mantik

A workflow tool for the development of AI on HPC

SOS26 - 12.03.2024 - Kristian Ehlert, Sara Grau

MAELSTROM

www.maelstrom-eurohpc.eu

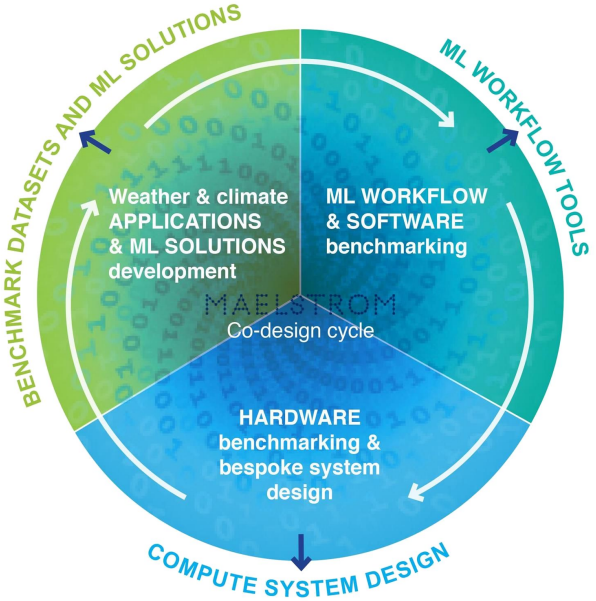
EuroHPC Joint Undertaking

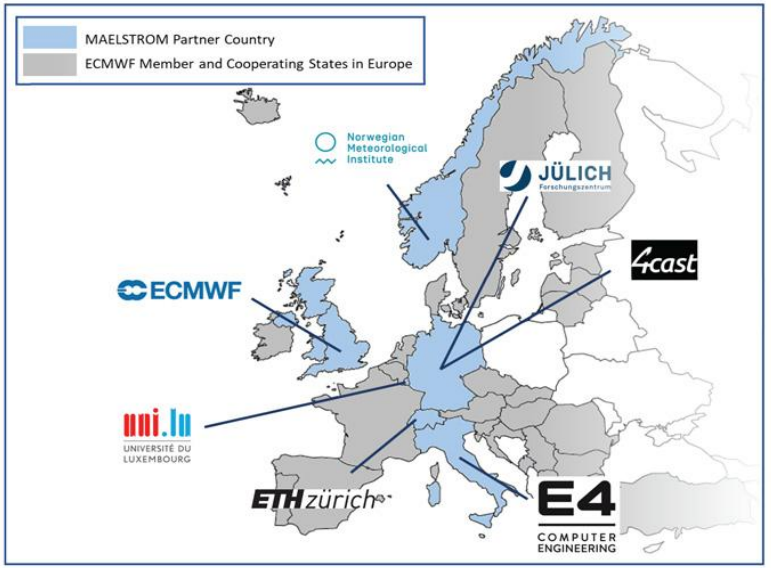
"The MAELSTROM project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955513. The JU receives support from the European Union's Horizon 2020 research and innovation programme and United Kingdom, Germany, Italy, Luxembourg, Switzerland,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

Coordinated by ECMWF

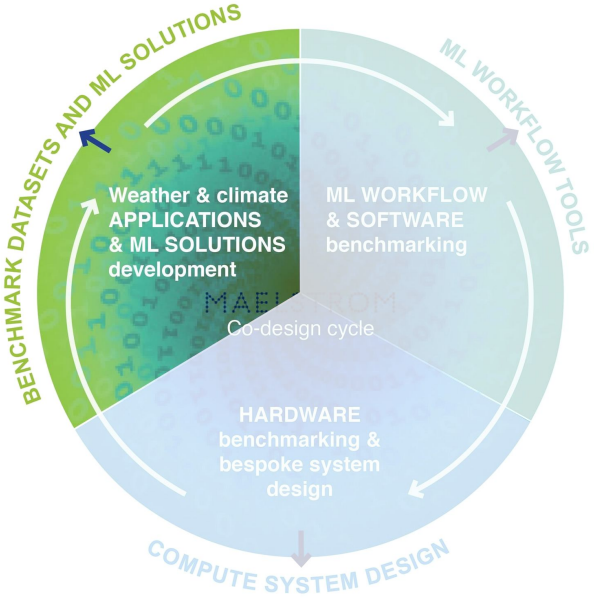
4cast

The Maelstrom project



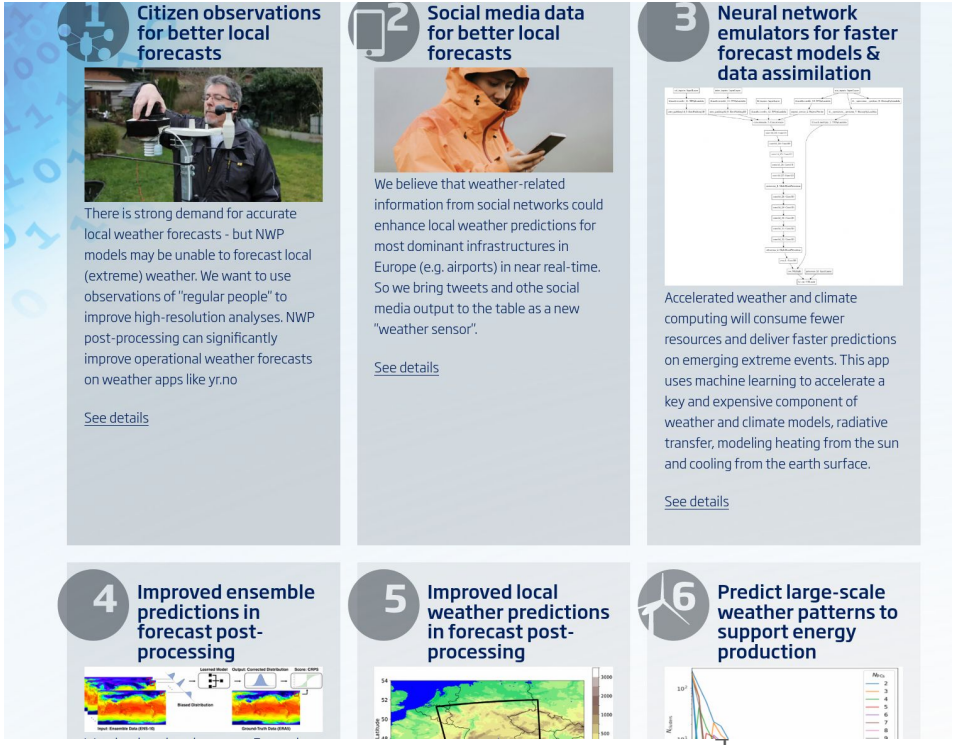


ML applications in Maelstrom



ML applications in Maelstrom

- 6 different ML applications are developed within Maelstrom
  - Open source code and data (~TB)
- General aim: improving weather forecasts



Two examples

- Improve temperature predictions with citizen weather stations
  - ML model augments forecasts with temperature measurements

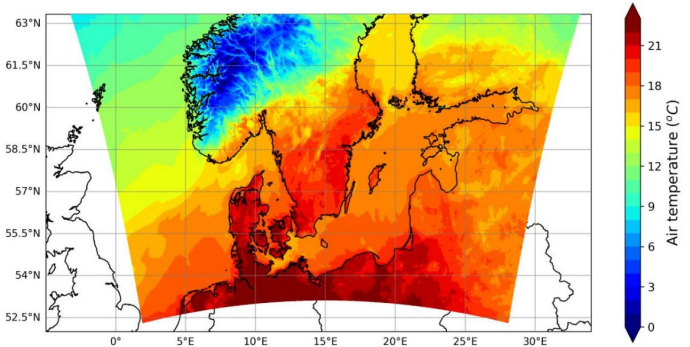
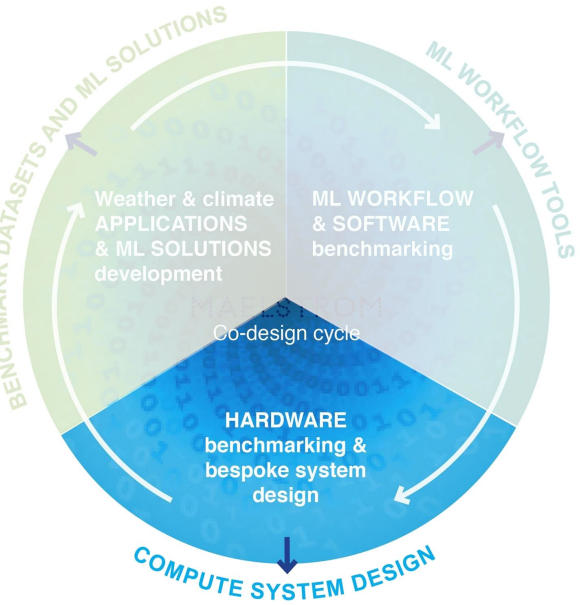


Fig. 4: An example forecast for 13:00Z on September 16, 2023 made using the final U-Net model. The map shows the 50th percentile forecast for the southern half of the domain.

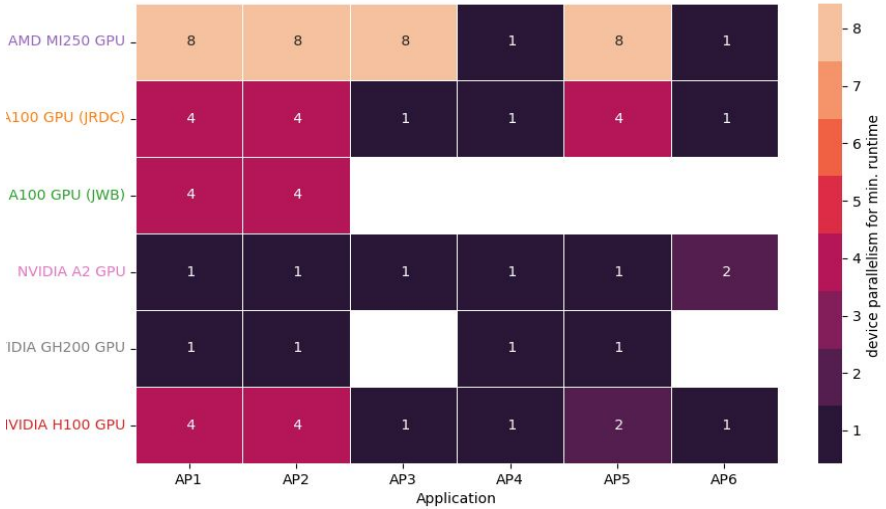
- Use citizen observations on social media to improve precipitation forecasts
  - Fine-tune NLP classifier to decide if "raining" or "not raining" from Tweet

Hardware benchmarking in Maelstrom

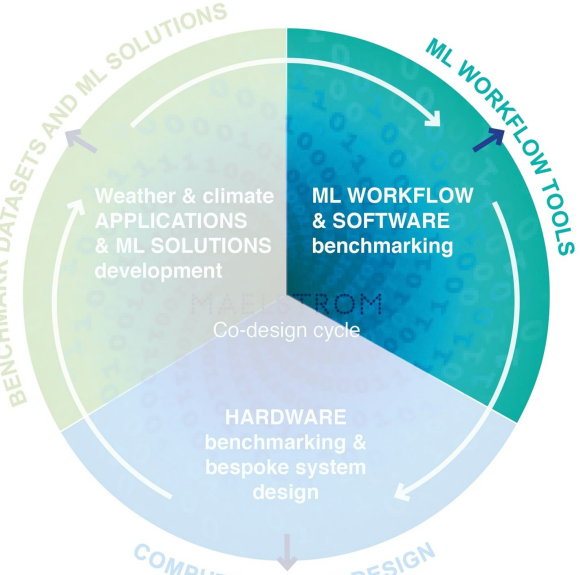


Hardware benchmarking in Maelstrom

- Access required to at least two HPC clusters
- Benchmarking on more than ten different machines with various configurations



ML workflow tool: Mantik



ML tool requirements

Interface to Compute Resources

- Abstract away infrastructure
- Unified access to compute resources (HPC, Cloud)

Reproducibility

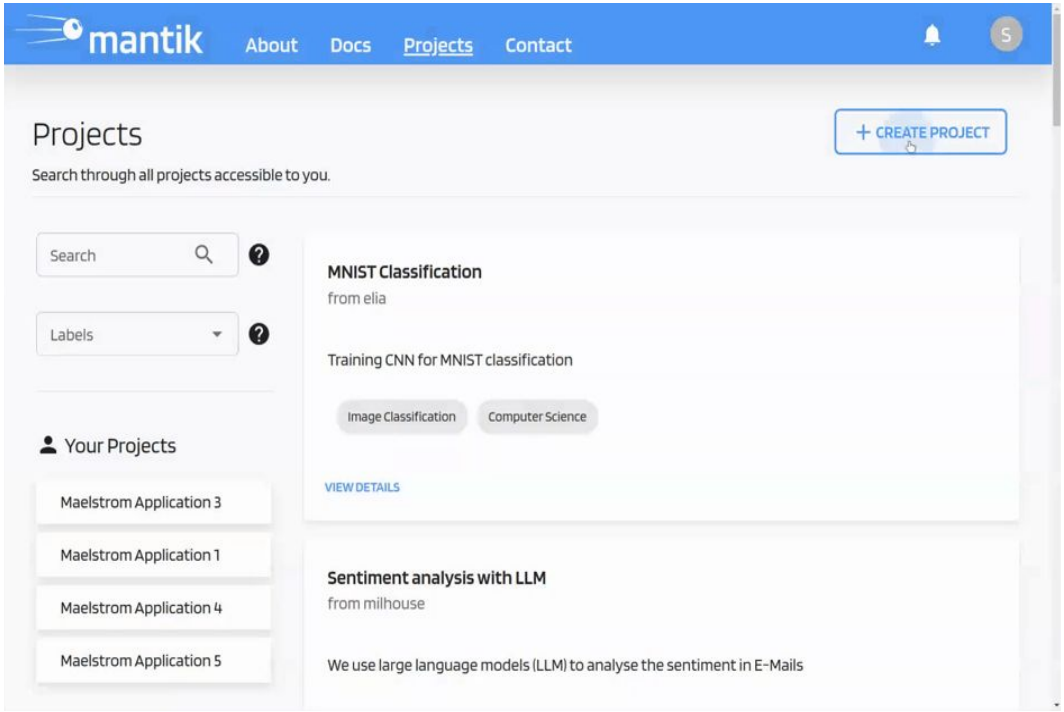
- Recording of input parameters, metrics, models
- Data versioning

Collaboration

- Sharing ML solutions
- Exchange of knowledge
- Improvement of ML solutions

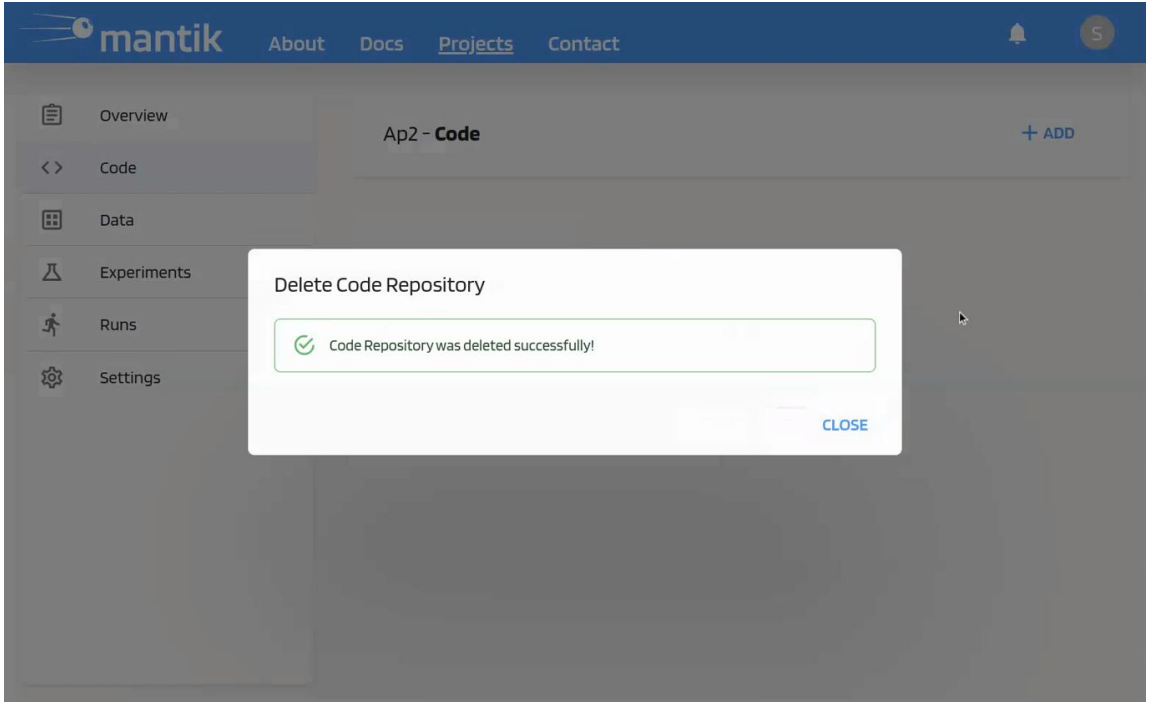
Projects in Mantik

- Basis of working with Mantik
- Holds major assets
  - Code
  - Experiments
  - Runs
- Allow for collaborations on projects



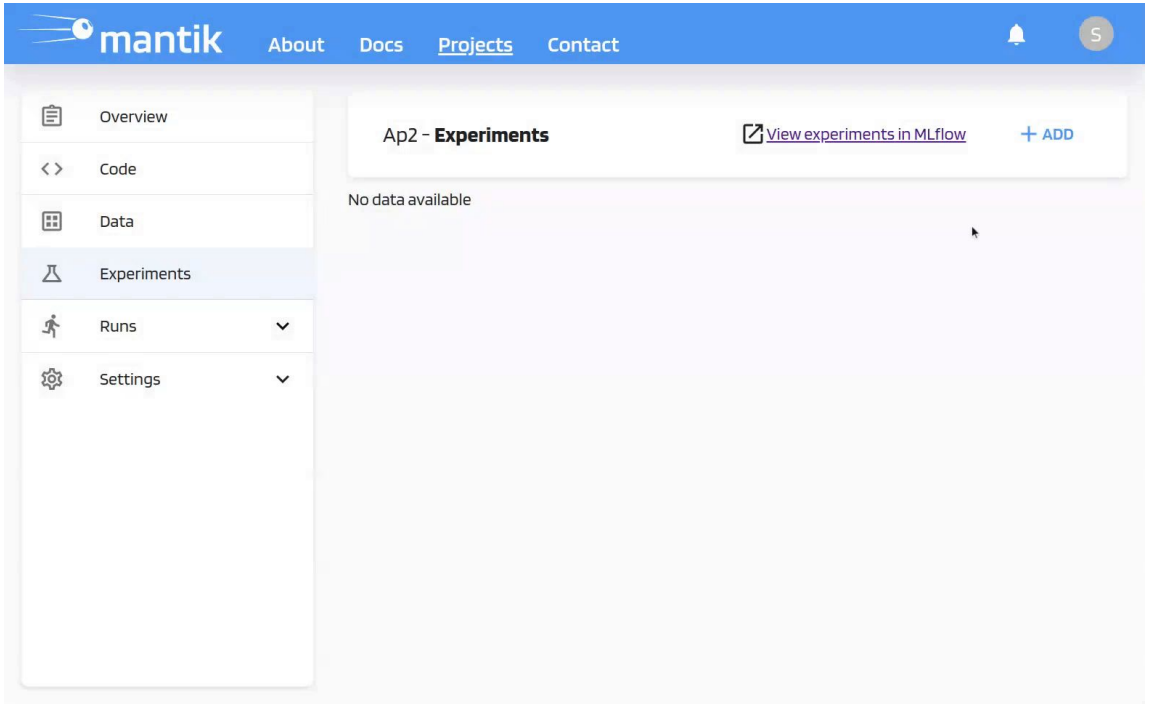
Code in Mantik

- Code linked to git repositories
  - Simplifies versioning
  - Supports major public repository providers



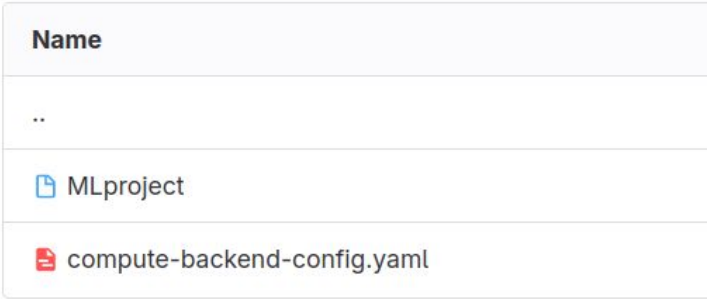
Experiments in Mantik

- Holds tracked parameters and metrics
- Container that attaches metrics/parameters of runs



Configuration of Runs

- Two files define full execution on HPC
  - *MLproject* file defines Run to be executed on HPC
  - Compute backend file configures HPC environment



Configuration of execution command

- *MLproject* file defines Run to be executed on HPC
- Entry points define various tasks to be executed as Run

```
entry points:
        train:
```

Configuration of execution command

- *MLproject* file defines Run to be executed on HPC
- Entry points define various tasks to be executed as Run
- Modifiable parameters may have default values

```
entry points:
    train:
        parameters:
            learning_rate:
                type: float
                default: 3e-5
                …
            validation_size:
                …
            …
```

Configuration of execution command

- *MLproject* file defines Run to be executed on HPC
- Entry points define various tasks to be executed as Run
- Modifiable parameters may have default values
- Command defines execution

```
entry points:
        train:
                parameters:
                        learning_rate:
                                type: float
                                default: 3e-5
                                …
                        validation_size:
                                …
                        …

                command: >
                        python train.py
                                -- validation_size (validation_size)
                                -- …
```

Configuration of execution command

- *MLproject* file defines Run to be executed on HPC
- Entry points define various tasks to be executed as Run
- Modifiable parameters may have default values
- Command defines execution
- Various tasks can be defined in the same file

```
entry points:
    train:
        parameters:
            learning_rate:
                type: float
                default: 3e-5
                …
            validation_size:
                …
            …

        command: >
            python train.py
                -- validation_size (validation_size)
                -- …

    split-dataset:
        parameters: …
        command: >
            python build.py
                -- ..
    benchmark: …
```

Configuration of HPC environment

- Compute backend file configures runtime environment
- Specify HPC access via API

```
Firecrest:
  ApiUrl: https://firecrest.cscs.ch
  TokenUrl: https://auth.cscs.ch/auth/realms/firecrest-clients/protocol/openid-connect/token
  Machine: daint
```

Configuration of HPC environment

- Compute backend file configures runtime environment
- Specify HPC access via API
- Define runtime environment on compute/login node

```
Firecrest:
  ApiUrl: https://firecrest.cscs.ch
  TokenUrl: https://auth.cscs.ch/auth/realms/firecrest-clients/protocol/openid-connect/token
  Machine: daint

Environment:
  PreRunCommand:
        Command: |
        module load Stages/2023 GCCcore/.11.3.0 Python/…
        source /p/scratch/deepacf/maelstrom/…/bin/activate
        ExecuteOnLoginNode: false
  Variables:
        GIT_PYTHON_REFRESH: quiet
```

Configuration of HPC environment

- Compute backend file configures runtime environment
- Specify HPC access via API
- Define runtime environment on compute/login node
- Specify job resources

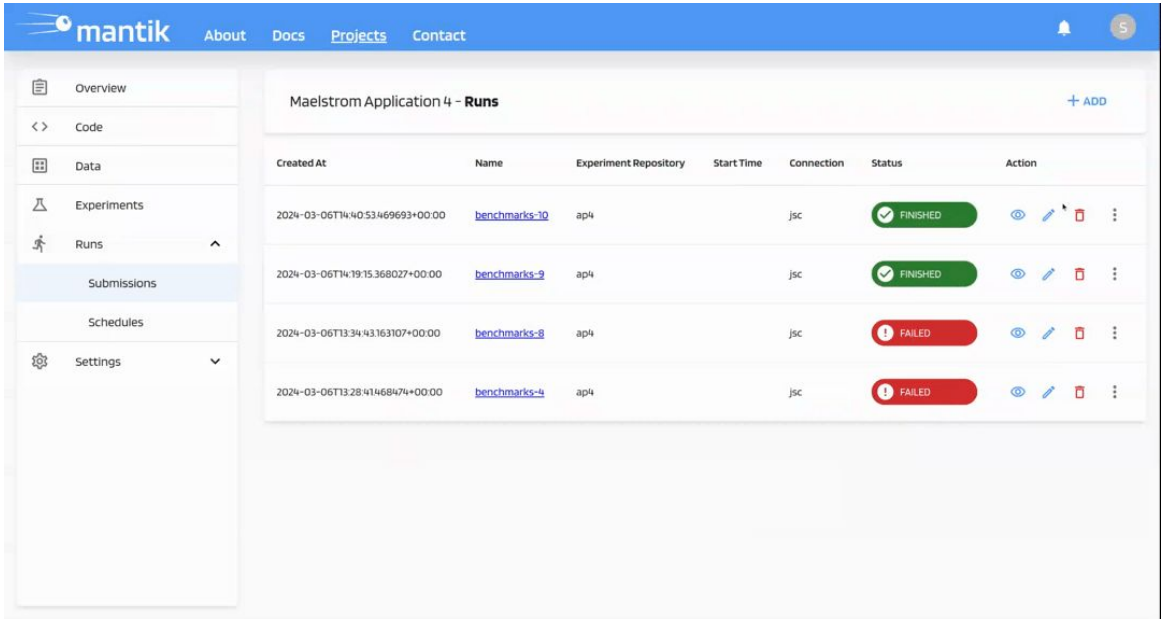
```
Firecrest:
  ApiUrl: https://firecrest.cscs.ch
  TokenUrl: https://auth.cscs.ch/auth/realms/firecrest-clients/protocol/openid-connect/token
  Machine: daint

Environment:
  PreRunCommand:
          Command: |
          module load Stages/2023 GCCcore/.11.3.0 Python/…
          source /p/scratch/deepacf/maelstrom/…/bin/activate
          ExecuteOnLoginNode: false
  Variables:
          GIT_PYTHON_REFRESH: quiet

Resources:
  Queue: normal
  Nodes: 1
  NodeConstraints: gpu
```

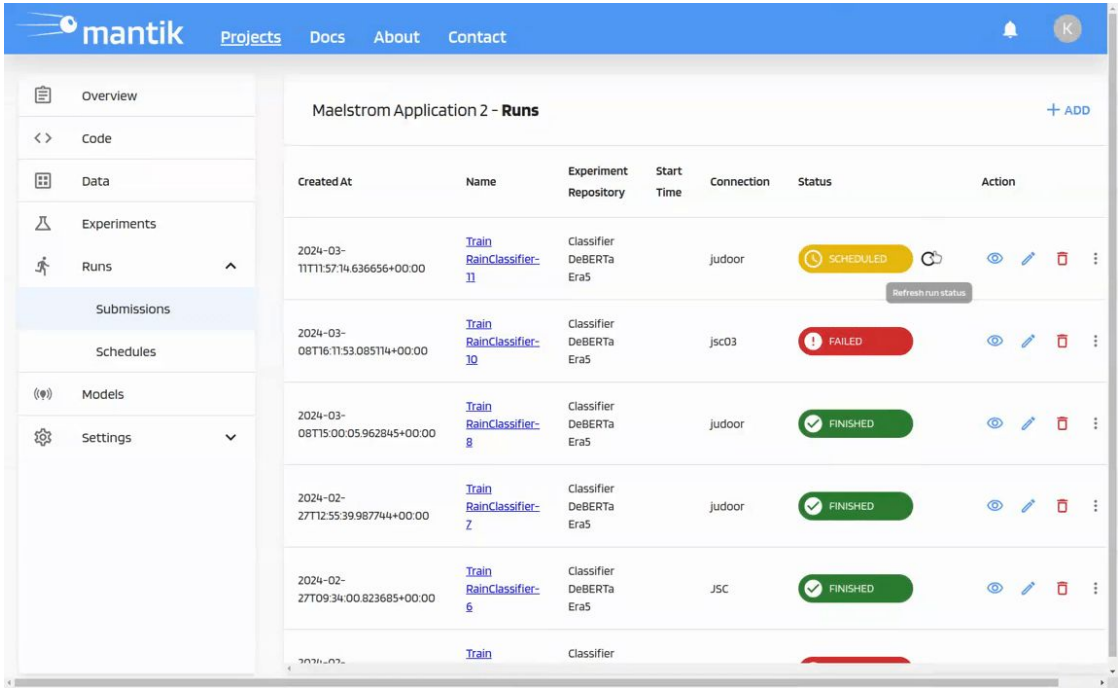
Submitting a Run

- Execution of Code on HPC
- Modify parameters on-the-fly
- Re-run functionality



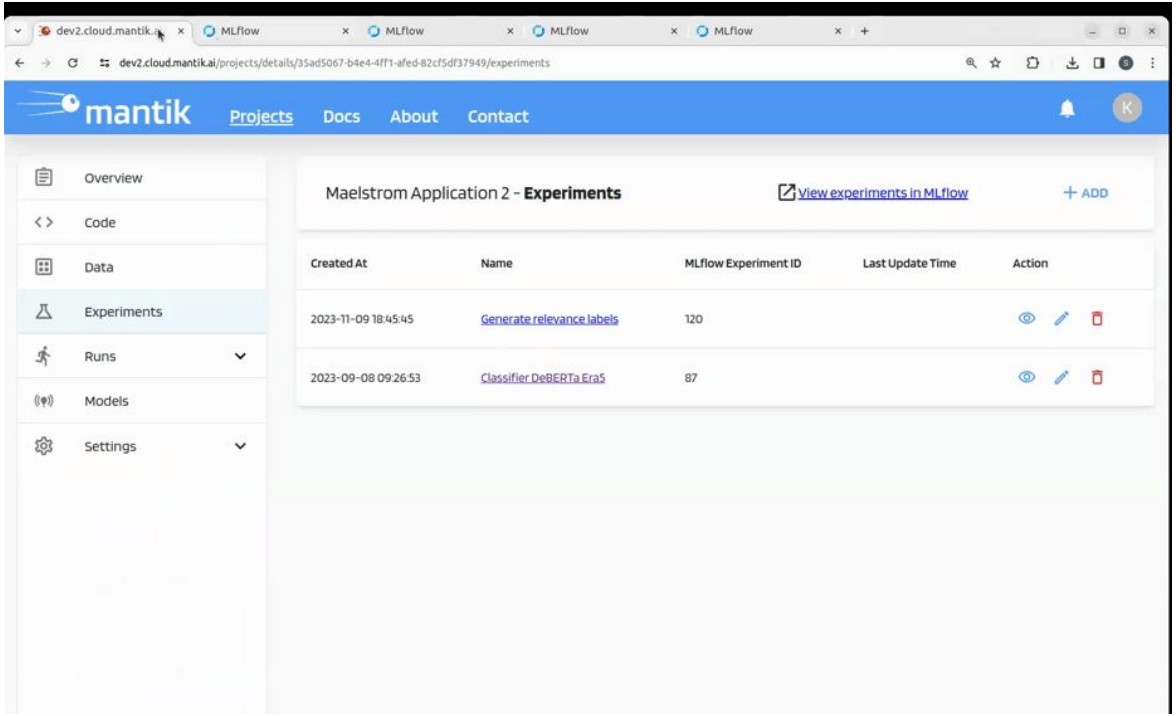
Fetching Status and Logs of a Run

- View status of job
- Check HPC and application logs
- Download run artifacts to local machine



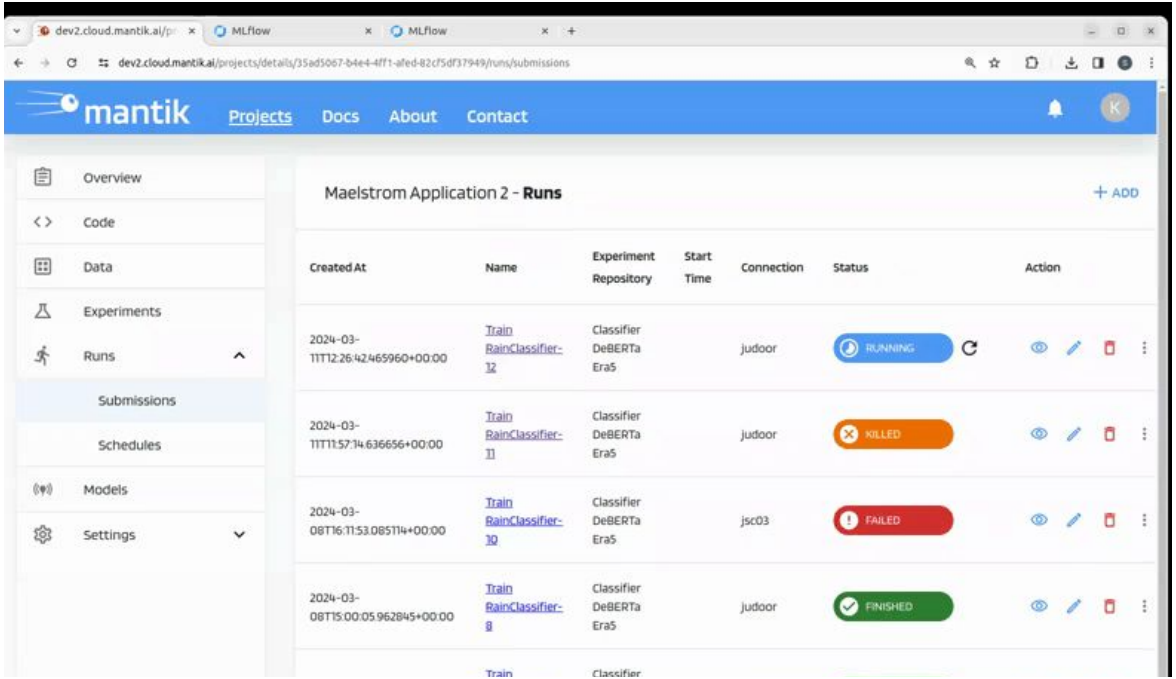
Tracking of results

- Real-life tracking of metrics, e.g. loss, accuracy
- Interactively compare model performance



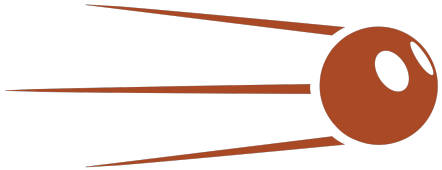
Tracking of results

- Real-life tracking of metrics, e.g. loss, accuracy
- Interactively compare model performance



Conclusions

- Platform built as GUI for ease of use
  - General **access** and interaction with multiple **HPC** clusters
  - Allows developers to code applications **locally** on their laptop
  - Built in **reproducibility** of Runs
  - Real-life **tracking** of results via same interface
  - Enables **collaborative** development of projects
- Built on top of feature rich libraries
  - HPC APIs: FirecREST, UNICORE
  - Tracking: MLflow
- (Current) Limitations
  - Data not as flexible
  - Manual runtime setup required
- Planned features
  - Model deployment for inference with “one click”



mantik

https://cloud.mantik.ai/

Contacts:
kristian.ehlert@4-cast.de
markus.abel@ambrosys.de

MAELSTROM

www.maelstrom-eurohpc.eu

"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".

EuroHPC Joint Undertaking

Coordinated by ECMWF

4cast

Backup

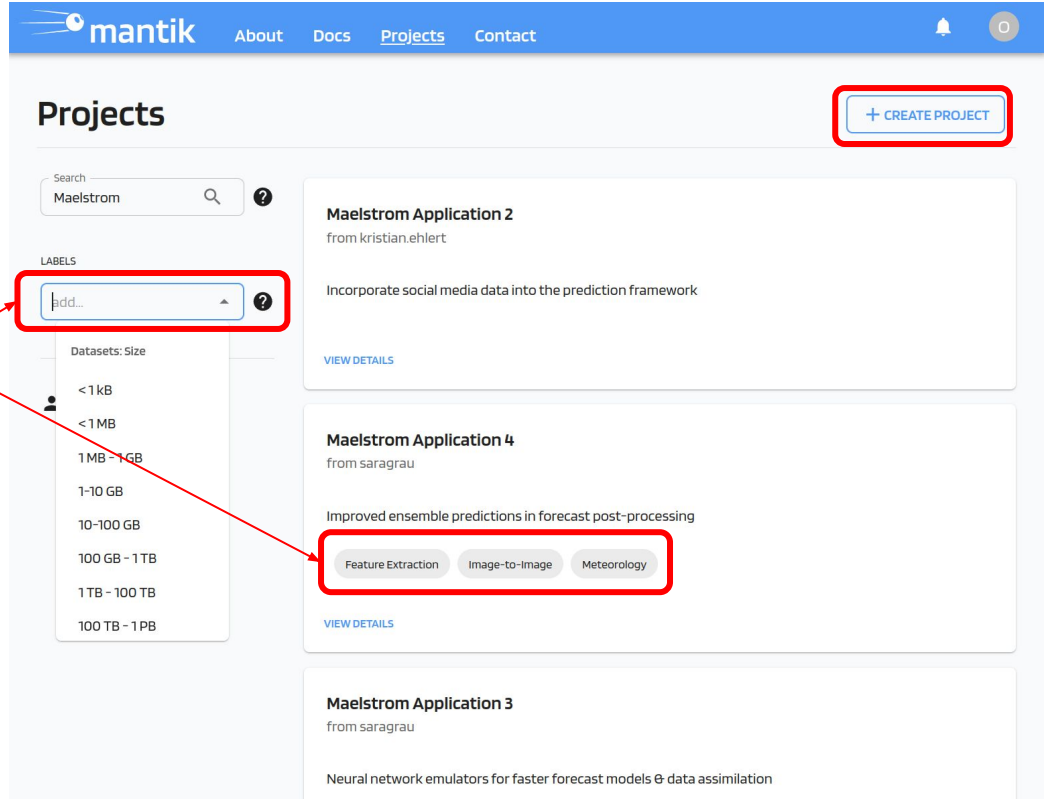
Projects Page

Search *Projects* by *Labels*

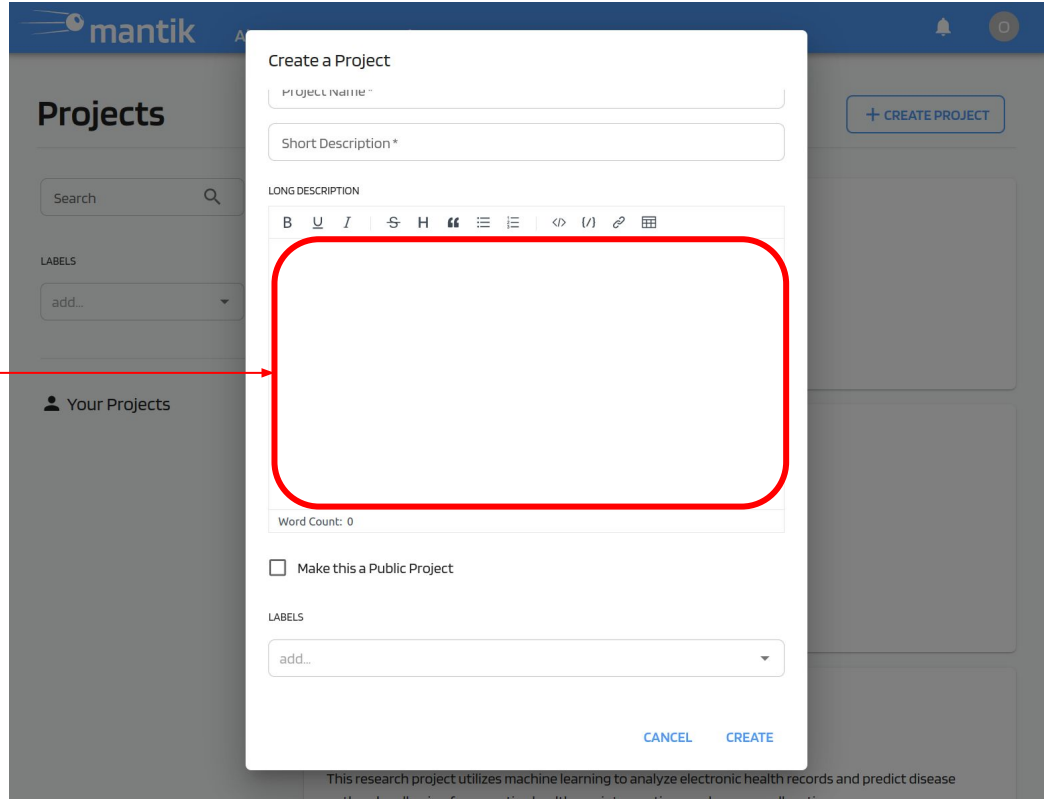
Scopes of Labels are predefined

Labels are grouped in different classes

New Labels can be suggested via a Service Desk



Projects Page



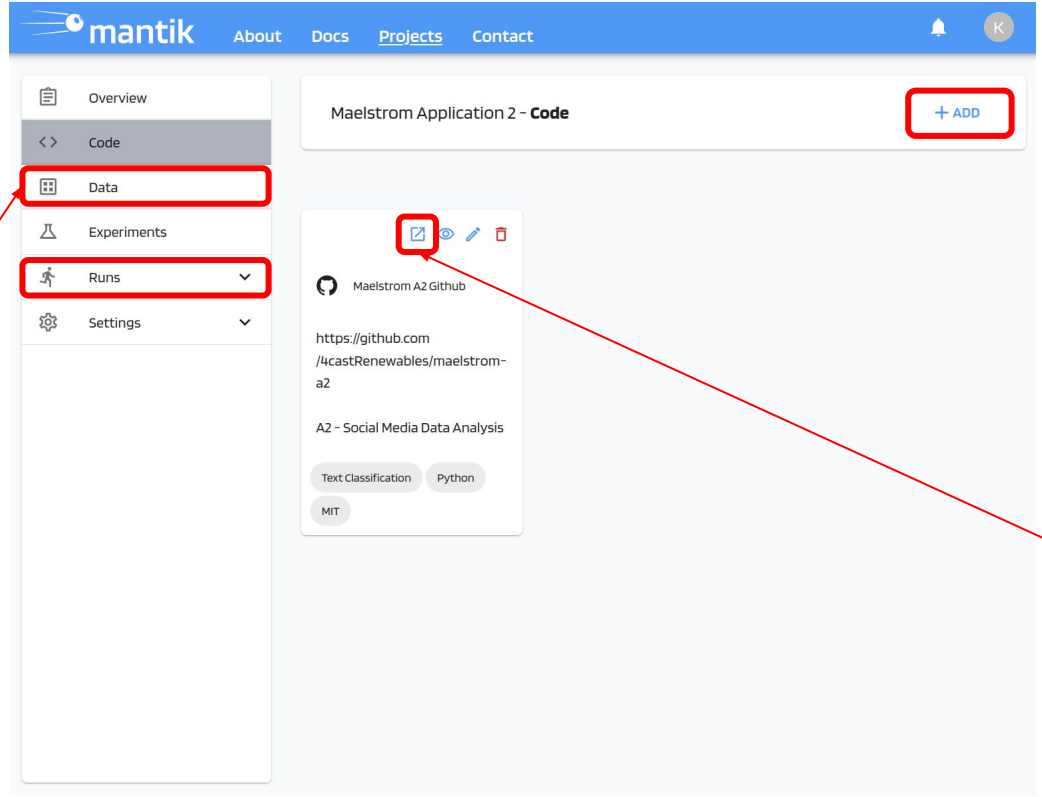
Description will show up in the *Overview* of the *Project*

Supports Markdown format

Project setup

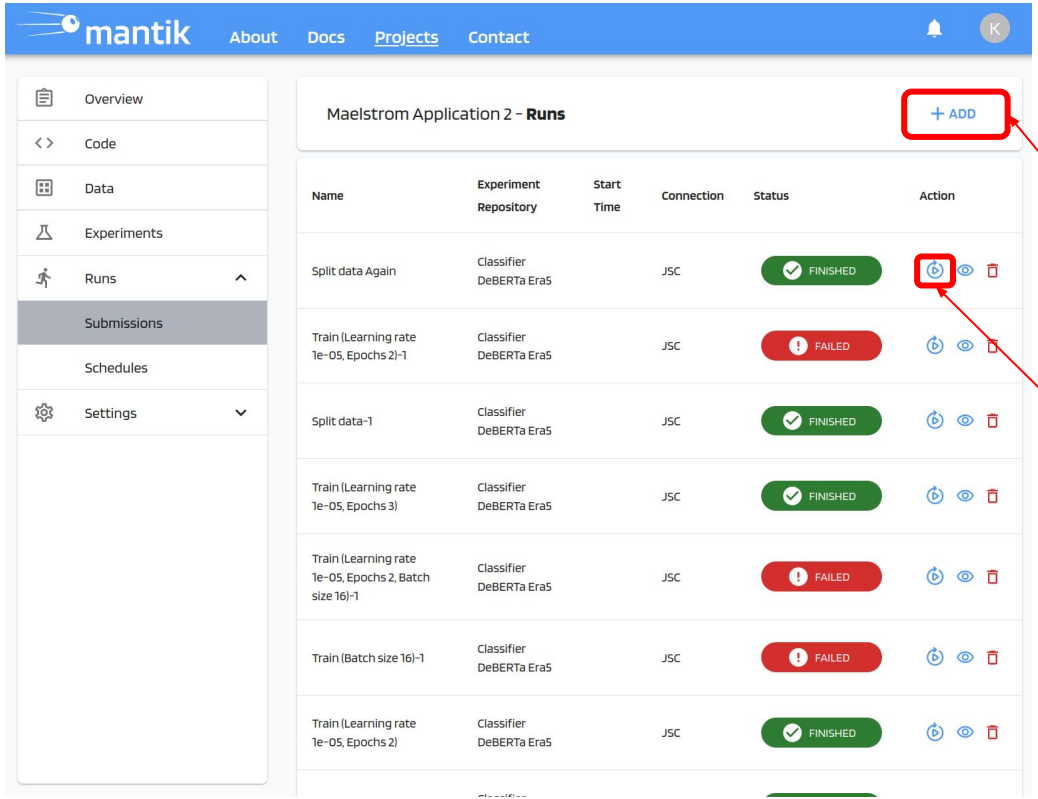
Data management Tab

Tracks history when you use a Mantik CLI command to move data from your machine to HPC



Link to your Git code repository

Submitting a *Run*



Every Submission of a compute job to a HPC cluster is called a *Run*

Runs are grouped in *Experiments (later more)*

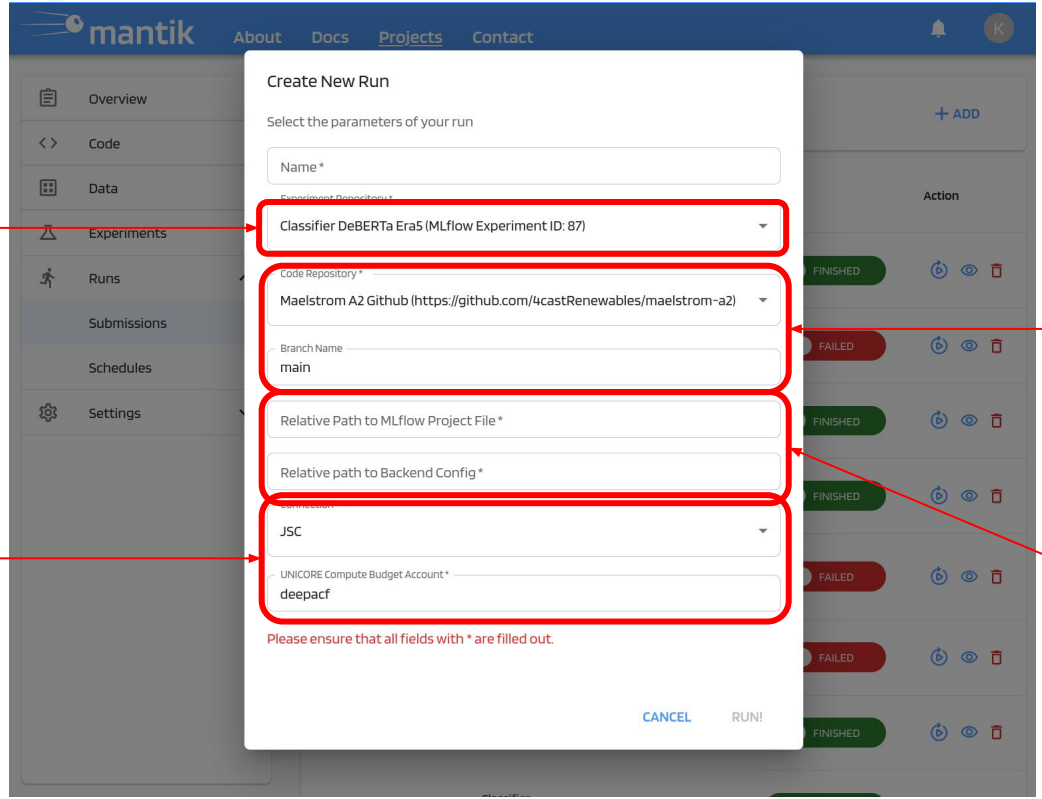
Re-Run Button

Change parameters and repeat an earlier *Run*

Submitting a *Run*

Every *Run* is part of a mlflow Experiment

Choose *Connection* of the cluster and the respective budget account

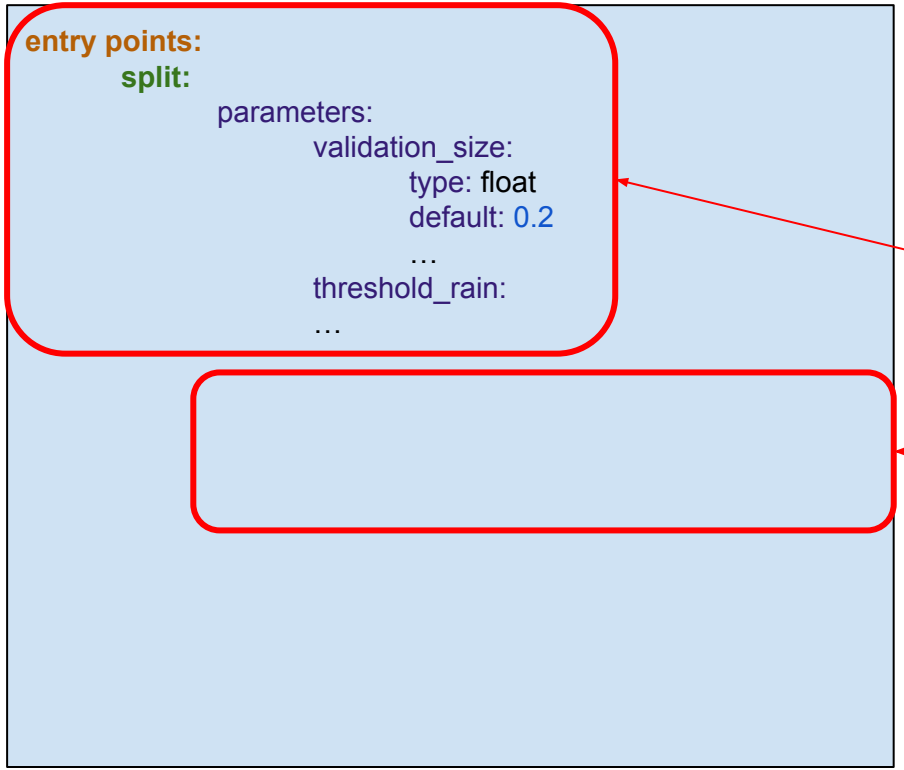


Mantik needs to find the code for submission

Most important:

Path to the **2 necessary configuration files** in the Code repository

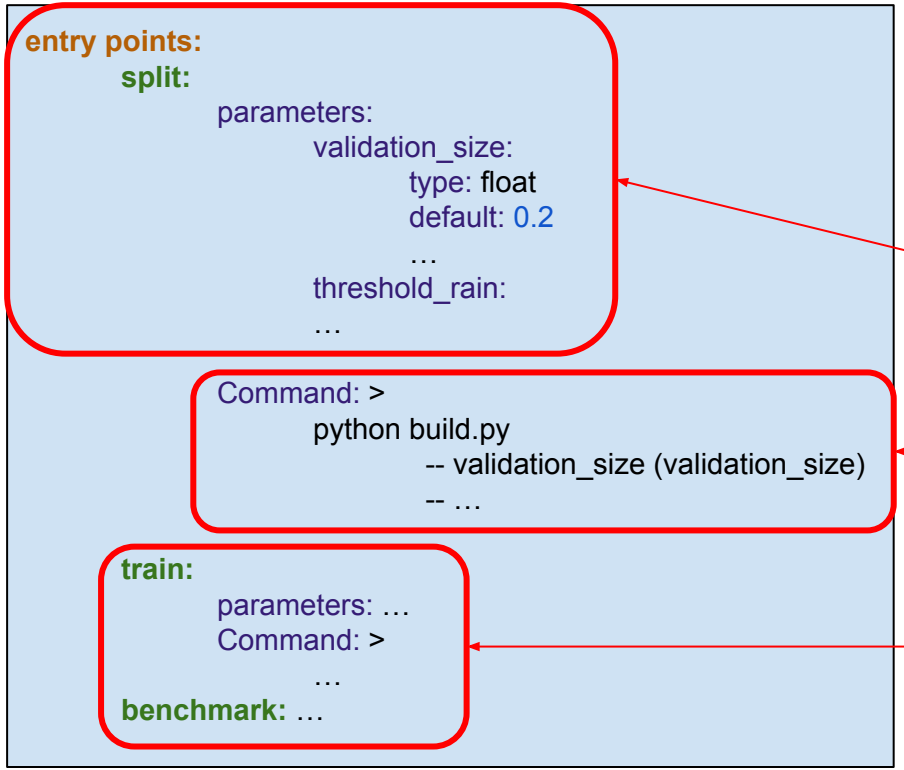
Configuration File I - MLproject



Entry points specify a pre-defined scenario through parameters that will handed over to the respective script

Actual python command that executes the script

Configuration File I - MLproject



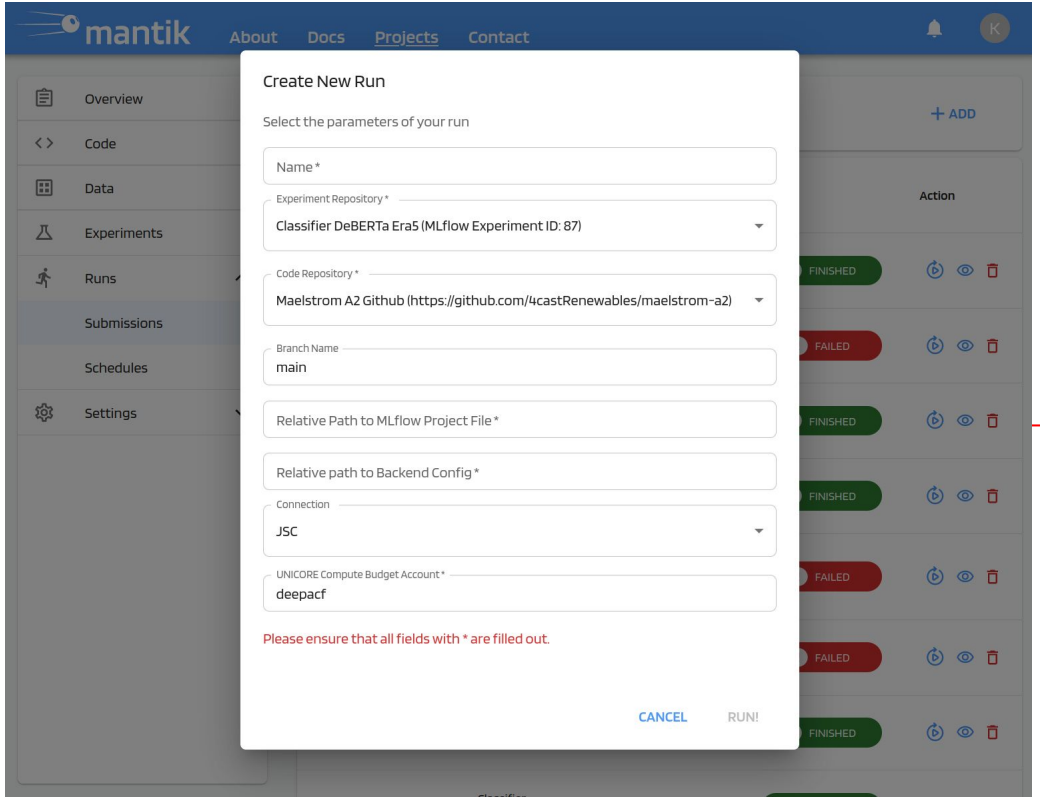
⚠ The file has to be named '**MLproject**' ⚠

Entry points specify a pre-defined scenario through parameters that will handed over to the respective script

Actual python command that executes the script

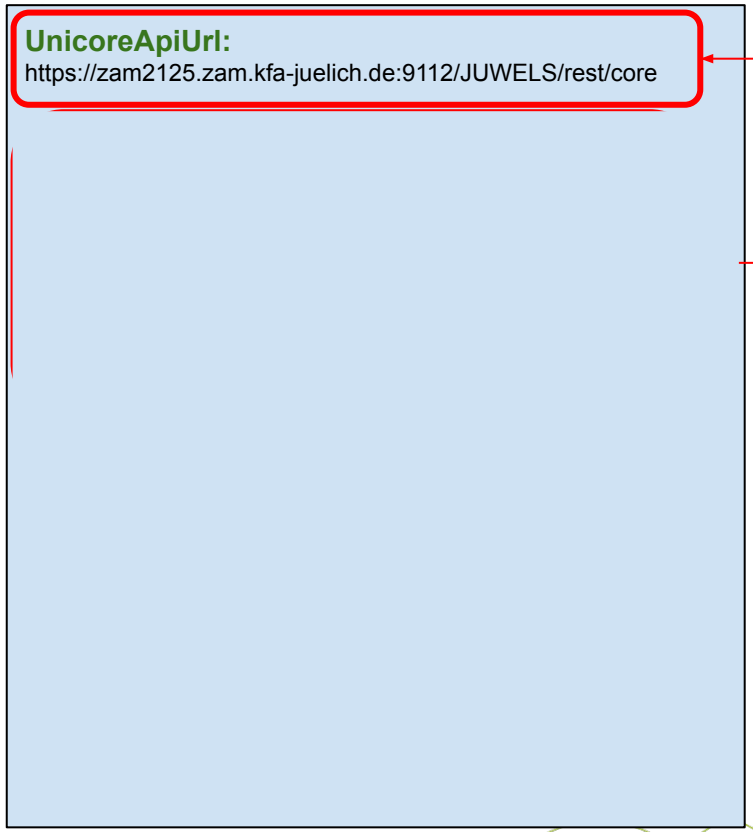
Additional entry points addressing other scripts (for example to Benchmark a *Run*)

Updated a *Run* form



Mantik identifies the parameters specified in the MLproject config file and updates the form accordingly

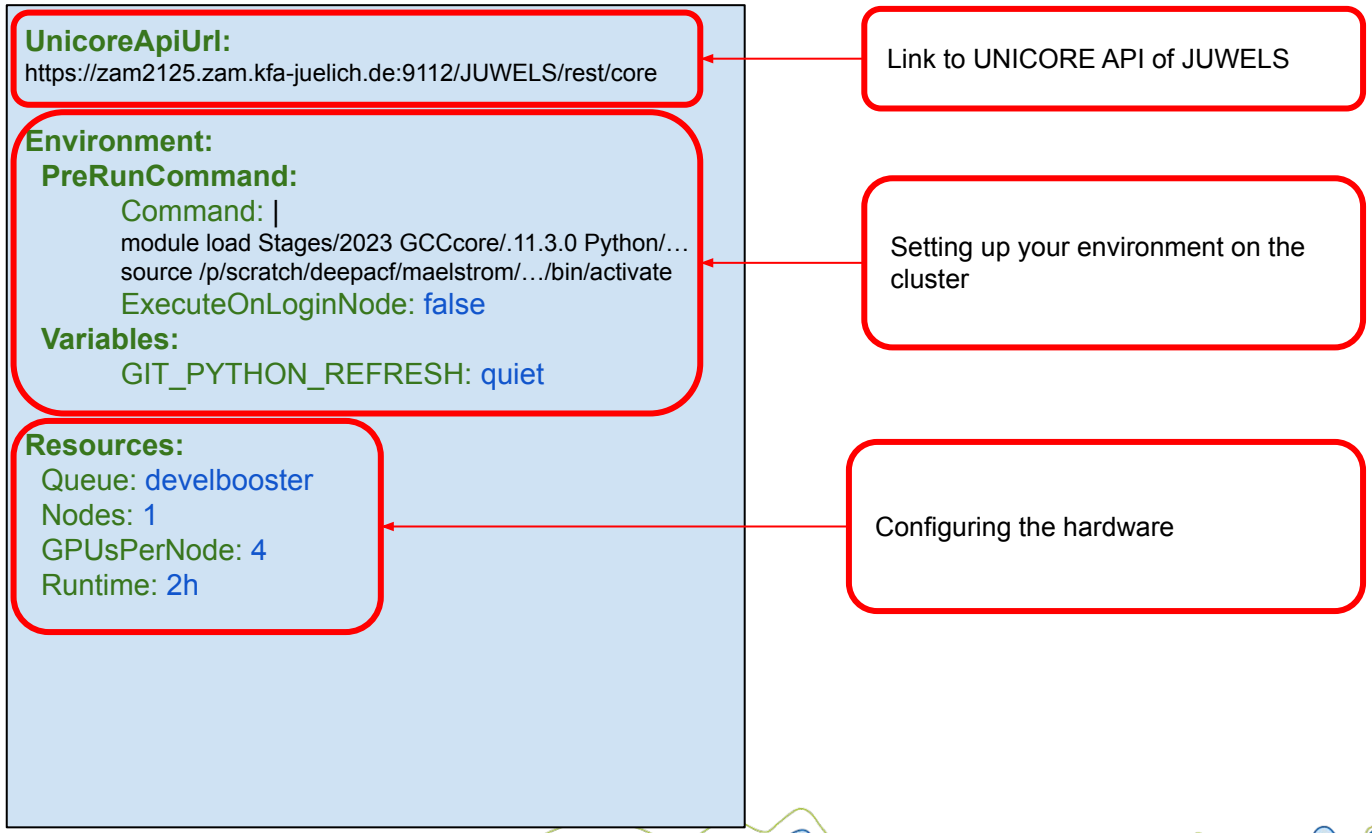
Configuration File II - Backend



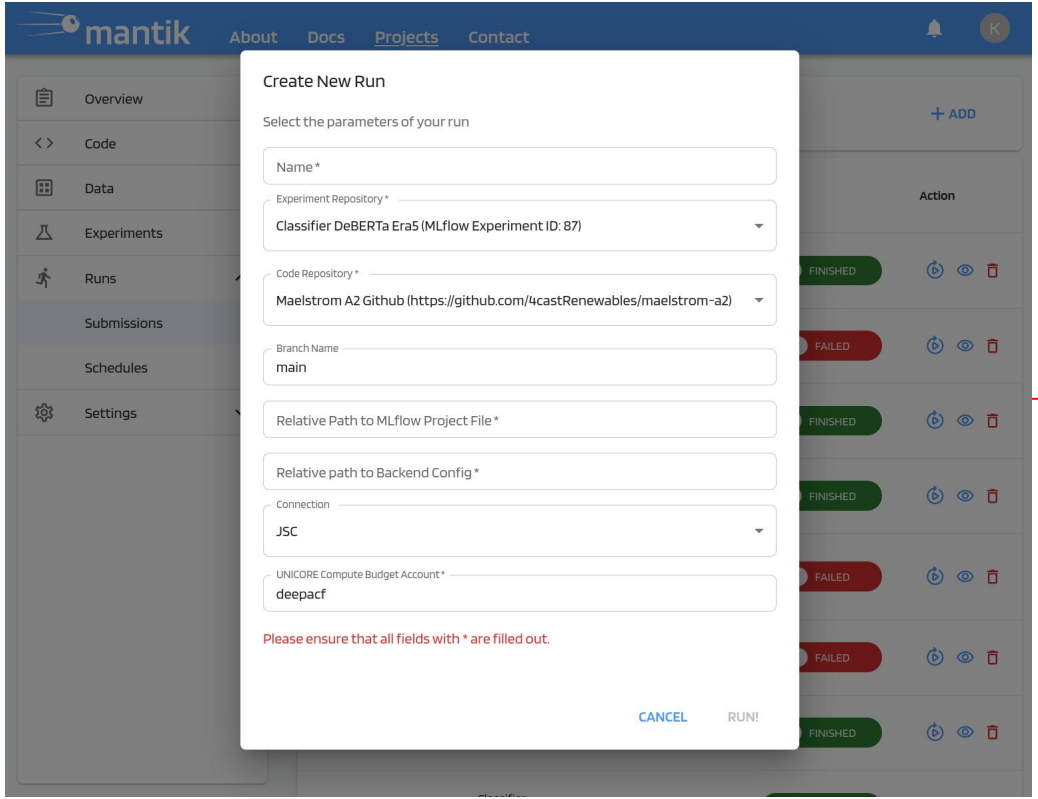
Link to UNICORE API of JUWELS

Setting up your environment on the cluster

Configuration File II - Backend



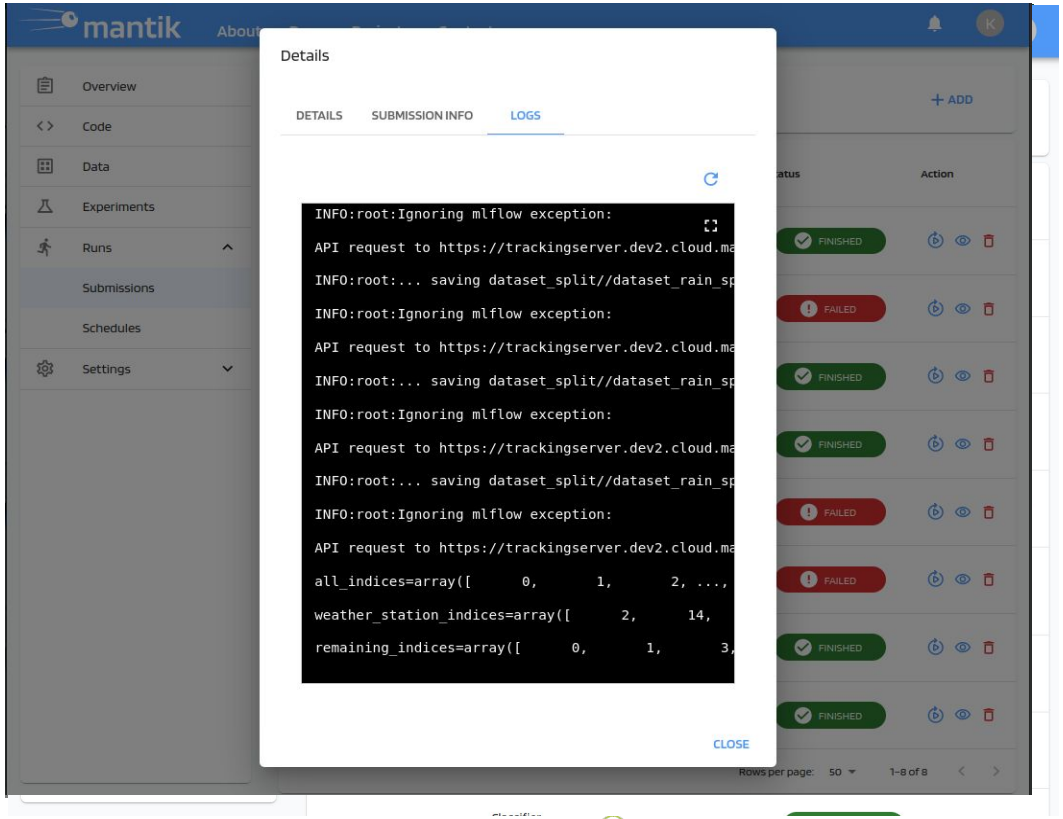
Updated a *Run* form



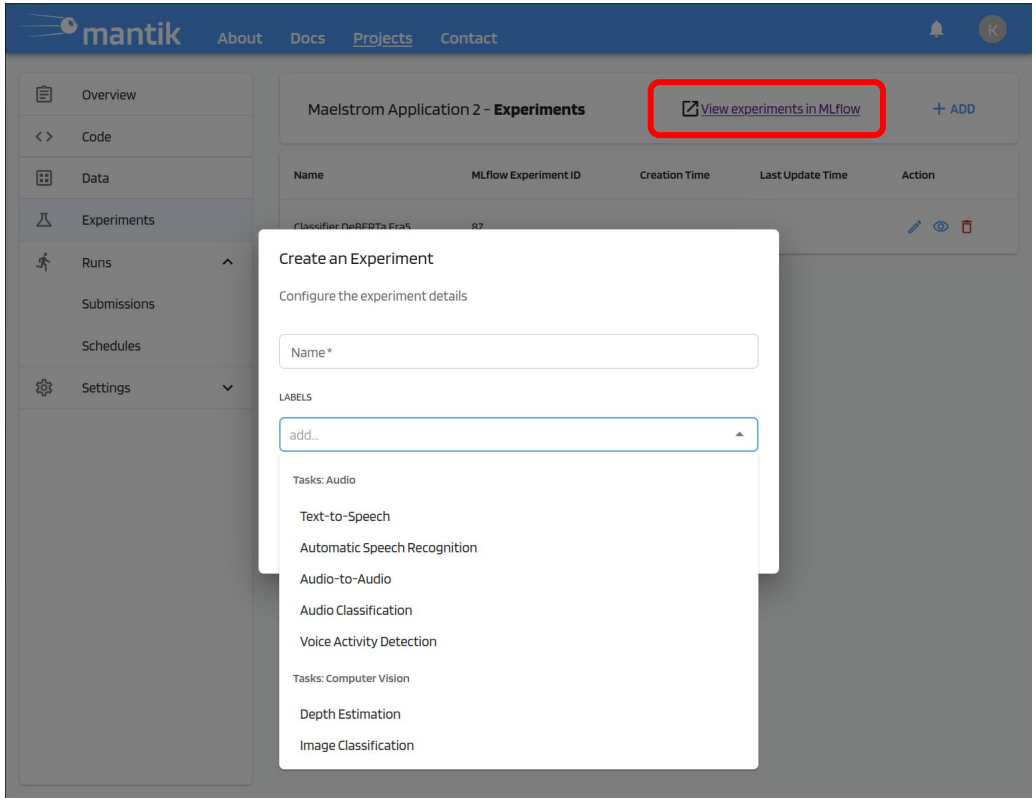
Mantik adds content of backend config to the *Run* form

Enables adjustments to the HPC configuration on the fly

Run Details & Creating *Run Schedules*



Embedding mlflow in Mantik



Embedding mlflow in Mantik

- **Tracking of experiments including**
  - Input parameters
  - Metrics
  - Artifacts (e.g. Figures)
  - Models
- **in real-time on cloud.mantik.ai/mlflow/**
- **Visualization in MLflow GUI**
- **Model versioning**
- **Inference with versioned models**