.

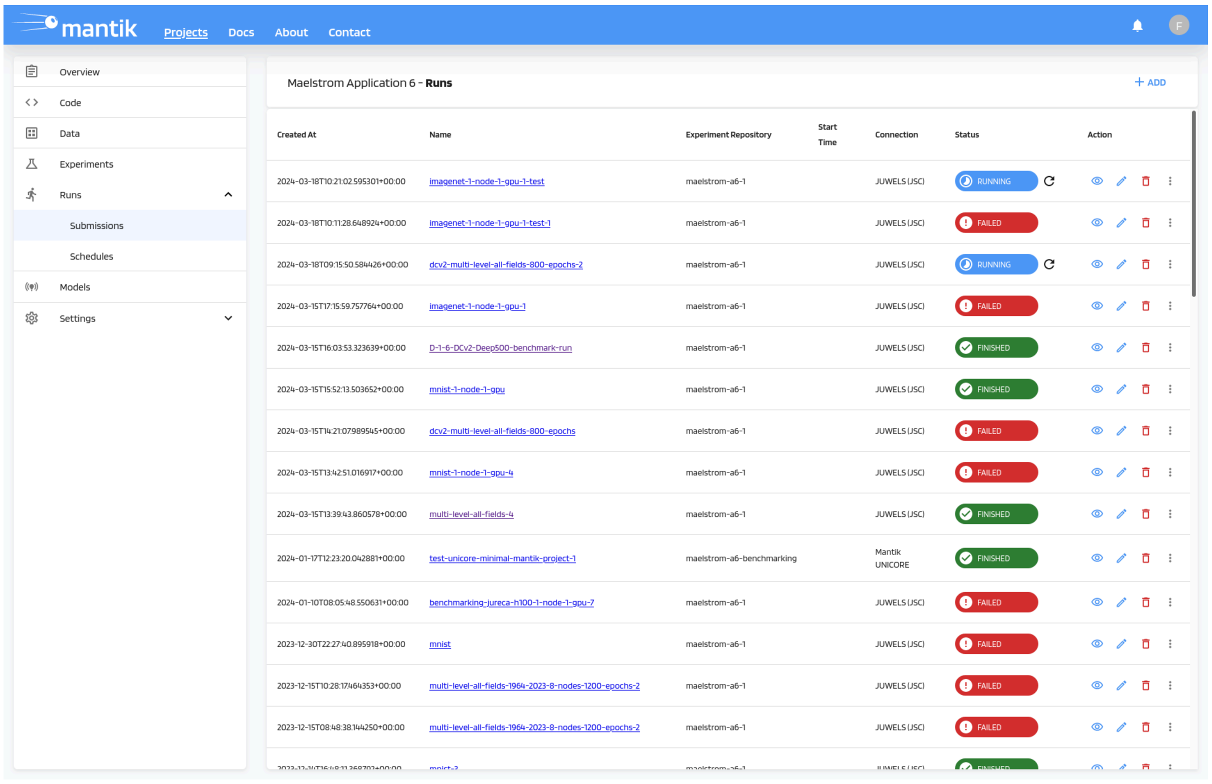
Roles are hereditary i.e. every user of a *Group* or *Organization* will be assigned the same role. Project owners are therefore advised to not assign roles with a higher level than Reporter to entire *Groups* or *Organizations*. Should a member of a *Group* or *Organization* need higher level access, roles can be overwritten by a personalised invitation to the user.

Project Role	Description	Rights
Guest	A user who visits the project	Read only.
Reporter	A user that is part of a project, but not involved in the research	All of Guest and deploy models for inference
Researcher	A user who is doing the research within the project.	All of Reporter, update code, data, run, experiment and model repositories.
Maintainer	A user who is managing the project	All of Researcher and deploy code from a repository for e.g. training, update project info.
Owner	A user who owns the project	All of Maintainer and change all project settings including manage user/user group roles, invite users/user groups to the project and delete the project.

***Table 1**: Project roles and respective rights in Mantik.*

3.2.2. Run submission details

After a Run has been submitted to an HPC cluster (see D2.4 section 3.2.2 on model training and data on HPC clusters, p. 18), it will show up as an entry in the *Run Submissions* tab of the *Project* page as seen in Figure 3. For every submission to the cluster, the user is given a set of information. This includes the creation date, a user-chosen name, the corresponding MLflow Experiment, the cluster it was submitted to and the status of the Run. The status indicates the state of a run in the process of the scheduling mechanisms of the respective HPC site.

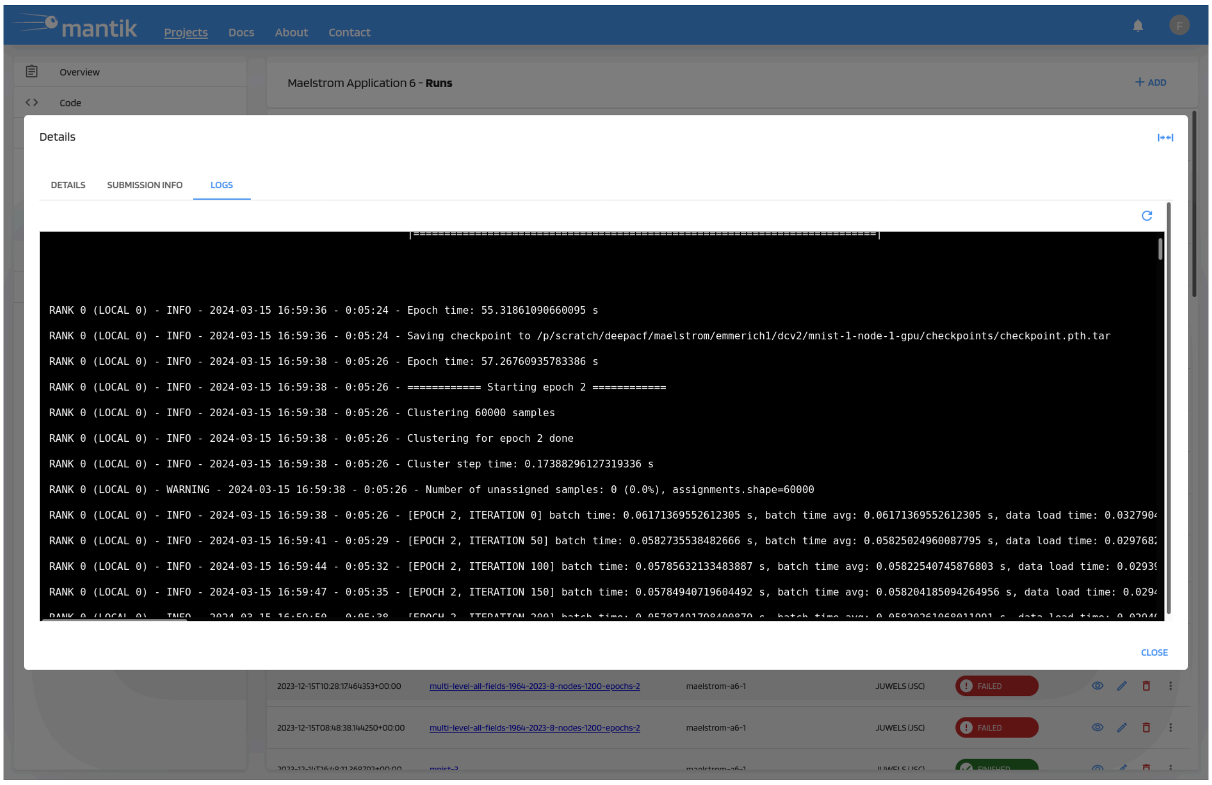


Maelstrom Application 6 - **Runs**

Created At	Name	Experiment Repository	Start Time	Connection	Status	Action
2024-03-18T10:21:02.595301+00:00	imagenet-1-node-1-gpu-1-test	maelstrom-a6-1		JUWELS (JSC)	RUNNING	
2024-03-18T10:11:28.648924+00:00	imagenet-1-node-1-gpu-1-test-1	maelstrom-a6-1		JUWELS (JSC)	FAILED	
2024-03-18T09:15:50.584426+00:00	dcv2-multi-level-all-fields-800-epochs-2	maelstrom-a6-1		JUWELS (JSC)	RUNNING	
2024-03-15T17:15:59.757764+00:00	imagenet-1-node-1-gpu-1	maelstrom-a6-1		JUWELS (JSC)	FAILED	
2024-03-15T16:03:53.323639+00:00	D-1-6-DCv2-Deep500-benchmark-run	maelstrom-a6-1		JUWELS (JSC)	FINISHED	
2024-03-15T15:52:13.503652+00:00	mnist-1-node-1-gpu	maelstrom-a6-1		JUWELS (JSC)	FINISHED	
2024-03-15T14:21:07.989545+00:00	dcv2-multi-level-all-fields-800-epochs	maelstrom-a6-1		JUWELS (JSC)	FAILED	
2024-03-15T13:42:51.016917+00:00	mnist-1-node-1-gpu-4	maelstrom-a6-1		JUWELS (JSC)	FAILED	
2024-03-15T13:39:43.860578+00:00	multi-level-all-fields-4	maelstrom-a6-1		JUWELS (JSC)	FINISHED	
2024-01-17T12:23:20.042881+00:00	test-unicore-minimal-mantik-project-1	maelstrom-a6-benchmarking		Mantik UNICORE	FINISHED	
2024-01-10T08:05:48.550631+00:00	benchmarking-jureca-h100-1-node-1-gpu-7	maelstrom-a6-1		JUWELS (JSC)	FAILED	
2023-12-30T22:27:40.895918+00:00	mnist	maelstrom-a6-1		JUWELS (JSC)	FAILED	
2023-12-15T10:28:17.464353+00:00	multi-level-all-fields-1964-2023-8-nodes-1200-epochs-2	maelstrom-a6-1		JUWELS (JSC)	FAILED	
2023-12-15T08:48:38.144250+00:00	multi-level-all-fields-1964-2023-8-nodes-1200-epochs-2	maelstrom-a6-1		JUWELS (JSC)	FAILED	
2023-12-14T16:48:11.368792+00:00	mnist-3	maelstrom-a6-1		JUWELS (JSC)	FINISHED	

Figure 3: *Run submission list of a project with action buttons on the right side.*

Mantik offers the user the possibility to interact with the Run through several action buttons. These functions will be explained in detail in this subsection.
By clicking on the eye symbol, the user can access detailed information about the submitted run (see Figure 4). Logs delivered by the HPC cluster can be viewed through the Mantik GUI as well as the submission info from Mantik. This feature allows the researcher to access details provided by the cluster, enhancing the workflow. If something went wrong and a submission failed, adjustments can be made easily via the re-run option. This can be found when clicking the sandwich menu on the right hand side of the action buttons. Here, the original run form re-opens, adjustments to the backend configuration can be performed and then be submitted with minimal effort. Naturally, runs can be deleted or renamed if needed.



```
RANK 0 (LOCAL 0) - INFO - 2024-03-15 16:59:36 - 0:05:24 - Epoch time: 55.31861090660095 s
RANK 0 (LOCAL 0) - INFO - 2024-03-15 16:59:36 - 0:05:24 - Saving checkpoint to /p/scratch/deepacf/maelstrom/emmerich1/dcv2/mnist-1-node-1-gpu/checkpoints/checkpoint.pth.tar
RANK 0 (LOCAL 0) - INFO - 2024-03-15 16:59:38 - 0:05:26 - Epoch time: 57.26760935783386 s
RANK 0 (LOCAL 0) - INFO - 2024-03-15 16:59:38 - 0:05:26 - ============ Starting epoch 2 ============
RANK 0 (LOCAL 0) - INFO - 2024-03-15 16:59:38 - 0:05:26 - Clustering 60000 samples
RANK 0 (LOCAL 0) - INFO - 2024-03-15 16:59:38 - 0:05:26 - Clustering for epoch 2 done
RANK 0 (LOCAL 0) - INFO - 2024-03-15 16:59:38 - 0:05:26 - Cluster step time: 0.17388296127319336 s
RANK 0 (LOCAL 0) - WARNING - 2024-03-15 16:59:38 - 0:05:26 - Number of unassigned samples: 0 (0.0%), assignments.shape=60000
RANK 0 (LOCAL 0) - INFO - 2024-03-15 16:59:38 - 0:05:26 - [EPOCH 2, ITERATION 0] batch time: 0.06171369552612305 s, batch time avg: 0.06171369552612305 s, data load time: 0.032790
RANK 0 (LOCAL 0) - INFO - 2024-03-15 16:59:41 - 0:05:29 - [EPOCH 2, ITERATION 50] batch time: 0.0582735538482666 s, batch time avg: 0.05825024960087795 s, data load time: 0.029768
RANK 0 (LOCAL 0) - INFO - 2024-03-15 16:59:44 - 0:05:32 - [EPOCH 2, ITERATION 100] batch time: 0.05785632133483887 s, batch time avg: 0.05822540745876803 s, data load time: 0.0293
RANK 0 (LOCAL 0) - INFO - 2024-03-15 16:59:47 - 0:05:35 - [EPOCH 2, ITERATION 150] batch time: 0.05784940719604492 s, batch time avg: 0.05820418509426456 s, data load time: 0.029
```

2023-12-15T10:28:17.464353+00:00	multi-level-all-fields-1964-2023-8-nodes-1200-epochs-2	maelstrom-a6-1	JUWELS (JSC)	FAILED
2023-12-15T08:48:38.144250+00:00	multi-level-all-fields-1964-2023-8-nodes-1200-epochs-2	maelstrom-a6-1	JUWELS (JSC)	FAILED

Figure 4: *HPC logs tab in the Run details menu.*

Furthermore, a user can download their entire Run directory on the cluster via the 'Download' option in the menu to the right. Should budget accounts run out after some time, the user will still be able to analyse and learn from past endeavours through this convenient feature. Additionally, it facilitates troubleshooting by providing access to historical data, preserving valuable insights for knowledge retention. Moreover, saved log files serve as educational resources, support performance analysis, and inform strategic decision-making, contributing to a more efficient and informed workflow.

3.2.3. Additional and planned features

In this section, we delve into the ongoing evolution of the graphical user interface (GUI) of the Mantik platform, exploring both current enhancements and future planned features. As technology advances and user expectations evolve, it becomes imperative to continually refine and expand the functionality of the GUI to ensure a seamless and intuitive user experience. By incorporating additional features and refining existing ones, Mantik aims to empower users with greater control, flexibility, reproducibility and efficiency in their machine learning workflow. In this context, we outline the envisioned enhancements that are set to further elevate the user experience and productivity within the Mantik platform:

- By now, only public repositories on the SaaS platforms GitLab, GitHub, and Bitbucket, and public repositories on self-hosted GitLab platforms can be added as a source of ML applications. Private repositories are generally not supported yet. Expanding the capability to add all kinds of self-hosted and private repositories within the Mantik platform presents several advantages for users. The expansion of repository sources within Mantik contributes to a more inclusive, collaborative, and versatile machine learning development environment.

- Currently, the MLflow GUI is accessible separately via our self-hosted MLflow instance. However, in the future, we plan to integrate the MLflow GUI into the Mantik interface by forking the open-source code, thereby unifying the frontends.

- The models tab in the *Project* page is to be implemented after the end of the MAELSTROM project. For each run with a ‘Finished’ status, a user can register the output machine learning model of a run. This model will then show up at the *Models* page. More convenience functions related to models are planned here but it is of lower relevance for the MAELSTROM project.

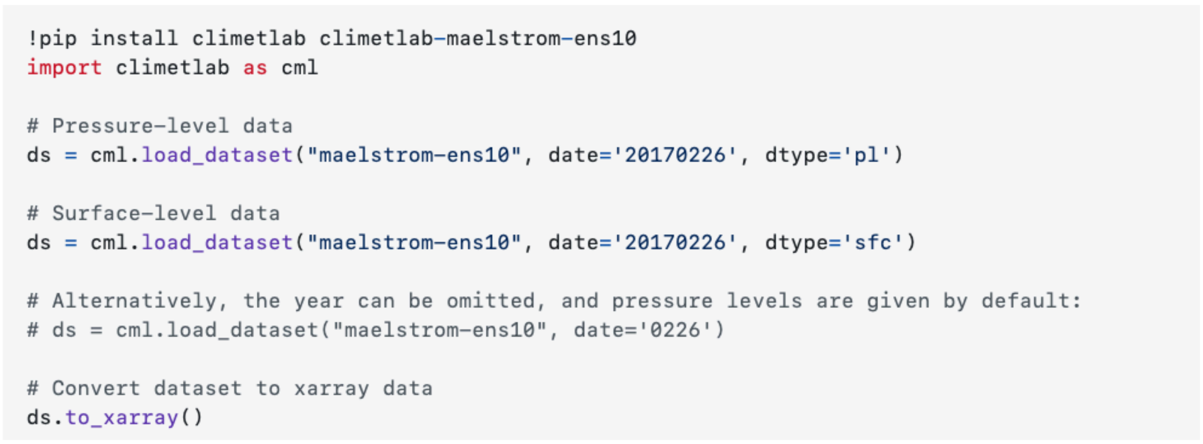
3.4 Data Input/Output Acceleration (Task 2.5)

All applications use the python package CliMetLab[12] to provide their datasets. This simplifies reproducibility of results by external researchers and allows students to investigate state-of-the-art research. For details we refer the user to deliverable 2.4 section 3.3.1.

We observe that I/O is one of the main bottlenecks in the performance of the applications during the benchmarking deliverables. This is because weather and climate applications rely on large datasets and the size of the data is an order of magnitude larger than the size of the machine learning models (see [6]) and the machine learning solutions are affected by I/O issues.

To mitigate this issue, in the first step in application 4, ETH proposes a CliMetLab plugin to download and use of ENS-10 dataset [6]. The plugin allows the downloading of a subset of the dataset for a fixed date and surface level without the need to download (or process) the whole dataset (which is about 3TB) and convert it to *xarray* format.

The following figure shows an example of using the plugin to download the ENS-10 dataset.



```
!pip install climetlab climetlab-maelstrom-ens10
import climetlab as cml

# Pressure-level data
ds = cml.load_dataset("maelstrom-ens10", date='20170226', dtype='pl')

# Surface-level data
ds = cml.load_dataset("maelstrom-ens10", date='20170226', dtype='sfc')

# Alternatively, the year can be omitted, and pressure levels are given by default:
# ds = cml.load_dataset("maelstrom-ens10", date='0226')

# Convert dataset to xarray data
ds.to_xarray()
```

Figure 5: *An example of using CliMetLab plugin for downloading ENS-10 dataset in application 4.*

In addition to the plugin above, application 4 came up with another solution for solving the I/O issue. To this end, in deliverable 1.4, they showed that the dataset could be saved in NumPy format instead of NetCDF format (which is the default format in application 4 for training neural networks). This resulted in I/O acceleration up to 4.9 times for their application as converting NumPy arrays is much faster than NetCDF format for PyTorch codes.

Since D2.4, further work was done on the data loader for application 1. In the application 1 dataset, each file in the archive contains a full 58 hour forecast. In a previous implementation, each file was loaded in its entirety before the next file was loaded. The contents of these files were then shuffled to create randomised samples for the training. However, due to limited RAM on the nodes, only 3 files could be loaded at a time. This puts restrictions on how shuffled the data could be, leading to suboptimal training. We changed the sampling strategy by reading a single hour from each file, allowing us to shuffle across a much larger variety of weather situations. Initially, this created an I/O

[12] https://climetlab.readthedocs.io/en/latest/

bottleneck due to the way we retrieved the data with *xarray*. This implementation caused the entire file to be read each time a single hour was requested. This problem was discovered by investigating the I/O-traces generated by the darshan tool, which we became familiar with through our collaboration with the IO-SEA project in D2.4. The tool made it clear that the amount of data read from disk was much higher than expected, leading us to understand where the problem likely lay.

3.5 Deployment and Infrastructure (Task 2.6)

Researchers benefit greatly from the ability to seamlessly transition between local development environments and development on HPC clusters to create cutting-edge machine learning models.

Since Mantik is designed with an abstract interface, it also allows for a quick adaptation of any other HPC facility. The only requirement is that it hosts a service that exposes a REST API. In the case of UNICORE, no additional effort is required to enable users to execute their research applications on clusters that provide a UNICORE API. Hence, model training on specialised machines of E4, developed in WP3, is feasible as soon as UNICORE is deployed.
In the case of other interface technologies that may be hosted by other research facilities, as soon as that interface has been adopted by Mantik, their users will be able to run their applications from the platform. In the time since Deliverable 2.4 we integrated such an interface for FirecREST, provided by CSCS.

In order to incorporate FirecREST into Mantik, 4cast has collaborated with the team at CSCS that maintains FirecREST. With most required information about the software being publicly available (i.e. software library and API documentation), and additional information that could be gained by communicating with CSCS, the implementation of FirecREST was carried out without requiring any changes of Mantik's software architecture, nor introducing any conflicting changes for existing Mantik projects and applications. The Mantik Compute Backend was extended to allow the submission via FirecREST. Furthermore, an internal (non-user-facing) abstract interface to Mantik Runs was introduced within the Mantik REST API to allow submission of Runs to the Mantik Compute Backend, and retrieve any information (status, metadata, logs, tracked parameters and metrics, etc.) or interact with submitted Runs (cancelling, downloading files from the cluster, re-submitting a Run). All features provided to users interacting with clusters running UNICORE are therefore provided to CSCS cluster users as well.

In Deliverable 2.4, we documented our collaboration with the RED-SEA project and their tools for analysing network interconnect. These tools can be used to analyse potential bottlenecks when running large-scale workloads on many HPC nodes. Since Deliverable 2.4, we have produced further traces with application 1 and uploaded these to their publicly available repo[13]. Specifically, we ran application 1 with 256 MPI tasks providing RED-SEA and the interconnect community with data for large-scale machine learning workloads.

[13] https://gitraap.i3a.info/jesus.escudero/vef-traces-repository/

The researchers of the MAELSTROM project use Deep500 recipes[14] for benchmarking and found their recipe produces the same results on three different infrastructures for the application A4 as seen in Deliverable 2.2. Additional software for performance optimization and portability of solutions is being developed.

[14] see Deliverable 2.3

4 Updated architecture of Mantik

In the development journey of Mantik, the convergence of frontend and backend implementation during the last 12 months of the MAELSTROM project marks a significant milestone. This chapter encapsulates the results of our efforts from the software development perspective, reporting on the structure of the software architecture. Throughout the process, we encountered challenges, made pivotal decisions and leveraged innovative solutions to realise Mantik's vision as a cutting-edge tool for all machine learning research instrumentalizing HPC infrastructure.

The reader will find in this chapter a short representation on the overall interaction of the Mantik components and their deployment on Amazon Web Services (AWS), including an overview on decisions made from the software development perspective.

4.1 Deployment on AWS

AWS is a leading cloud computing platform offering a wide range of scalable and cost-effective services to businesses and developers worldwide. We chose AWS for its reliability, scalability, and extensive suite of services, providing us with the flexibility and infrastructure needed to deploy Mantik securely and efficiently.

Three AWS accounts are used for Mantik. A shared service account together with the Kiste project[15], a development environment and a production environment. Kiste is an associated partner in the development of Mantik and focuses on ML development in the field of Geophysics. Mantik leverages a multi-stack deployment architecture on AWS. This approach affords deployment flexibility, enabling independent updates and modifications to distinct components of the application while avoiding disruptions for other users.

Now we describe in further detail the components implemented in our AWS cloud instance. In the following paragraphs the functions of each component listed in the architecture flowchart seen in Figure 6 will be explained. The reader will follow the processing of a request through the architecture.

Every request can come from a *Mantik Client,* for example tracked parameters through the integration of the Mantik and MLflow Python packages into the application code, or from the web browser directly, like run submissions triggered through the GUI on the Website.
The request enters first through a DNS host and is then forwarded to a *Load Balancer*. It distributes incoming network traffic across multiple services to ensure efficient utilisation and prevent overload on any individual service
This *Load Balancer* component is the central gatekeeper for all further processing of a request. Any request coming in will either go directly to the *MLflow Tracking Server,* to the *Frontend* or will be processed by the *Mantik API.* The API, the *Compute Backend* and the Tracking Server pathways will be outlined in the following paragraphs, while the frontend deserves a subsection of its own since its implementation marks the most significant work provided in the last 12 project months.

[15] https://kiste-project.de

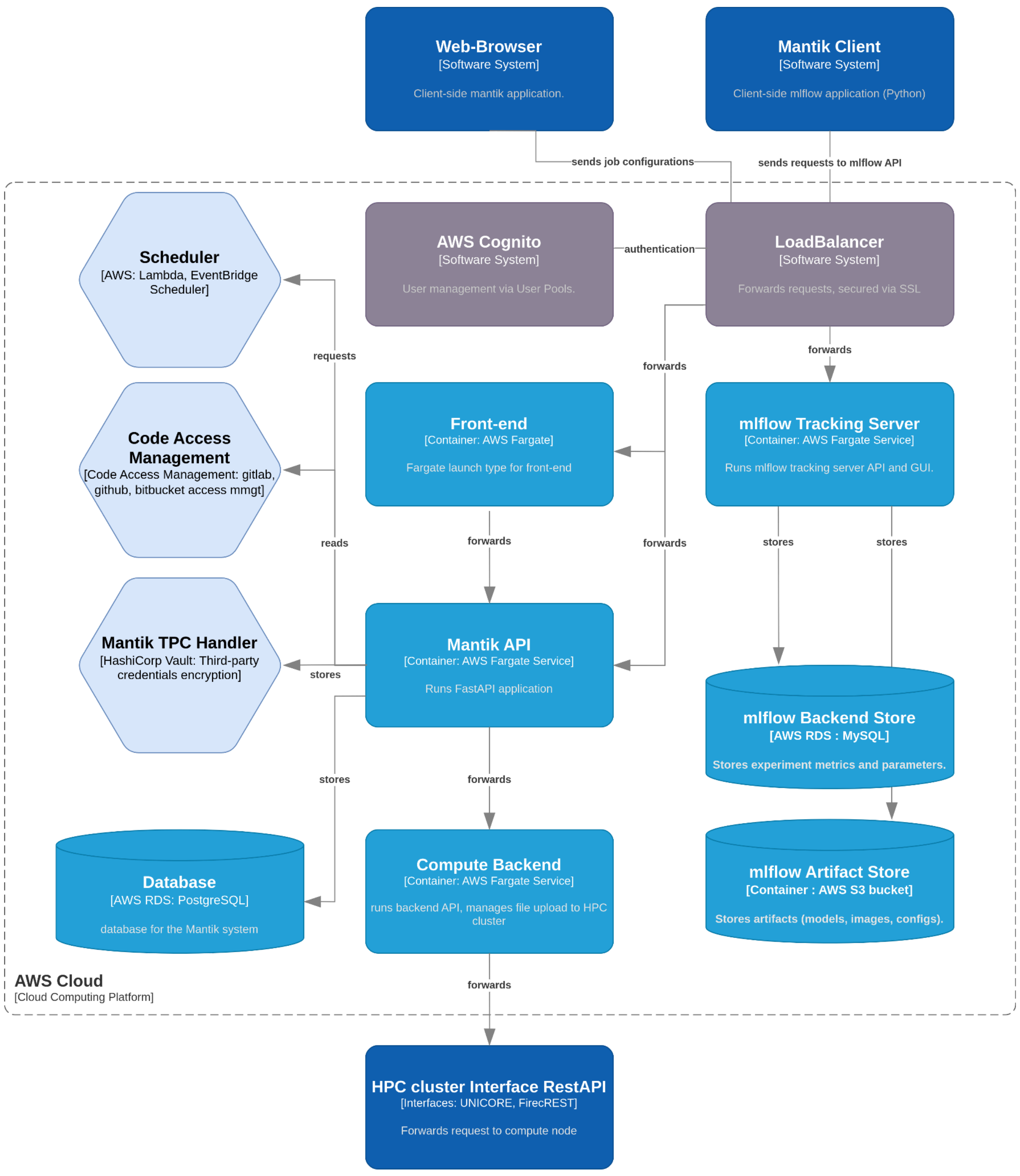

Web-Browser
[Software System]
Client-side mantik application.
Mantik Client
[Software System]
Client-side mlflow application (Python)
sends job configurations
sends requests to mlflow API
Scheduler
[AWS: Lambda, EventBridge Scheduler]
AWS Cognito
[Software System]
User management via User Pools.
authentication
LoadBalancer
[Software System]
Forwards requests, secured via SSL
requests
forwards
forwards
Code Access Management
[Code Access Management: gitlab, github, bitbucket access mmgt]
Front-end
[Container: AWS Fargate]
Fargate launch type for front-end
mlflow Tracking Server
[Container: AWS Fargate Service]
Runs mlflow tracking server API and GUI.
reads
forwards
forwards
stores
stores
Mantik TPC Handler
[HashiCorp Vault: Third-party credentials encryption]
stores
Mantik API
[Container: AWS Fargate Service]
Runs FastAPI application
mlflow Backend Store
[AWS RDS : MySQL]
Stores experiment metrics and parameters.
stores
forwards
Database
[AWS RDS: PostgreSQL]
database for the Mantik system
Compute Backend
[Container: AWS Fargate Service]
runs backend API, manages file upload to HPC cluster
mlflow Artifact Store
[Container : AWS S3 bucket]
Stores artifacts (models, images, configs).
AWS Cloud
[Cloud Computing Platform]
forwards
HPC cluster Interface RestAPI
[Interfaces: UNICORE, FirecREST]
Forwards request to compute node

Figure 6: *Architecture flowchart of Mantik.*

MLflow tracking server

The *MLflow Tracking Server* is a self-hosted instance of the open-source software MLflow, allowing users to use its versioning and tracking features within Mantik. It is set up in an AWS Fargate Service. It exposes its own REST API and a GUI. Through the implemented user authentication components, Mantik offers a secured instance which is not native to MLflow.

The MLflow Tracking Server interacts with two databases: the *MLflow Backend Store* and the *MLflow Artifact Store*. The Backend Store, a MySQL database set up through AWS RDS service, stores experiment metrics and parameters, providing structured data storage for MLflow experiments.
On the other hand, the *Artifact Store*, implemented using the cloud object storage solutions AWS S3 bucket, stores artifacts associated with MLflow experiments, such as models, figures, and configuration files.
These databases together enable the *MLflow Tracking Server* to store and manage the results, parameters, and artifacts generated during machine learning experiments.

Mantik API and Compute Backend

An asynchronous REST API that can handle multiple requests at the same time, was implemented using the Python FastAPI Framework. Various services are connected to the API to facilitate different functionalities.

The *Scheduler* service manages the scheduling of run submissions, allowing for run submissions like benchmarking to be scheduled in advance. Additionally, the *Code Access Management* service enables access to public code from external sources, including GitLab (SaaS and self-hosted), GitHub (SaaS), and Bitbucket (SaaS). The basis of the API is a relational PostgreSQL database, hosted on AWS RDS. This database stores essential information such as user details, project configurations, *Group* and *Organization* memberships, as well as project-specific data like references to code repositories and MLflow experiments. Additionally, run details required for submission to HPC clusters, as well as associated MLflow run references and run schedules, are also stored within this database.
After all information necessary to submit a run to the HPC cluster was gathered, the *Mantik API* forwards the request to the API of the *Compute Backend*. It serves as a bridge to the cluster and initiates the creation of MLflow runs. The *Compute Backend* also facilitates the upload of user-defined files to the run directory on the HPC cluster. The requested resources of the cluster are specified in the backend configuration file (for details see D2.4 p. 16). Furthermore, user-defined environment variables are set in the run environment alongside internal Mantik environment variables by the *Compute Backend* before a job is placed by it in the queue of the job scheduler of the requested HPC site.

Security assessments on HPC credentials

Protecting confidential information provided by the users is a central responsibility of every software development team aiming to provide a publicly available platform handling sensitive data. This applies especially when these information are access credentials to HPC infrastructure.

In the pursuit of secure handling of third-party credentials, our approach revolves around seamlessly authenticating users for various system interactions without necessitating repeated credential input. This entails scenarios such as triggering runs, moving files between systems, and fetching code, all of which should occur without user intervention once credentials are stored. Rather than developing a proprietary solution, we have opted to leverage existing vault technology, particularly exploring options like HashiCorp Vault (Open-Source) and AWS SM (Secrets Manager).
While both offer distinct advantages and considerations, we have chosen HashiCorp Vault due to its versatility and scalability, despite the initial setup complexity and ongoing maintenance requirements. By implementing the *Mantik Third-Party-Credentials Handler* as a service that integrates HashiCorp Vault, we provide users with a robust and secure credential management solution tailored to their needs and environments.

4.2 Frontend implementation

The frontend serves as the primary point of interaction for users, providing a seamless and intuitive experience as they navigate through various features and functionalities. From the initial design concepts to the implementation of robust frontend components, this section offers insights into the technologies and methodologies employed.
We adhered to standards of modern frontend development considering the balance between user-sided requirements, like intuitive structuring and optical appeal, and developer-sided requirements, like maintainability, scalability or coding efficiency. Therefore, Mantik was structured around a component-based architecture, where UI elements are encapsulated into reusable components.

Frameworks/Libraries

Most interactive user interfaces (UI) are based on frameworks and libraries. Libraries like React.js or Vue.js, or frameworks like Angular offer reusable components, state management, and routing capabilities, streamlining the development process. In the following paragraphs we explain our choices:

React.js
By deciding to base the Mantik UI on the React library with a large developer base, we ensure that our open-source platform is easier adaptable and customizable for continuing future developments. Reactjs is currently the most popular open-source JavaScript library for building user interfaces, primarily for web applications. React offers the possibility to create interactive and dynamic UI components and is known for its component-based architecture and declarative approach to building UIs.

Next.js
Next.js is a popular open-source React framework used for building web applications. It enables developers to create server-side rendered React applications, offering features like automatic code splitting, server-side rendering (SSR), and simplified client-side routing. Next.js provides a streamlined development experience, allowing developers to focus on building robust and performant applications without having to worry about complex configuration setups for server-side rendering and routing. Since Next.js supports SSR, the HTML content of pages is generated on the server and sent to the client, making it easily accessible to search engine bots. This ensures that search engines can effectively analyse and index the content of Next.js applications, improving their visibility and ranking in search engine results pages (SERPs).

Typescript
We use React in combination with Typescript, which is a superset of JavaScript. That means that any valid JavaScript code is also valid TypeScript. TypeScript extends JavaScript by adding optional static typing, which allows developers to define types for variables, function parameters, return values, and more. It makes the code more maintainable as the typing already serves as a low level code documentation and eases debugging due to preventing incorrectly written code from being deployed.

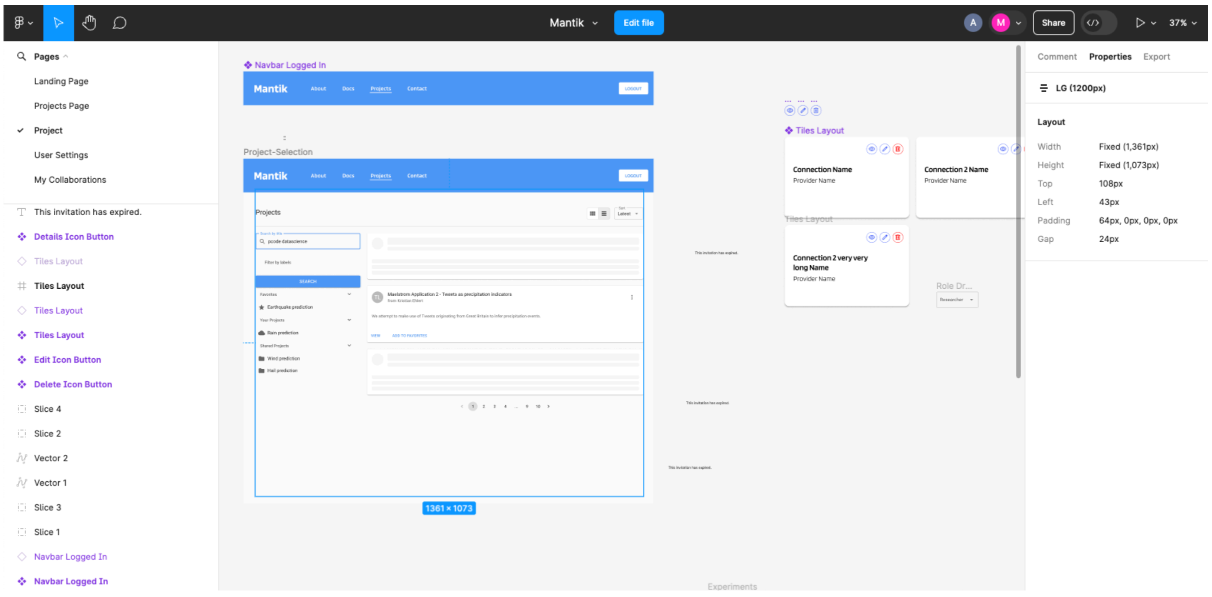
Styling/Design

Designing and styling a modern web UI involves conceptualising the visual identity, wireframing with tools like Figma, and styling with CSS. Component-based frameworks offer pre-designed UI elements for faster development. Responsive design ensures accessibility across devices using techniques like media queries and flexbox/grid layouts. Collaboration between designers and developers is crucial for aligning vision with technical requirements.

MUI
To develop quickly and easily build modern, visually appealing user interfaces, we used MUI, a popular open-source React UI framework based on Google's Material Design guidelines. It provides a set of reusable React components that implement the Material Design principles. It includes many important features, which enable building applications that facilitate testing and barrier free as well as responsive design.

Figma

To allow real-time collaboration when making design decisions, we created and edited first versions of the UI in Figma. Figma is an intuitive cloud-based program crafted to streamline the design process of web-based user interfaces. With its robust suite of tools, Figma offers its users a flexible and minimal effort approach to UI design.



LG (1200px)

Layout

Width	Fixed (1,361px)
Height	Fixed (1,073px)
Top	108px
Left	43px
Padding	64px, 0px, 0px, 0px
Gap	24px

Figure 7: *Designing Mantik in Figma.*

Testing

Testing web user interfaces during development is crucial to ensure the functionality, usability, and reliability of the final product. In this process bugs are identified or potential errors. Various testing techniques, such as unit testing, user-centric testing and integration testing are employed to assess different aspects of the UI's performance and functionality.
Additionally, tools like Jest and React Testing Library provide developers with efficient means to automate testing processes and ensure comprehensive coverage across different browsers and devices.

React Testing Library and Jest

We use the React Testing Library (RTL) with Jest for testing purposes.
The RTL is used to test React components regarding their user-specific functionality, like testing user actions e.g. clicks or typing. We also installed the msw package, which lets us mock data from the OpenAPI specs file, so in our tests we always have access to the real data models.

4.2 Get started with Mantik

As a final subsection, we want to summarise here how a user can access Mantik and briefly represent the entities of the platform. More detail is provided in the prior chapters and Deliverables.

Mantik can be accessed through the webpage[16] as depicted above in Figure 1. Here a user can register an account using a valid email address. After the typical confirmation process, the chosen credentials can be used for the login. After clicking the 'Discover' button, the user is directed to the *Projects* overview page where all publicly accessible projects are visible and can be filtered by a search bar and by labels.
Every ML research is structured as a *Project,* for example every MAELSTROM application has its own *Project.* In order to use Mantik properly, the Mantik Python package[17] and the MLflow package must be integrated into the application code as described in D2.4. Two additional configuration files are needed that tell Mantik which parameters to track and the HPC cluster which resources are to be used, the *MLproject* file and the *Backend configuration* (see MAELSTROM application 2 as an example provided as chapter 4 in D2.4). Naturally, a code repository has to be provided by the researcher and credentials to the cluster.
After everything is set up, the user can schedule a run directly or is given the possibility to schedule multiple runs ahead. The tracked parameters can be found in the *Experiments* tab of the respective *Project* page. Every function of Mantik is thoroughly documented[18].

[16] https://cloud.mantik.ai
[17] available via pip install
[18] https://mantik-ai.gitlab.io/mantik

5 Conclusion

In this report, the final utilisation of workflow tools for MAELSTROM was described. The new architecture of Mantik was shown in greater detail as well as the recent developments on the RBAC and user management system. Furthermore, summaries on the usage of CliMetLab, Deep500 and JUBE were shown.

Reaching the end of the project, Mantik is now a fully operational ML development platform. Researching the available ML platforms, it became apparent that no single tool is able to map the entirety of possibilities that can occur in a ML workflow. Simultaneously, the use of ML approaches is exponentially increasing in the last few years, hence increasing the need for workflow tools massively. While some tools may offer access to HPC infrastructure, little possibility is given to ensure reproducibility and shareability of solutions in many existing toolsets. Collaboration and monitoring of ML models is rarely addressed.
Every new ML tool has to consider its scope, use-cases, targeted clients and range of functions very carefully. Developed as part of a publicly funded research project, Mantik therefore prioritises to facilitate collaboration between users and reproducibility of solutions, aiming for a shallow learning curve, hence easy adaptation. We integrated the open-source versioning and tracking package MLflow, tapping into its expansive user base. This integration allows researchers to submit ML training scripts directly to the JSC and CSCS HPC clusters through a single platform with little needed adjustments. The architecture of Mantik was designed in a way that new clusters can be integrated modularly, reducing expected implementation time for additional clusters significantly. Organising the work content in *Project* entities proved to be intuitive and RBAC will allow a large number of researchers to collaborate effectively. Being able to save the configurations of a training script submission to a cluster for extended periods of time while empowered to change user-selected parameters with minimal effort, enhances the reproducibility of solutions in a self-explanatory fashion.

Code for all applications and software packages is publicly hosted on GitHub. All data used for the ML applications is available via the Python plugin CliMetLab. With this also only configurable parts of the data can be accessed such that HPC access is not required and therefore even available to students. This allows for easy experimenting and reproducing of results.

In conjunction with WP3, benchmarking of hardware is a main part of the Maelstrom project including software to aid this process. To simplify benchmarking on various hardware we used the command line tool JUBE developed by JSC. In addition, considerable effort has been undertaken to optimise I/O bottlenecks encountered during development of the ML applications.

As a research project in weather and climate science MAELSTROM dealt with exceptionally large data sets. This made us an optimal candidate for cooperating with other EuroHPC sister projects that investigate process optimization. A joint investigation together with the I/O SEA project took place during the last 6 months of the project leading to improvements in I/O speed for application 4 as seen in section 3.4.

References

[1] Emmerich, Fabian. “The Maelstrom Protocol - Workflow for the Development of AI on HPC.” Platform for Advanced Scientific Computing (PASC), June 27, 2022, Congress Center Basel, Basel, Switzerland. Talk.

[2] Emmerich, Fabian. “Mantik – A Workflow Tool for the Development of AI on HPC.” High Performance Edge And Cloud computing (HiPEAC), January 17, 2023, Pierre Baudis Convention Centre, Toulouse, France. Talk.

[3] Emmerich, Fabian. “Mantik - A Workflow Tool for the Development of AI on HPC.” EuroHPC Summit 2023, 20-23 March 2023, Gothenburg, Sweden. Poster.

[4] Ehlert, Kristian. “Mantik - A Workflow Tool for the Development of AI on HPC.” SOS26 Conference, March 12, 2024, Cocoa Beach, Florida, USA. Talk.

[5] Tal Ben-Nun, Maciej Besta, Simon Huber, Alexandros Nikolaos Ziogas, Daniel Peter, and Torsten Hoefler. “A modular benchmarking infrastructure for high-performance and reproducible deep learning.” In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2019.

[6] Saleh Ashkboos, Langwen Huang, Nikoli Dryden, Tal Ben-Nun, Peter Dueben, Lukas Gianinazzi, Luca Kummer, and Torsten Hoefler. “ENS-10: A dataset for post-processing ensemble weather forecasts.” In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Document History

Version	Author(s)	Date	Changes
0.1	Oliver Kindler (4cast)	21/03/2024	Initial draft
0.2	Kristian Ehlert (4cast), Fabian Emmerich (4cast)	22/03/2024	Additions and other edits
0.3	Saleh Ashkboos (ETH), Thomas Nipen (MetNorway)	25/03/2024	Additions and other text passages
1.0	Oliver Kindler (4cast)	28/03/2024	Final version

Internal Review History

Internal Reviewers	Date	Comments
Peter Dueben (ECMWF)	28/03/2024	Minor comments and suggestions provided
Saleh Ashkboos (MetNorway)	28/03/2024	Minor comments and suggestions provided
Mattia Paladino (E4)	28/03/2024	Minor comments and suggestions provided

Estimated Effort Contribution per Partner

Partner	Effort
ETH	1 PM
4cast	2 PM
Total	**3 PM**