

AI Intro Basics #5

MLflow

an open-source platform for managing
the machine learning lifecycle

Tobias Göcke, Helen Theissen, Marek Jacob



E-AI Basic Tutorials

Tutorial E-AI Basics 4: January Wednesday 22, 2025, 11-12 CET

"MLOps" - Machine Learning Operations

4.1 Overview (20') [RP]

4.2 MLOps in relation to traditional Weather forecasting (20') [MJ]

4.3 Road to MLOps (20') [DN]

5.1

Tutorial E-AI Basics 5: February Wednesday 19, 2025, 11-12 CET

MLflow - an open-source platform for managing the machine learning lifecycle

5.1 Overview - User perspective (20') [TG]

5.2 Logging to MLflow as a ML software developer (20') [HT]

5.3 Running MLflow server as a user and as a service (20') [MJ]

Tutorial E-AI Basics 6: February Wednesday 26, 2025, 11-12 CET

CI/CD - Continuous Integration and Continuous Deployment of ML codes

6.1 Overview – What can CI/CD do for you? (20') [MJ]

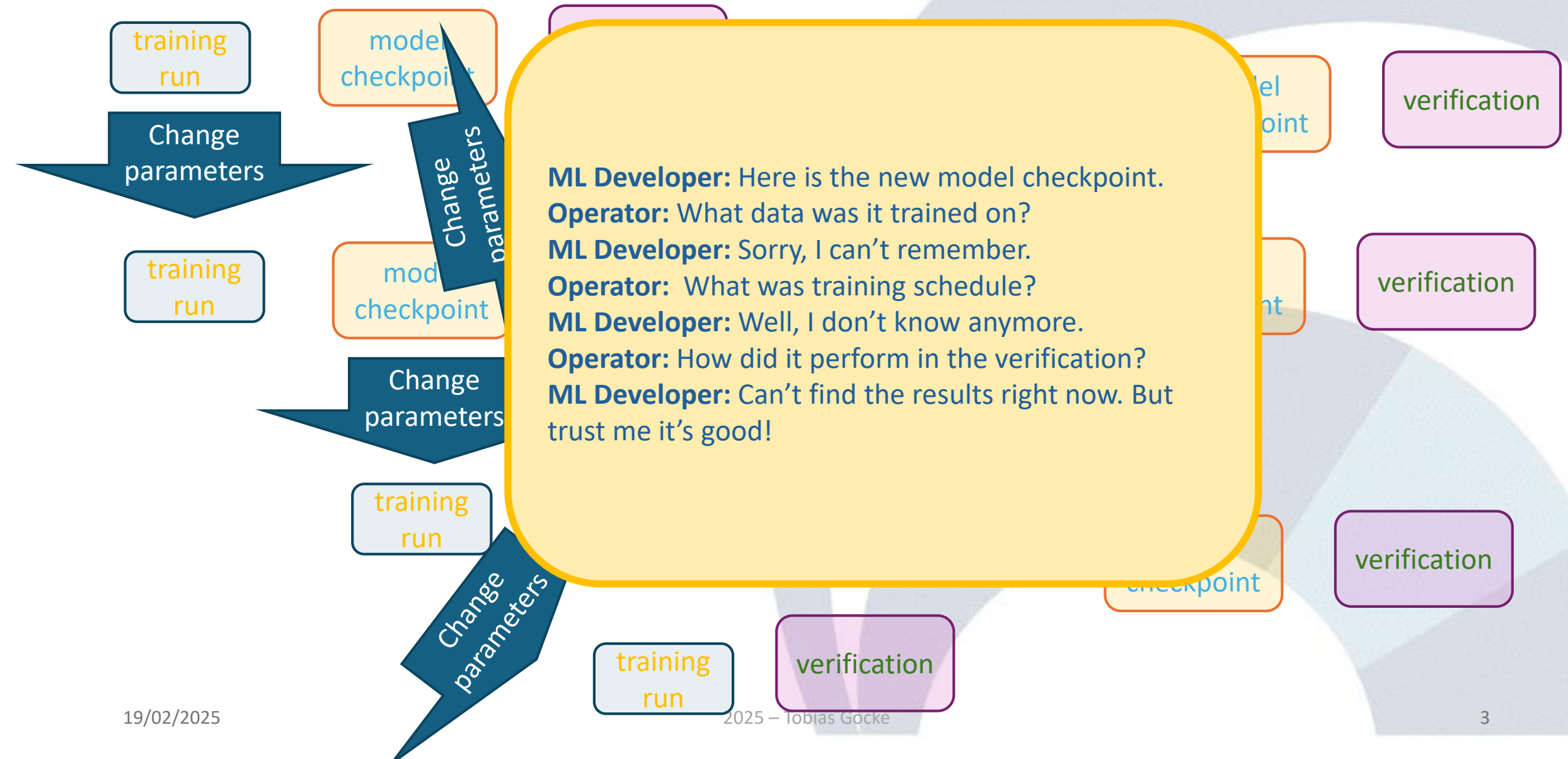
6.2 Basic tests with Pytest (20') [JD]

6.3 Setting up a runner (20') [FP]



AI generated Image,
© Marek Jacob 2024

Why experiment tracking?



Experiment tracking options (subjective comparison after short research)



- Open source
- User-friendly
- Well documented
- Configurable
- Flexible
- Integrates with different environment (pytorch)
- Can set up a local stand alone server easily



TensorBoard

- Integrates especially well with tensorflow
- Not ideal for large scale projects?



Weights & Biases

- Hosted service
- Has to be paid

Experiment tracking with MLFlow

- Place where to log and organize training runs
 - Experiments, each consisting of various runs
- Reproducibility
 - Log software versions, configs, model checkpoints
- Live monitoring
 - Validation metrics, system metrics, plots, etc
- Powerful tools to compare runs
 - Diffs in parameters
 - Multi run loss plots
 - Understand hyper parameters
- Different modes
 - local
 - Stand-alone server
 - Web-server



Quick-start mlflow

- Install

```
pip install mlflow
```

(virtual python environment)

- Log

```
import mlflow
mlflow.set_experiment("my_...")
[...]
with mlflow.start_run(run_name="..."):
    mlflow.log_params(...)
    mlflow.log_metric("...", ...)
    mlflow.log_figure(...)
    mlflow.log_artifact(...)
```

- View

```
mlflow ui
```

The screenshot shows the MLflow web interface. The top navigation bar includes the MLflow logo, version 2.16.2, and tabs for 'Experiments' and 'Models'. The 'Experiments' tab is active, displaying a list of experiments on the left and details for 'aicon_prototype_v3' on the right.

Experiments List:

- hyperparameter_study
- aifronts_felix
- seven
- hyperparameter_study_3
- hyperparameter_study_m...
- test_exp
- data_r3b5
- data_r3b6
- aicon_gpnl
- aicon_1y
- aicon_anemoui
- aicon_BIG
- aicon_train_exp
- forcing
- 2years

Experiment Details: aicon_prototype_v3

Uses anemoui version from January 2025 <https://gitlab.dkrz.de/aicon/anemoui/anemoui-core/-/commit/b091e84ab9a2be679f5>

Runs Table:

Run Name	Created	Duration	Models
rf15_h4_3h_2011_2021_val2022_clpprecmax	13 days ago	6.9h	-
rf15_h4_3h_2011_2021_val2022_clpprec	13 days ago	-	-
rf16_h5_3h_2011_2021_val2022_noclp	13 days ago	-	-
rf16_h5_3h_2011_2021_val2022_noclp	13 days ago	-	-
rf15_h4_3h_2011_2021_val2022_clpprec	14 days ago	-	-
rf15_h4_3h_2011_2021_val2022_clampstd	14 days ago	-	-
rf15_h4_3h_2011_2021_val2022_clampstd	16 days ago	-	-
rf15_h4_3h_2011_2021_val2022_noclamp	16 days ago	15.6d	AICON_prototype_v...

18 matching runs

Mlflow web user interface

Experiments

mlflow 2.16.2

Experiments

Models

172.16.112.218:5051/#/experiments/28?searchFilter=&orderByKey=attributes.start_time&orderByAsc=false&startTime=ALL&lifecycleFilter=Active&modelVersionFilter=All+Run

170%

GitHub

Docs

Experiments

Search Experiments

hyperparameter_study

aifronts_felix

seven

hyperparameter_study_3

hyperparameter_study_m...

test_exp

data_r3b5

data_r3b6

aicon_gpnl

aicon_1y

aicon_anemoi

aicon_BIG

aicon_train_exp

forcing

2years

aicon_prototype_v3

Provide Feedback

Uses anemoi version from January 2025 <https://gitlab.dkrz.de/aicon/anemoi/anemoi-core/-/commit/b091e84ab9a2be679f5>

Runs

Evaluation

Experimental

Traces

Experimental

metrics.rmse < 1 and params.model = "tree"

Time created

State: Active

Run Name	Created	Duration	Models
rf15_h4_3h_2011_2021_val2022_clpprecmax	13 days ago	6.9h	-
rf15_h4_3h_2011_2021_val2022_clpprec	13 days ago		-
rf16_h5_3h_2011_2021_val2022_noclp	13 days ago		-
rf16_h5_3h_2011_2021_val2022_noclp	13 days ago		-
rf15_h4_3h_2011_2021_val2022_clpprec	14 days ago		-
rf15_h4_3h_2011_2021_val2022_clampstd	14 days ago		-
rf15_h4_3h_2011_2021_val2022_clampstd	16 days ago		-
rf15_h4_3h_2011_2021_val2022_noclamp	16 days ago	15.6d	AICON_prototype_v...

18 matching runs

runs

Columns configurable

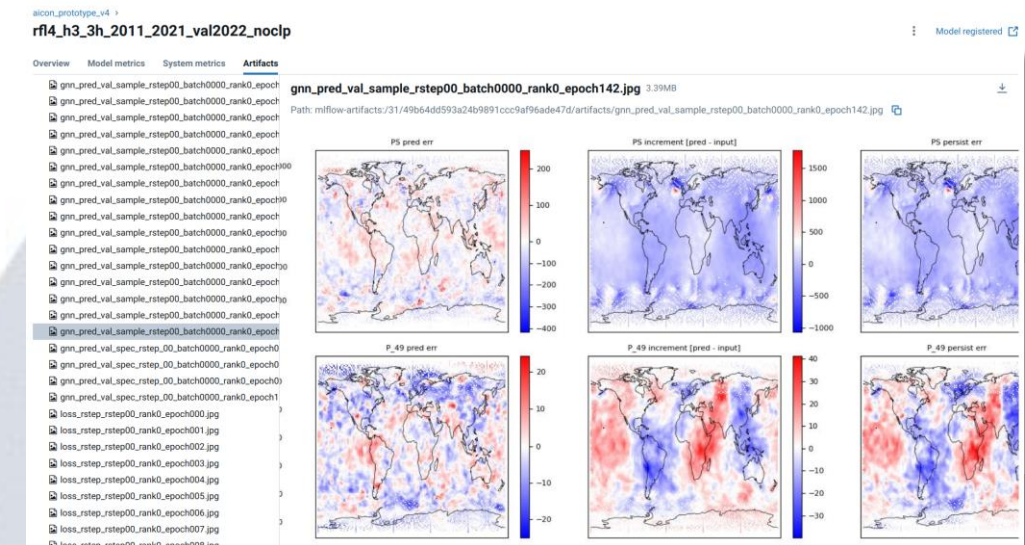
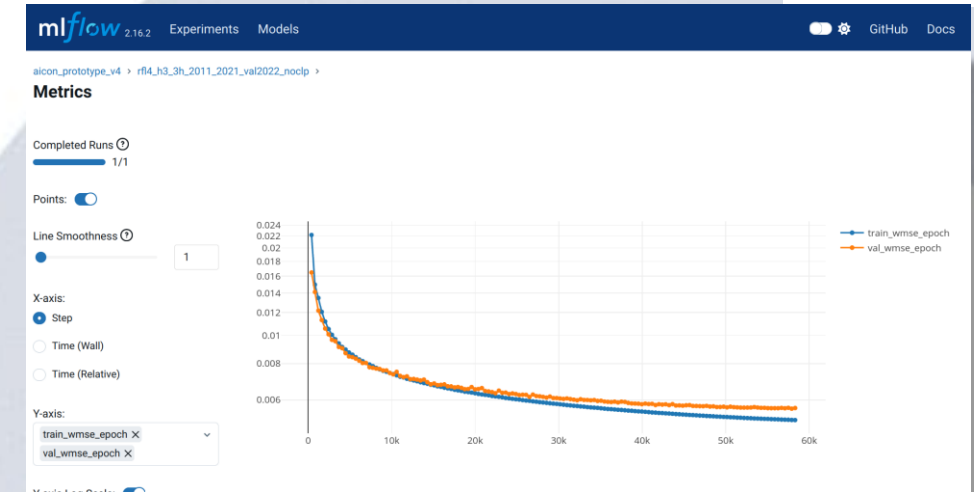
19/02/2

2025 – Tobias Göcke

7

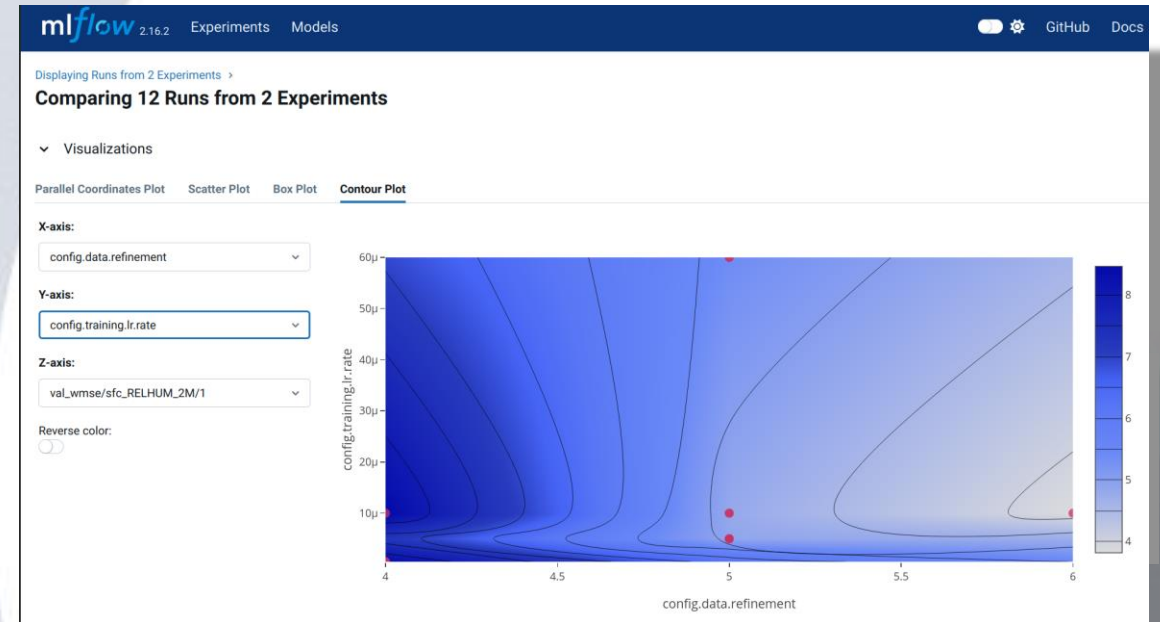
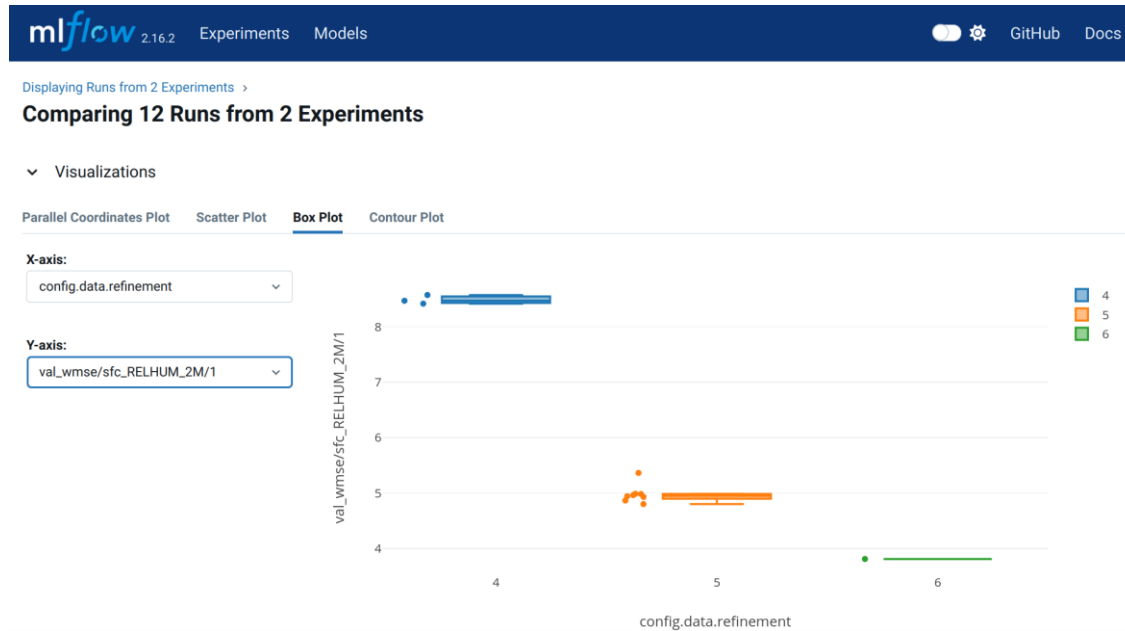
Mlflow web user interface

- Plot various metrics
- Compare different runs
- Log plots
- Compare parameter settings
- Summary plots for hyper parameter studies
- Compare across different experiments

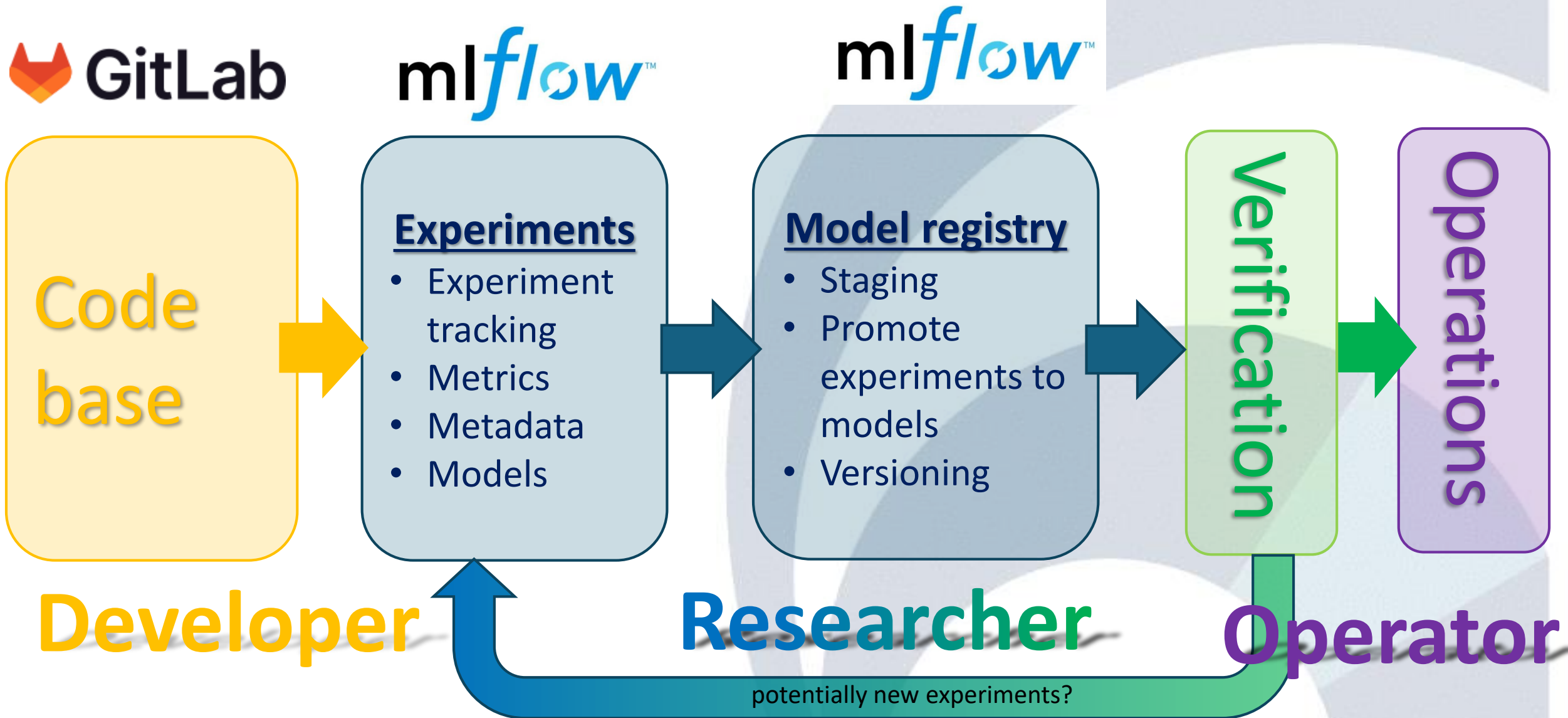


Mlflow web user interface

- Powerful tools to evaluate hyperparameters scans
- Box plots for single parameter dependency
- Contour plot for double parameter dependency



Mlflow in the MLOps pipeline (current early setup, DWD example)



Mlflow model registry

- Inference checkpoints can be logged as experiment artifacts
- These can be promoted to the model registry
- A model can have a description and tags
- Versions of that model can hold their own tags and descriptions
- Each version also links to a specific experiment run where the inference checkpoint can be found.
- The most interesting models can thus be stored in prominent place, augmented with information, potentially ready for inference
- @DWD we currently experiment with this tool. Use versions to document training history.

The screenshot displays the Mlflow Model Registry interface. At the top, the navigation bar includes 'mlflow 2.16.2', 'Experiments', and 'Models'. The 'Models' tab is active, showing a list of 'Registered Models'. The selected model is 'AICON_prototype_v3', created on 2025-02-13 at 10:13:08 and last modified on 2025-02-14 at 11:17:59.

The 'Description' section provides context: 'This model is based on an experiment series with R03B05 resolution. Initially the experiment was created to test the anemoi-core version and the clamping mechanism. Clamping was then post-pond because it did not work as expected. Some restarts were made with different clamping settings even (it worked?). The version that worked fine (<http://172.16.112.218:5051/#/experiments/28/runs/7b7952b8f89e435689f1d19cadf70dd3>) was used as a base version for this model. Then further rollouts were done as forks to improve long term stability of the mode. Those are appended as newer versions. of this model.'

The 'Tags' section shows a table with one tag: 'grid' with value 'R03B05'. Below this is a form to add new tags with fields for 'Name' and 'Value', and an 'Add' button.

The 'Versions' section includes a 'Compare' button and a table listing the model's versions. A toggle for 'New model registry UI' is visible in the top right.

Version	Registered at	Created by	Tags	Aliases	Description
Version 7	2025-02-13 11:53:02		rollout: 4 epochs: 194	Add	
Version 6	2025-02-13 11:52:14		epochs: 185 rollout: 4	Add	Due to somewhat clumsy confi...
Version 5	2025-02-13 11:51:46		rollout: 2 epochs: 181	Add	
Version 4	2025-02-13 11:51:31		rollout: 1 epochs: 176	Add	resumed run smaller LR init v...
Version 3	2025-02-13 11:51:18		rollout: 1 epochs: 121	Add	resumed run with fresh optimi...
Version 2	2025-02-13 11:50:44		rollout: 1 epochs: 116	Add	R03B05 run (without clamping)...

Summary

- Organize experiments
- Reproducibility
- Evaluate hyperparameter studies
- Register models
- Transition to inference, routine verification and operations

E-AI Basic Tutorials

Tutorial E-AI Basics 4: January Wednesday 22, 2025, 11-12 CET

"MLOps" - Machine Learning Operations

4.1 Overview (20') [RP]

4.2 MLOps in relation to traditional Weather forecasting (20') [MJ]

4.3 Road to MLOps (20') [DN]

5.2

Tutorial E-AI Basics 5: February Wednesday 19, 2025, 11-12 CET

MLflow - an open-source platform for managing the machine learning lifecycle

5.1 Overview - User perspective (20') [TG]

5.2 Logging to MLflow as a ML software developer (20') [HT]

5.3 Running MLflow server as a user and as a service (20') [MJ]

Tutorial E-AI Basics 6: February Wednesday 26, 2025, 11-12 CET

CI/CD - Continuous Integration and Continuous Deployment of ML codes

6.1 Overview – What can CI/CD do for you? (20') [MJ]

6.2 Basic tests with Pytest (20') [JD]

6.3 Setting up a runner (20') [FP]



AI generated Image,
© Marek Jacob 2024

Logging in MLFlow

Step 1

MLflow server for local experimentation

```
$ mlflow ui
```

Step 2

```
#Set up MLflow
import mlflow
mlflow.set_tracking_uri(uri="http://127.0.0.1:5000")
mlflow.set_experiment("Wind Chill Example")
```

The screenshot shows the MLflow web interface in a browser. The address bar highlights the URL `127.0.0.1:5000/#/experiments/1351956649343`. The MLflow logo and version `2.20.2` are in the top left. The navigation bar includes 'Experiments' and 'Models' tabs. The 'Experiments' section shows a list with 'Default' and 'Wind Chill Example' (selected with a green circle). The 'Wind Chill Example' details are shown on the right, including a search bar with `metrics.rmse < 1 and params.model = "tree"`, filters for 'Time created', 'State: Active', and 'Datasets', and a table of runs. The table has columns 'Run Name', 'Created', and 'Dataset'. One run is listed: 'logging 01' created '16 minutes ago'. A green box labeled 'Experiment name' points to the 'Wind Chill Example' entry in the list.

Run Name	Created	Dataset
logging 01	16 minutes ago	-

Logging in MLFlow

Start an MLflow run

Log parameters

Set tag

Training loop

Log metrics

```
with mlflow.start_run(run_name="logging 01"):

    # Log the hyperparameters
    mlflow.log_params({"hidden_dim": hidden_dim})

    mlflow.set_tag("Training Info", "Basic model for wind chill prediction")

    for epoch in range(n_epoch):
        optimizer.zero_grad() # Clear gradients

        y_pred = model(x_train) # Forward pass

        loss = criterion(y_pred, y_train) # Compute loss
        loss.backward() # Backpropagate error
        optimizer.step() # Update weights

        train_loss.append(loss.item()) # Save loss

        y_pred=model(x_val) # predict on validateion dataset
        vloss=criterion(y_pred,y_val)

        # Log the losses metrics
        mlflow.log_metric("loss", loss.item(), step=(epoch+1))
        mlflow.log_metric("val_loss", vloss.item(), step=(epoch+1))
```

Logging in MLFlow

Model signature →

```
# Infer the model signature  
signature = infer_signature(x_train, model(x_train))
```

Log model →


```
# Log the model  
mlflow.pytorch.log_model(model, "model", signature=signature)
```

Log figure →

```
# Log the figure  
mlflow.log_figure(plt.gcf(), "figure.png")
```

[Link](#)

Logged Run in UI


2.20.2

Experiments
Models

[GitHub](#)
[Docs](#)

[Wind Chill Example](#) >

logging 01

⋮

Register model








Overview

Model metrics


System metrics

Traces

Artifacts


Experiment ID	135195664934308426 
Status	 Finished
Run ID	df2f069914154f4c8f243f417c479f92 
Duration	6.3s
Datasets used	—
Tags	Training Info: Basic model for wind chill predict... 
Source	 example_with_logging.py  e888911
Logged models	 pytorch
Registered models	—

Parameters (1)

 Search parameters

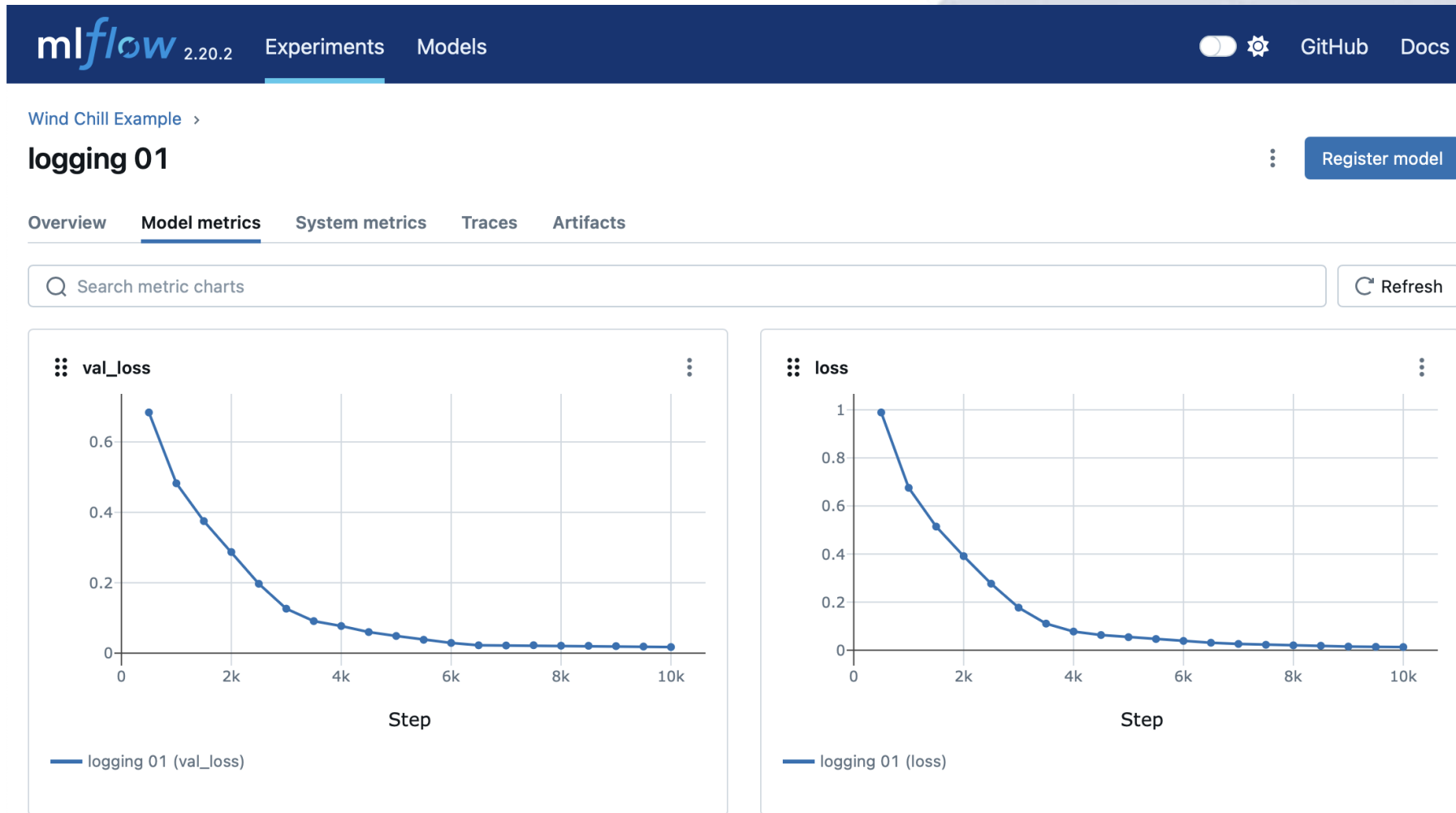
Parameter	Value
hidden_dim	20

Metrics (2)


 Search metrics

Metric	Value
val_loss	0.01734837330877781
loss	0.01294349879026413

Logged Run in UI



Logged Run in UI


2.20.2
Experiments
Models

[GitHub](#)
[Docs](#)

[Wind Chill Example >](#)

logging 01

Register model

Overview
Model metrics
System metrics
Traces
Artifacts

model

data

MLmodel

conda.yaml

python_env.yaml

requirements.txt

figure.png

model

Register model

Path: file:///Users/ecm6435/repos/e-ai-mlflow/mlruns/135195664934308426/df2f069914154f4c8f243f417c479...

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. You can also [register it to the model registry](#) to version control

Model schema

Input and output schema for your model. [Learn more](#)

Name	Type
Inputs (1)	
- (required)	any
Outputs (1)	
- (required)	any

Validate the model before deployment

Run the following code to validate model inference works on the example input data and logged model dependencies, prior to deploying it to a serving endpoint

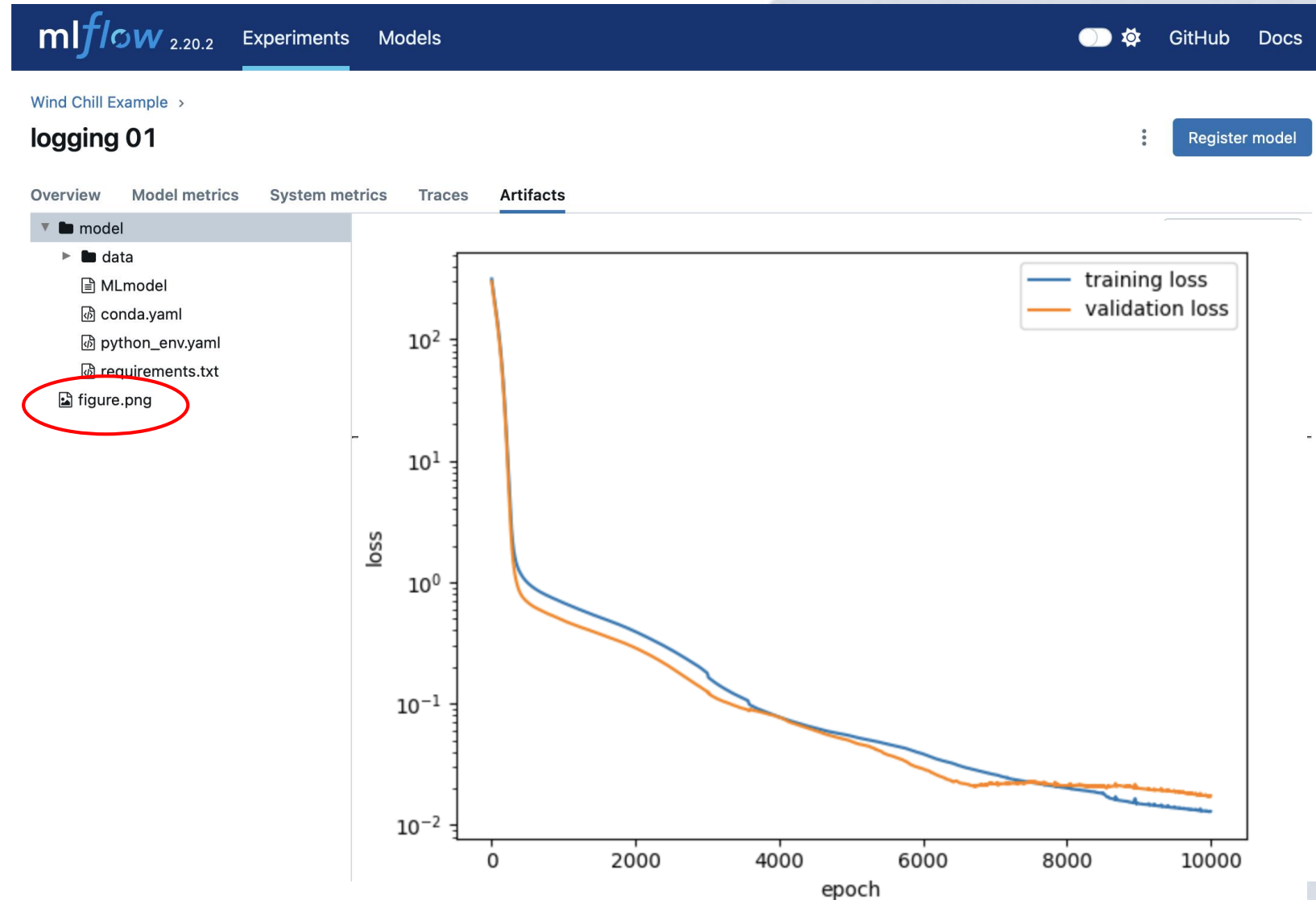
```
import mlflow

model_uri =
'runs:/df2f069914154f4c8f243f417c479f92/model'

# Replace INPUT_EXAMPLE with your own input example
to the model
# A valid input example is a data instance suitable
for pyfunc prediction
input_data = INPUT_EXAMPLE

# Verify the model with the provided input data using
the logged dependencies.
# For more details, refer to:
"
```

Logged Run in UI



Logging to the ECMWF MLflow server

Before starting a training run, you need to authenticate yourself to the MLflow server and obtain a token. A valid token is required before starting training.

This is done with the `anemai-training mlflow login` command.

The first time you run the command, you need to pass the URL with `--url` and you need to obtain a seed token:

```
$ anemai-training mlflow login --url https://mlflow.ecmwf.int
```

```
2024-10-14 10:54:10 INFO 🌐 Logging in to https://mlflow.ecmwf.int
```

```
2024-10-14 10:54:10 INFO 📝 Please obtain a seed refresh token from https://mlflow.ecmwf.int/seed
```

```
2024-10-14 10:54:10 INFO 📝 and paste it here (you will not see the output, just press enter after pasting):
```

```
Refresh Token: ***
```

```
2024-10-14 11:00:17 INFO Your MLflow login token is valid until 2024-11-12 11:00:17 UTC
```

```
2024-10-14 11:00:17 INFO ✅ Successfully logged in to MLflow. Happy logging!
```

Logging to the ECMWF MLflow server using the AnemoiMLflowClient

Unless your codebase is compatible with anemoi-training, it's recommended to install it without dependencies:

```
$ pip install anemoi-utils mlflow  
$ pip install anemoi-training --no-deps
```

pip might complain about missing dependencies, but you can ignore that.

You can use the custom mlflow client with authentication like this. It behaves like a normal `mlflow.MlflowClient`:

```
from anemoi.training.diagnostics.mlflow.client import AnemoiMlflowClient  
  
client = AnemoiMlflowClient("https://mlflow.ecmwf.int", authentication=True)  
  
# do regular mlflow client things  
client.search_experiments()  
client.log_artifact(...)
```

E-AI Basic Tutorials

Tutorial E-AI Basics 4: January Wednesday 22, 2025, 11-12 CET

"MLOps" - Machine Learning Operations

4.1 Overview (20') [RP]

4.2 MLOps in relation to traditional Weather forecasting (20') [MJ]

4.3 Road to MLOps (20') [DN]

5.3

Tutorial E-AI Basics 5: February Wednesday 19, 2025, 11-12 CET

MLflow - an open-source platform for managing the machine learning lifecycle

5.1 Overview - User perspective (20') [TG]

5.2 Logging to MLflow as a ML software developer (20') [HT]

5.3 Running MLflow server as a user and as a service (20') [MJ]

Tutorial E-AI Basics 6: February Wednesday 26, 2025, 11-12 CET

CI/CD - Continuous Integration and Continuous Deployment of ML codes

6.1 Overview – What can CI/CD do for you? (20') [MJ]

6.2 Basic tests with Pytest (20') [JD]

6.3 Setting up a runner (20') [FP]



AI generated Image,
© Marek Jacob 2024

5.3 Running MLflow server as a user and as a multi-user service

Run MLflow offline

```
import mlflow
with mlflow.start_run():
    mlflow.log_param("lr", 0.001)
```

If **tracking_uri** is not specified, MLflow logs data into local **mlruns** directory.

```
$ cat mlruns/0/7d*/params/lr
0.001
```

```
majacob@oflws12 $ tree
.
├── mlruns/
│   ├── 0/
│   │   ├── 7d0eebb3d24c485c9bc318c395d14449/
│   │   │   ├── artifacts/
│   │   │   ├── meta.yaml
│   │   │   ├── metrics/
│   │   │   ├── params/
│   │   │   │   └── lr
│   │   │   └── tags/
│   │   │       ├── mlflow.runName
│   │   │       ├── mlflow.source.name
│   │   │       ├── mlflow.source.type
│   │   │       └── mlflow.user
│   │   └── meta.yaml
│   └── models/
└── 8 directories, 7 files
```

Visualization with MLflow

```
$ mlflow server  
# or  
$ mlflow ui # alias for server
```

“Server” is available at <http://127.0.0.1:5000>
(for all users of localhost)

```
Terminal - majacob@oflws12: /data/majacob/tutorials/tutorial5  
majacob@oflws12 $ mlflow server  
[2025-02-17 13:18:25 +0100] [27308] [INFO] Starting gunicorn 23.0.0  
[2025-02-17 13:18:25 +0100] [27308] [INFO] Listening at: http://127.0.0.1:5000  
(27308)  
[2025-02-17 13:18:25 +0100] [27308] [INFO] Using worker: sync  
[2025-02-17 13:18:25 +0100] [27309] [INFO] Booting worker with pid: 27309  
[2025-02-17 13:18:25 +0100] [27310] [INFO] Booting worker with pid: 27310
```

mlflow 2.19.0 Experiments Models

Experiments

Search Experiments

Default

Default

Provide Feedback Add Description

Runs Evaluation Experimental Traces Experimental

metrics.rmse < 1 and params.model = "tree"

Time created State: Active Datasets

Sort: Created Columns Group by

Run Name	Created	Dataset	Duration	Source
marvelous-steed-593	16 minutes ago	-	15ms	ipython

mlflow 2.19.0 Experiments Models

Default

marvelous-steed-593

Overview Model metrics System metrics Artifacts

Description

No description

Search parameters

Parameter	Value
lr	0.001

No metrics recorded

MLflow tracking server

- MLflow tracking server is a stand-alone HTTP server that serves multiple REST API endpoints for tracking runs/experiments
- **Collaboration:** Multiple users can log runs to the same endpoint, and query runs and models logged by other users.
- **Sharing Results:** The tracking server also serves Tracking UI endpoint, where team members can easily explore each other's results.
- **Centralized Access:** The tracking server can be run as a proxy for the remote access for metadata and artifacts, making it easier to secure and audit access to data.

<https://mlflow.org/docs/latest/tracking/tutorials/remote-server.html>

MLflow server

0.1:5000/#/experiments/0?searchFilter=&orderByKey=attrib ☆ Q Search

Experiments Models GitHub Docs

Default Provide Feedback Add Description Share

Runs Evaluation Experimental Traces Experimental

metrics.rmse < 1 and params.model = "tree"

Time created State: Active Datasets

Sort: Created Columns Group by

Run Name	Created	Dataset
bold-smelt-568	3 minutes ago	-
marvelous-steed-593	2 hours ago	-

127.0.0.1:5000 is available to all users on the host!

```
Terminal - majacob@oflws12: /data/majacob/tutorials/tutorial5
majacob@oflws12 $ mlflow server
[2025-02-17 13:18:25 +0100] [27308] [INFO] Starting gunicorn 23.0.0
[2025-02-17 13:18:25 +0100] [27308] [INFO] Listening at: http://127.0.0.1:5000 (27308)
[2025-02-17 13:18:25 +0100] [27308] [INFO] Using worker: sync
[2025-02-17 13:18:25 +0100] [27309] [INFO] Booting worker with pid: 27309
[2025-02-17 13:18:25 +0100] [27310] [INFO] Booting worker with pid: 27310
```

Communication via network
Internal REST API

```
Terminal - IPython: tutorials/tutorial5
majacob@oflws12 $ ipython
Python 3.11.10 (main, Sep 18 2024, 22:13:39) [GCC]
Type 'copyright', 'credits' or 'license' for more information
IPython 8.30.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import mlflow
...: mlflow.set_tracking_uri("http://127.0.0.1:5000")
...:
...: with mlflow.start_run():
...:     mlflow.log_param("lr", 0.002)
...:
View run bold-smelt-568 at: http://127.0.0.1:5000/#/experiments/0/runs/64d70d2cfa094860b21abb68590f8a2d
View experiment at: http://127.0.0.1:5000/#/experiments/0

In [2]:
```


From local server to network server

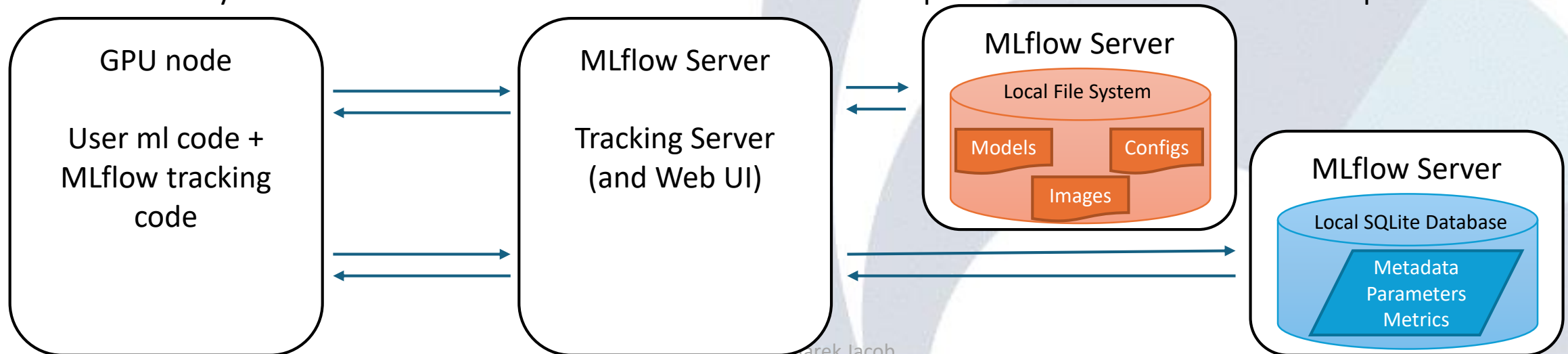
- Open server to other hosts in the network
`mlflow server --host 0.0.0.0 --port 5000`
(port is optionally)
-- host: The network address to listen on
(default: 127.0.0.1).
Use 0.0.0.0 to bind to all addresses if you want to access the tracking server from other machines.

Assumption: Your machine is part of a trusted network.

Advice: implement **authentication** and **encryption** if the server is connected directly to the internet.

Storage backends

- **Backend store:** Database to store all metadata, parameters and metrics.
Default: file based (limited performance, many small files).
Improved performance with SQLAlchemy-compatible **database** (e.g., SQLite, PostgreSQL, MySQL).
`--backend-store-uri "sqlite:///${DIR}/mlflow-store.db"`
- **Artifact store:** Store images, models, configs, etc.
E.g. Amazon S3 and S3-compatible storage; Azure Blob Storage; Google Cloud Storage; (S)FTP server; NFS; HDFS; local file system
`--artifacts-destination "${DIR}/mlflow-artifacts"`
MLflow server proxies access to artifacts
 - Only MLflow server sees bucket credentials and uses the permissions of the mlflow server process



Basic Authentication

- `export MLFLOW_AUTH_CONFIG_PATH="${DIR}/auth_config.ini"`
`mlflow server --app-name basic-auth`
- Implements basic HTTP authentication with **username** and **password**

auth_config.ini

```
[mlflow]
default_permission = READ
database_uri = sqlite:///work/some/path/basic_auth.db
admin_username = admin
admin_password = to-be-changed
authorization_function = mlflow.server.auth:authenticate_request_basic_auth
```

Update the default admin password as soon as possible!

- “This feature is still experimental and may change in a future release without warning.”
- (“admin” user is mandatory)

<https://mlflow.org/docs/latest/auth/index.html>

Basic Authentication

- `export MLFLOW_AUTH_CONFIG_PATH="${DIR}/auth_config.ini"`
`mlflow server --app-name basic-auth`
- Implements basic HTTP authentication with **username** and **password**

auth_config.ini

```
[mlflow]
default_permission = READ
database_uri = sqlite:///work/some/path/basic_auth.db
admin_username
admin_password
authorization_
```

`sqlite:///work/some/path/basic_auth.db`

Protocol: `sqlite://`

Host: *(empty)*

Separator between Host and Path: `/`

Path (absolute or relative): `/work/some/path/basic_auth.db`

→ <https://docs.sqlalchemy.org/en/13/dialects/sqlite.html#connect-strings>

as soon as possible!

basic_auth

ease

- “This feat without w...g
- (“admin” user is mandatory)

<https://mlflow.org/docs/latest/auth/index.html>

User Administration via Python API

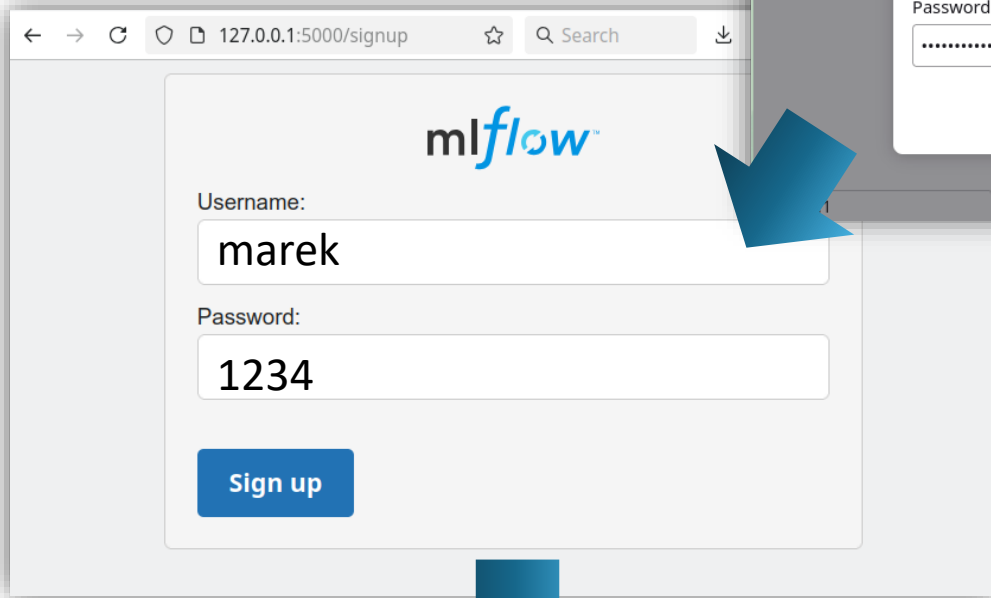
```
$ MLFLOW_TRACKING_USERNAME=admin MLFLOW_TRACKING_PASSWORD=to-be-changed ipython  
IPython 8.30.0 -- An enhanced Interactive Python. Type '?' for help.
```

```
In [1]: from mlflow.server import get_app_client  
from getpass import getpass  
tracking_uri = "http://127.0.0.1:5000/"  
user = "admin"  
# do not `password = "my_new_password"! Ipython would log the new password  
password = getpass(f"Please enter a new password:\n")  
auth_client = get_app_client("basic-auth", tracking_uri=tracking_uri)  
auth_client.update_user_password(user, password)
```

Further API endpoints for permissions, user creation, etc. available in the documentation:
<https://mlflow.org/docs/latest/auth/index.html>

Add New User

127.0.0.1:5000/signup

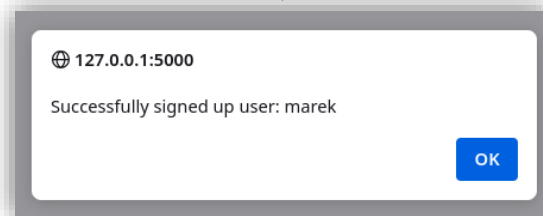


mlflow™

Username:
marek

Password:
1234

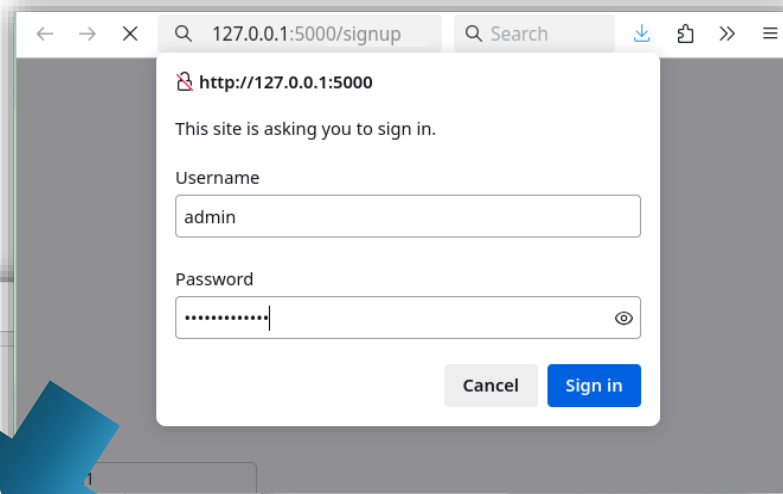
Sign up



127.0.0.1:5000

Successfully signed up user: marek

OK



127.0.0.1:5000/signup

http://127.0.0.1:5000

This site is asking you to sign in.

Username
admin

Password
.....

Cancel Sign in

Provide Credentials to MLflow Client

```
$ mkdir ~/.mlflow
$ touch ~/.mlflow/credentials
$ chmod 600 ~/.mlflow/credentials
$ vim ~/.mlflow/credentials
```

```
~/.mlflow/credentials
[mlflow]
mlflow_tracking_username = marek
mlflow_tracking_password = 1234
```

Alternative Environment Variables:
MLFLOW_TRACKING_USERNAME
MLFLOW_TRACKING_PASSWORD

User Setup Convenience

```
majacob@oflws12 $ python mlflow_setup.py
/home/majacob/.mlflow/credentials does not exist...
... create a new one
Please enter your mlflow username for server http://localhost:5000/:
marek
Please enter your mlflow (initial) password:
  ← Enter 1234 secretly
... testing user marek
... testing user marek
Please enter a new password for mlflow on http://localhost:5000/:

Please repeat that password:

... password updated in /home/majacob/.mlflow/credentials
... an successfully tested on http://localhost:5000/
```

```
majacob@oflws12 $ cat ~/.mlflow/credentials
[mlflow]
mlflow_tracking_username = marek
mlflow_tracking_password = 1234
```

mlflow_setup.py

```
17 import configparser
18 from getpass import getpass
19 import os
20 import sys
21 import pathlib
22
23 from mlflow.server import get_app_client
24
25 # Configure you ml flow server
26 tracking_uri = "http://localhost:5000/"
27
28
29 def setup_config(config_file):
30     """
31     """
32     print(f"{config_file} does not exist...")
33     print("... create a new one")
34
35     config_file.parent.mkdir(mode=0o700, parents=True, exist_ok=True)
36     user = input(f"Please enter your mlflow username for server {tracking_uri}:\n")
37     password = getpass(f"Please enter your mlflow (initial) password:\n")
38
39     # create empty file
40     open(config_file, "w").close()
41
42     # set permissions to user read/write only
43     config_file.chmod(0o600)
44
45     with open(config_file, "a") as f:
46         f.write("[mlflow]\n")
47         f.write(f"mlflow_tracking_username = {user}\n")
48         f.write(f"mlflow_tracking_password = {password}\n")
49
50     try:
51         print(f"    ... testing user {user}")
52         test_connection(user)
53     except Exception as e:
54         print(e)
55         print("Wrong username or password.")
56         os.remove(config_file)
57         print(f"    ... deleting {config_file}")
58         sys.exit(1)
59
60
61 test_connection(user):
62     client = get_app_client("basic-auth", tracking_uri=tracking_uri)
```

MLflow permissions

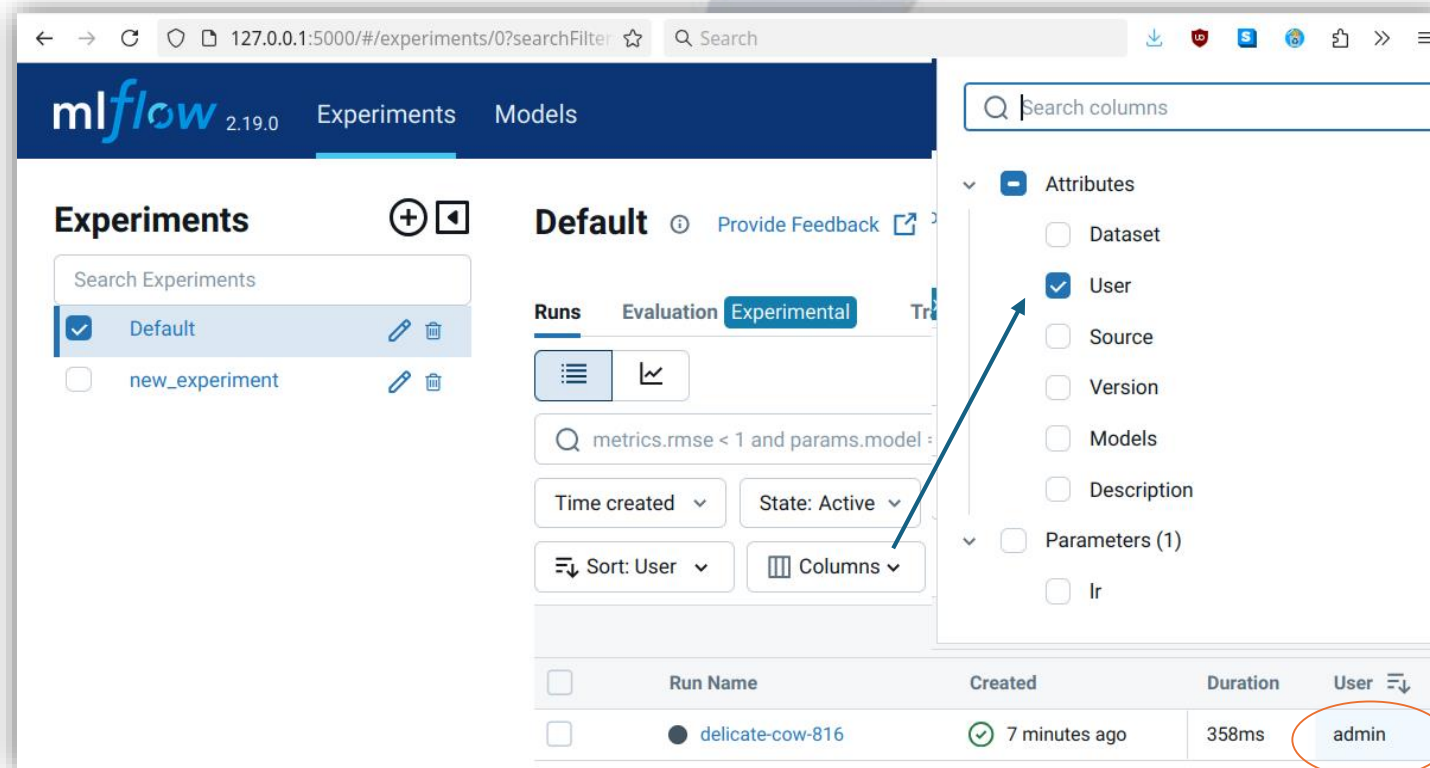
- User can not delete/edit foreign experiments
- User can not extend foreign experiments
 - Can't log to "Default" experiment
 - Start new experiment:

```
mlflow.set_experiment("new_experiment")
```

```
In [1]: import mlflow
...: mlflow.set_tracking_uri("http://127.0.0.1:5000")
...:
...: with mlflow.start_run():
...:     mlflow.log_param("lr", 0.002)
...:
```

MlflowException Traceback (most recent call last)

MlflowException: API request to endpoint /api/2.0/mlflow/runs/create failed with error code 403 != 200. Response body: 'Permission denied'



Recap: MLflow Server command

- `pip install mlflow`
- `export MLFLOW_AUTH_CONFIG_PATH="${DIR}/auth_config.ini"`
`export OPENBLAS_NUM_THREADS=1 # reduce virt. mem footprint`
`mlflow server \`
 - `--app-name basic-auth \`
 - `--backend-store-uri "sqlite:///${DIR}/mlflow.db" \`
 - `--artifacts-destination "${DIR}/mlflow-artifacts" \`
 - `--workers 10 \ # increase number of workers`
 - `--host 0.0.0.0 --port 5000`
- Create user account using the admin account:
<http://127.0.0.1:5000/signup>
- Use MLflow client API to change passwords
- Advanced steps for public MLflow server:
 - Hide MLflow server behind HTTPS Reverse Proxy (NGINX or Apache httpd)
 - Align authentication with institutional identity management (auth mechanism is pluggable)

Run MLflow in a Screen session

```
#!/usr/bin/env bash
set -e
DIR=$( cd -- "$( dirname -- "${BASH_SOURCE[0]}" )" &> /dev/null && pwd )
cd "$DIR"

SCREEN_SESSION=mlflow
export MLFLOW_AUTH_CONFIG_PATH="${DIR}/auth_config.ini"

export OPENBLAS_NUM_THREADS=1

send_to_screen(){
  # Replace occurrences of $ with \$ to prevent variable substitution:
  string="${1//\$/\$\$}"
  screen -xr $SCREEN_SESSION -X stuff "$string\r"
}

# start a detached screen session
screen -dmS $SCREEN_SESSION

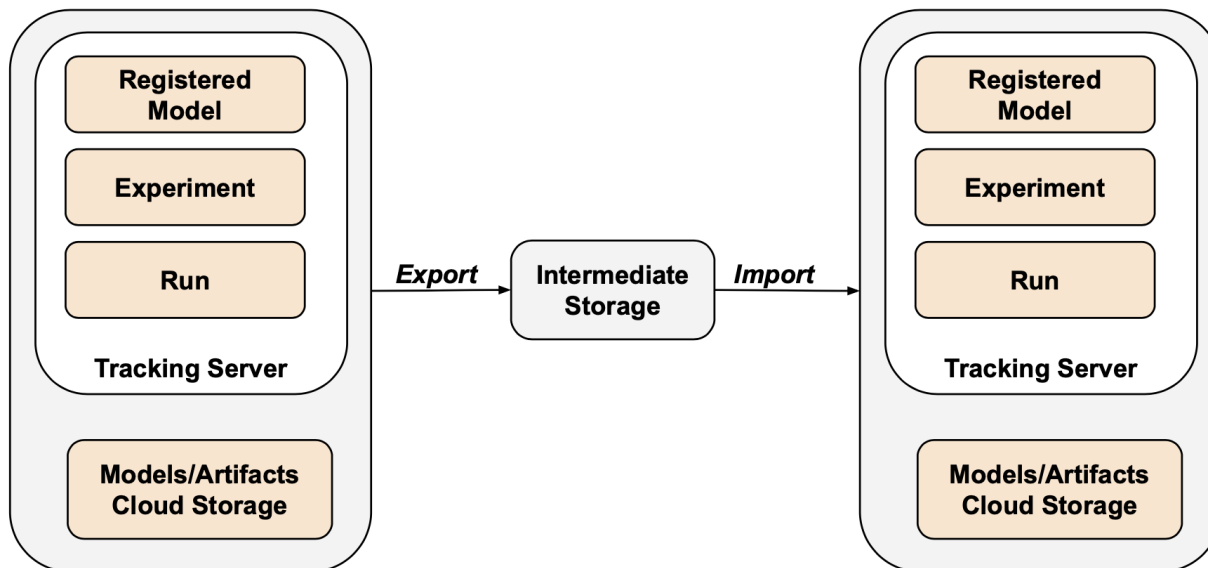
send_to_screen "date"
send_to_screen "echo \$PWD"
send_to_screen "echo \$MLFLOW_AUTH_CONFIG_PATH"
send_to_screen "source \"${DIR}/mlflow_venv/bin/activate\""
send_to_screen "mlflow server --app-name basic-auth --backend-store-uri \"sqlite:///${DIR}/mlflow.db\" --artifacts-destination \"${DIR}/mlflow-artifacts\" --workers 10 --host 0.0.0.0 --port 5000"
echo "Started mlflow in a detached screen session."
echo "Enter \"screen -xr $SCREEN_SESSION\" to attach."
echo "Then press 'ctrl+a d' to detach."
```

GNU Screen

- Terminal multiplexer
- Virtual consoles (terminals) are organised in sessions
- Allows a user to detach and reattach a session
- Separates processes from login session
- (Alternative to tmux)

MLflow Export Import

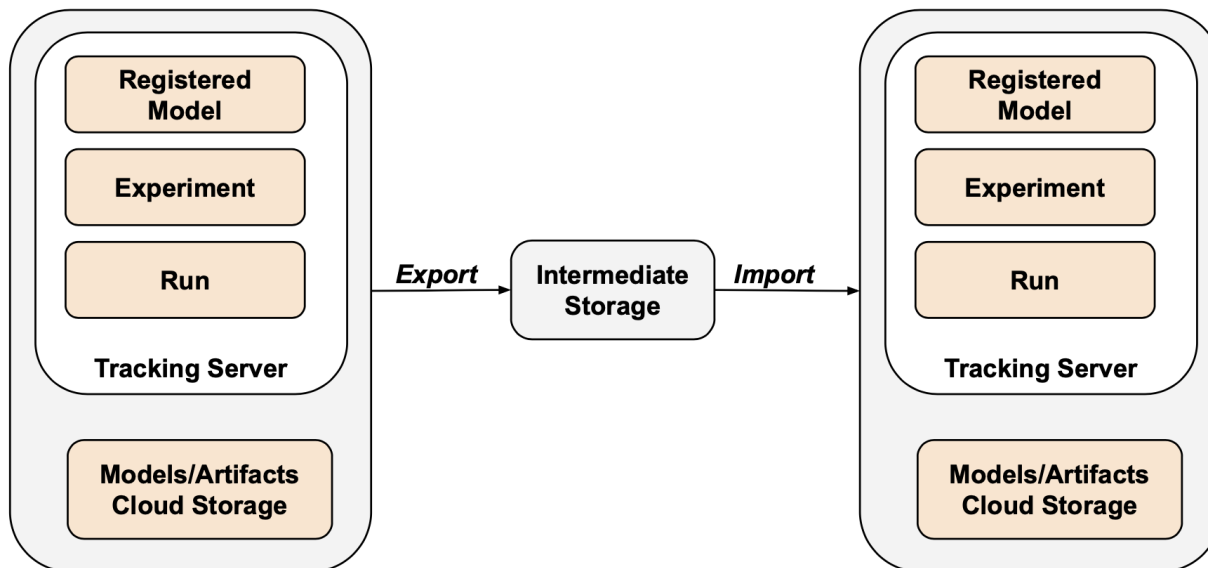
- Copy MLflow objects from one tracking server to another.
 - Also from file-based offline logging to a server.



- `pip install git+https://github.com/mlflow/mlflow-export-import/#egg=mlflow-export-import`
- 1. Export:
`export MLFLOW_TRACKING_URI=http://localhost:5000`
Note start `mlflow server` when exporting from file-based offline log
`export-experiment` --experiment Default --output-dir /tmp/export
- 2. Import:
`export MLFLOW_TRACKING_URI=http://some.remote:5001`
`import-experiment` --experiment-name Default2 --input-dir /tmp/export
- Authentication:
only with environment variables or
`export MLFLOW_TRACKING_URI=http://$USER:$PASS@some.remote:5001`

MLflow Export Import

- Copy MLflow objects from one tracking server to another.
 - Also from file-based offline logging to a server.



- `pip install git+https://github.com/mlflow/mlflow-export-import/#egg=mlflow-export-import`

- 1. Export:

```
export MLFLOW_TRACKING_URI=http://localhost:5000
# Note start `mlflow server` when exporting from f
export-experiment --experiment Default --output-di
```

- 2. Import:

```
export MLFLOW_TRACKING_URI=http://some.remote:5001
import-experiment --experiment-name Default2 --inp
```

- Authentication:

only with environment variables or

```
export MLFLOW_TRACKING_URI=http://$USER:$PASS@some
```

Anemoi:

The `anemoi-training mlflow sync` command wraps `mlflow-export-import`.

<https://anemoi.readthedocs.io/projects/training/en/latest/user-guide/tracking.html#logging-offline-and-syncing-with-an-online-server>

Further features of MLflow

- Model Registry
- Deployment
 - MLflow can deploy a model and make its inference available via REST API

```
from mlflow.models import infer_signature
signature = infer_signature(x_val.numpy(), model(x_val).detach().numpy())
model_info = mlflow.pytorch.log_model(model, "model", signature=signature)
```

```
mlflow models serve -m runs:/<run_id>/model -p 5000
```

You can then send a test request to the server as follows:

```
curl http://127.0.0.1:5000/invocations -H "Content-Type:application/json" --data '{"inputs": [[1, 2], [3, 4], [5, 6]]}'
```

- Can also be generated as docker container
- MLflow Recipes (Pipelines)
 - Define Recipes by combining multiple Steps (individual ML operations, e.g. ingesting data, fitting an estimator, evaluating, deployment)
 - For smaller problems, that share typical building blocks
- Model Evaluation
 - MLflow offers default evaluation methods for LLMs and classical ML codes.

Summary - Running MLflow server as a user and as a multi-user service

- MLflow is a powerful multi-user monitoring tool
 - Collaboration
 - Sharing Results
 - Document Training Linage
- Supports distributed services
 - MLflow tracking server
 - Metadata, parameters and metrics server
 - Artifact server
- Access control
 - Simple HTTP auth
- Data migration between servers and offline runs possible
 - Mlflow-export-import (+ Anemoi-training wrapper)

Further Information on E-AI

- Slides available at GitHub
<https://github.com/eumetnet-e-ai/tutorials>
- Recording will be available at EUMETNET SharePoint
<https://tInt19059.sharepoint.com/:f:/r/sites/E-AI/Shared%20Documents/Tutorials>
- Register for E-AI updates and SharePoint Access:
marek.jacob@eumetnet.eu
- Contacts:
 - Tobias Göcke: Tobias.Goecke@dwd.de
 - Helen Theissen: Helen.Theissen@ecmwf.int
 - Marek Jacob: Marek.Jacob@dwd.de
- E-AI Working Group „WG3 Operations“ for exchange on MLOps
<https://chat.europeanweather.cloud>

E-AI Basic Tutorials

Tutorial E-AI Basics 4: January Wednesday 22, 2025, 11-12 CET

"MLOps" - Machine Learning Operations

4.1 Overview (20') [RP]

4.2 MLOps in relation to traditional Weather forecasting (20') [MJ]

4.3 Road to MLOps (20') [DN]

Tutorial E-AI Basics 5: February Wednesday 19, 2025, 11-12 CET

MLflow - an open-source platform for managing the machine learning lifecycle

5.1 Overview - User perspective (20') [TG]

5.2 Logging to MLflow as a ML software developer (20') [HT]

5.3 Running MLflow server as a user and as a service (20') [MJ]

Tutorial E-AI Basics 6: **February Wednesday 26**, 2025, 11-12 CET

CI/CD - Continuous Integration and Continuous Deployment of ML codes

6.1 Overview – What can CI/CD do for you? (20') [MJ]

6.2 Basic tests with Pytest (20') [JD]

6.3 Setting up a runner (20') [FP]



AI generated Image,
© Marek Jacob 2024