

# **MLOps Engineering**

## **Machine Learning Operations V2.0.0**

### **Sessions 8 - 9**

**MsC in Business Analytics and Data Science**

**Madrid, May 2025**

# Agenda

- **Q&A Project phase**
- **15' Quiz**
- **Towards ML pipeline automation**
- **WandB and Mlflow 101**

# Evaluation methodology

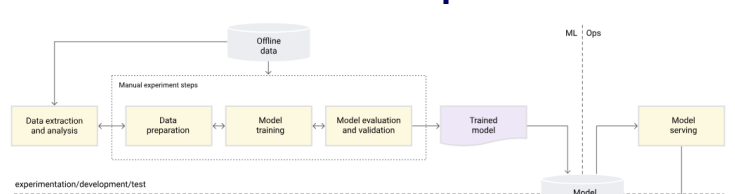
## v2.0.0

Class Participation: **0.0**  
Final exam fail: **3.5/10.0**

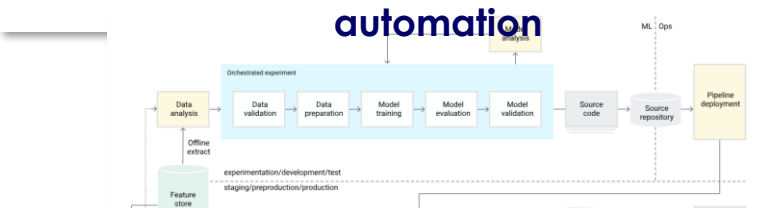
Activity	Weight	Description	(Tentative) Deadline
Ind. Class participation	20%	0 Relevant contributions in class and/ or via Teams	ON GOING
1st Group <b>Presentation</b>	15%	1 1st group deliverable (Business case, VC'ed, production-ready-code)	SESSION 8 (29 <sup>st</sup> May)
Intermediate test	10%	2 Initial core concepts & fundamental best practices	SESSION 9 (29 <sup>th</sup> May)
2 <sup>nd</sup> Group Work <b>Presentation</b>	25%	3 Final group project - Presentation (End-to-end CI/ CD)	SESSION 14 (25 <sup>th</sup> Jun)
Ind. Final Exam	30%	4 Final closed-book exam	SESSIONS 15 (25 <sup>th</sup> Jun)
Total	100%		

# MLOps is not a destination, but a journey. Involving people, processes, tools, data and governance

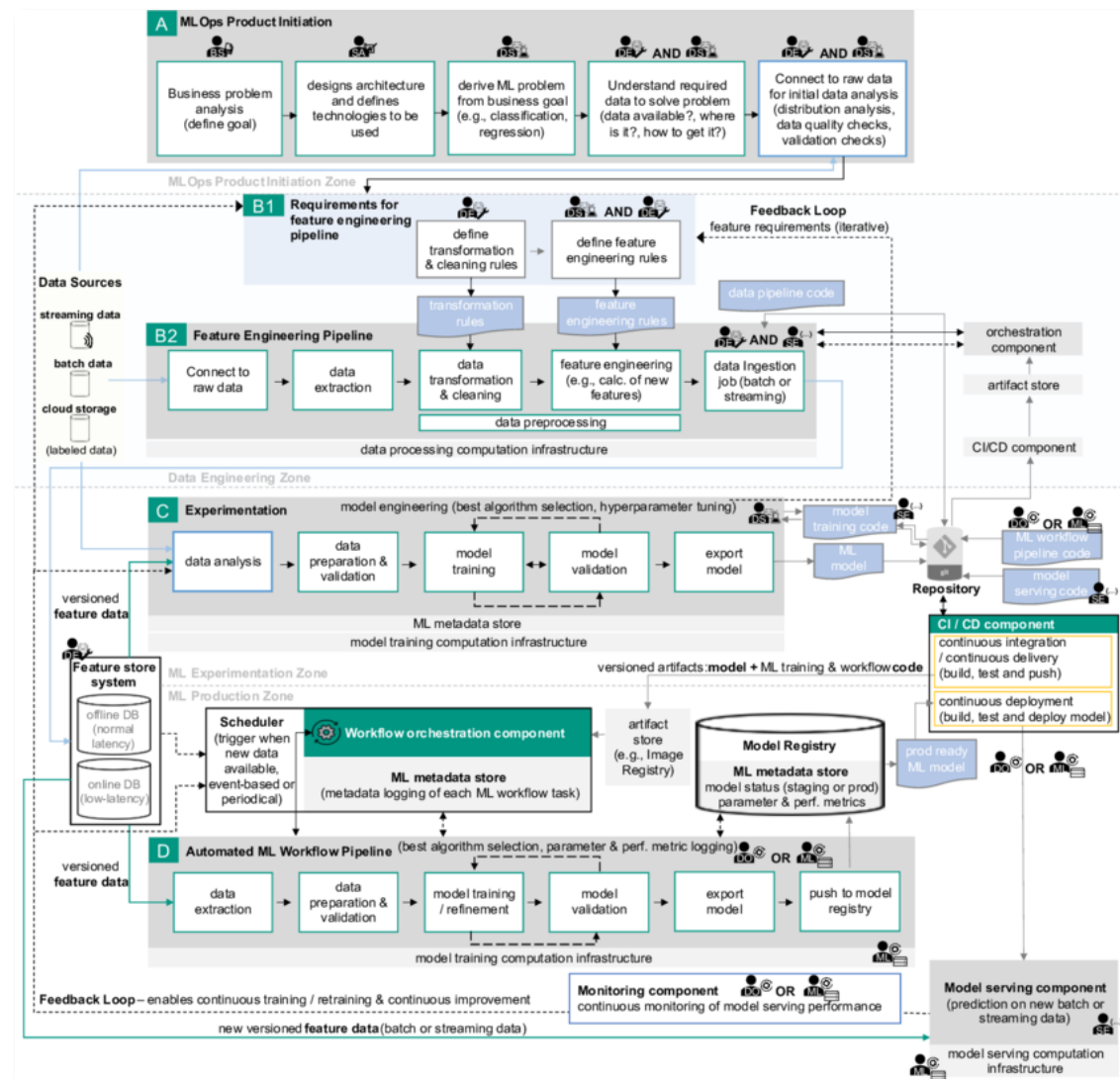
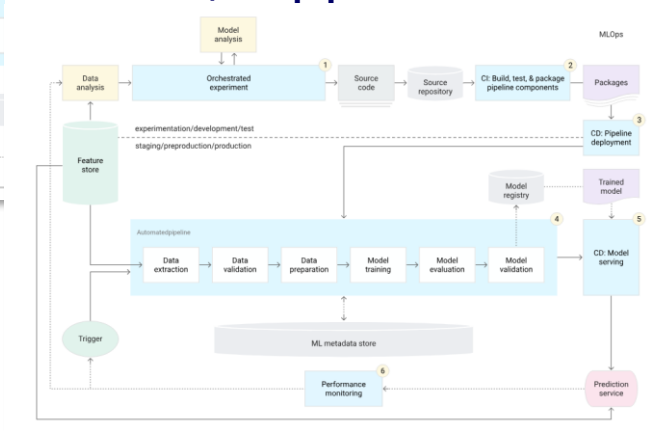
## Level 0: Manual process



## MLOps level 1: ML pipeline automation

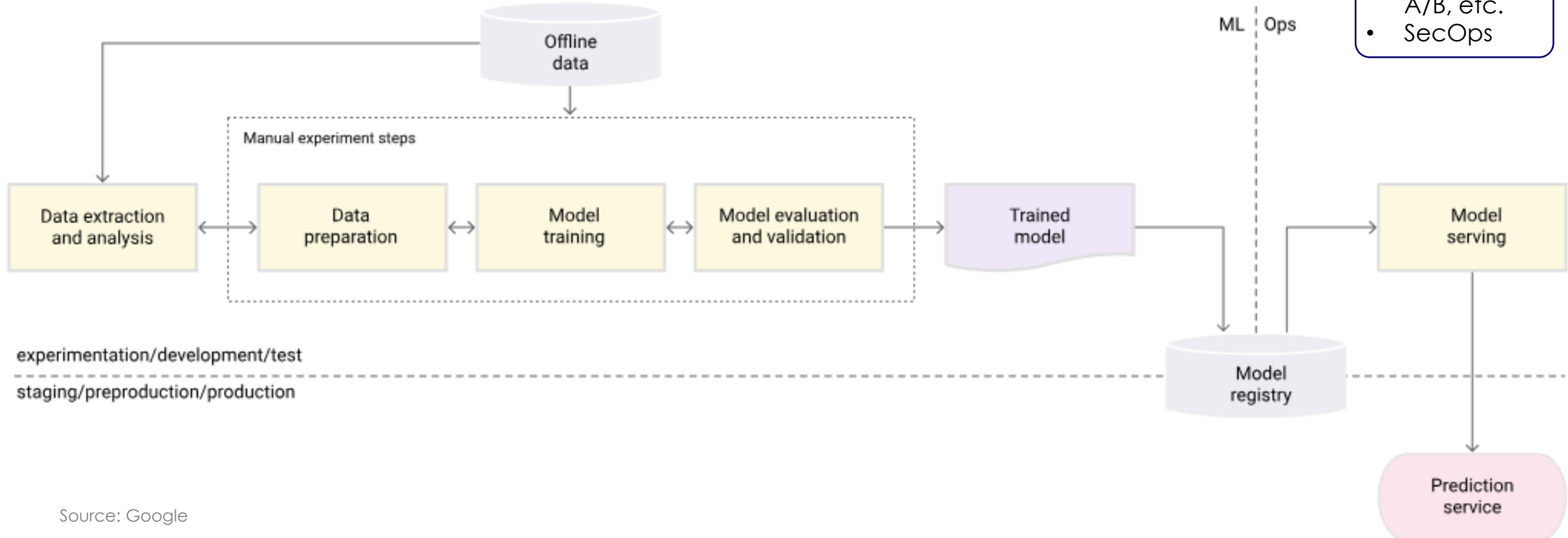


## Level 2: CI/CD pipeline automation



# Level 0: Manual ML workflows hinder scalability and reliability, highlighting the need for MLOps automation to streamline operations and reduce errors

Business Problem/ Opportunity definition



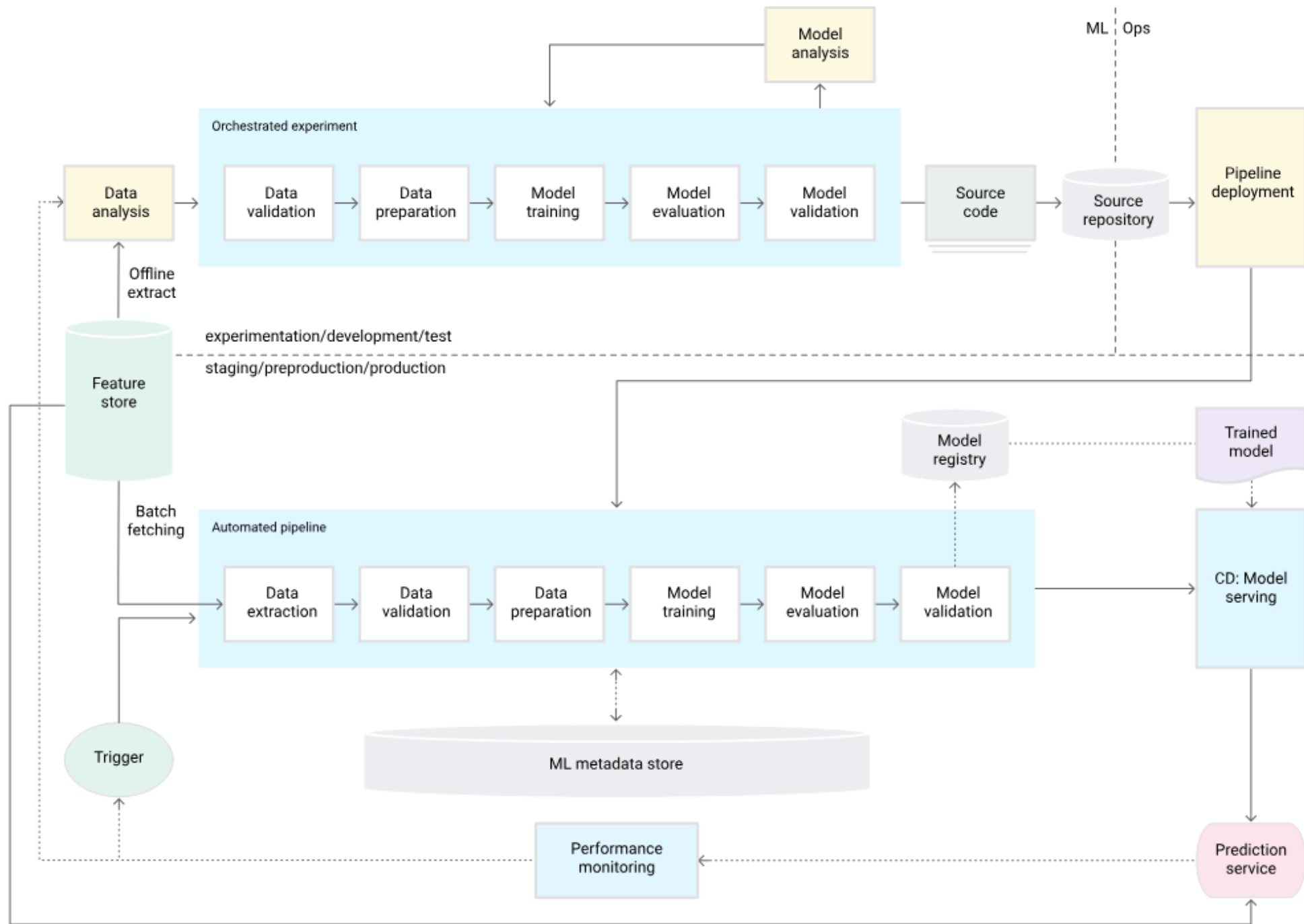


Automating ML  
pipelines enable  
fast experiments,  
continuous  
model updates,  
and reliable  
deployments

- **Rapid Experimentation**
  - **Automate pipeline** steps for quick iterations
  - Seamlessly transition **experiments to production**
- **Continuous Training**
  - **Automatically retrain** models with fresh data
  - **Triggered** by schedules, data availability, or performance
- **Experimental-Operational Symmetry**
  - Use **identical pipeline** in development and production
  - Ensures consistency and **simplifies management**
- **Modular Components**
  - **Decouple** code execution (EDA can still live in notebooks)
  - Reusable, isolated components **enhance reproducibility**
- **Continuous Model Delivery**
  - **Automatically deploy** updated models
  - **Regularly serve** improved prediction services

# Level 1

Enabling continuous training of the model by automating the ML pipeline



```
File Edit Selection View Go Run Terminal ... main_logging_examples
```

EXPLORER

- OPEN EDITORS
  - ml\_pipe\_sklearn.py ML\_pipeline... 9+
- MAIN\_LOGGING\_EXAMPLES
  - \_\_pycache\_\_
  - .pytest\_cache
  - Examples
  - logs
  - ML\_pipelines\_sklearn
    - conda.yml
    - ml\_pipe\_sklearn.py 9+
    - model\_separated.pkl
    - preprocessing\_pipeline.pkl
  - src
  - tests
  - .env
  - .gitignore
  - 0. main.py
  - environment.yml
  - fail\_pattern\_demo.py
  - logging\_config.yaml
  - logging\_demo.py
  - pytest.ini
  - test\_monkeypatch.py
- OUTLINE
- TIMELINE

ml\_pipe\_sklearn.py 9+ X

ML\_pipelines\_sklearn > ml\_pipe\_sklearn.py > ...

```
Run Cell | Run Below | Debug Cell | You, 2 hours ago | 1 author (You)
# %% Import libraries ----- You, 3 hours ago * Add pipeline basic example ...
1 from sklearn.base import BaseEstimator, TransformerMixin
2 import pandas as pd
3
4 from sklearn.datasets import load_iris
5 from sklearn.model_selection import train_test_split
6 from sklearn.linear_model import LogisticRegression
7 from sklearn.preprocessing import StandardScaler
8 from sklearn.pipeline import Pipeline
9 import pickle
10
Run Cell | Run Above | Debug Cell
# %% Load dataset -----
11 data = load_iris()
12 df = pd.DataFrame(data.data, columns=data.feature_names)
13 df['target'] = data.target
14 print(df.head())
15
Run Cell | Run Above | Debug Cell
# %% Typical data splitting -----
16
17 X = df.drop('target', axis=1)
18 y = df['target']
19
20
21
22 X_train, X_test, y_train, y_test = train_test_split(
23     X, y, test_size=0.2, random_state=42)
24 print(X_train.shape, X_test.shape)
25
Run Cell | Run Above | Debug Cell
# %% Manual preprocessing -----
26
27 scaler = StandardScaler()
28 X_train_scaled = scaler.fit_transform(X_train)
29 X_test_scaled = scaler.transform(X_test)
30 print(X_train_scaled[:5])
31
Run Cell | Run Above | Debug Cell
# %% Manual model fitting -----
32
33 model = LogisticRegression()
34 model.fit(X_train_scaled, y_train)
35
36
37
```

scikit learn

main Launchpad 7 5 9 DVC (Auto) You, 3 hours ago Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Python Type Checking: basic 3.10.17 ('ml\_pipe': conda)

[https://github.com/2025-IE-MLOps-course/main\\_logging\\_examples](https://github.com/2025-IE-MLOps-course/main_logging_examples)



# Harnessing the combined power of MLflow & WandB for Experimentation and Lifecycle Management

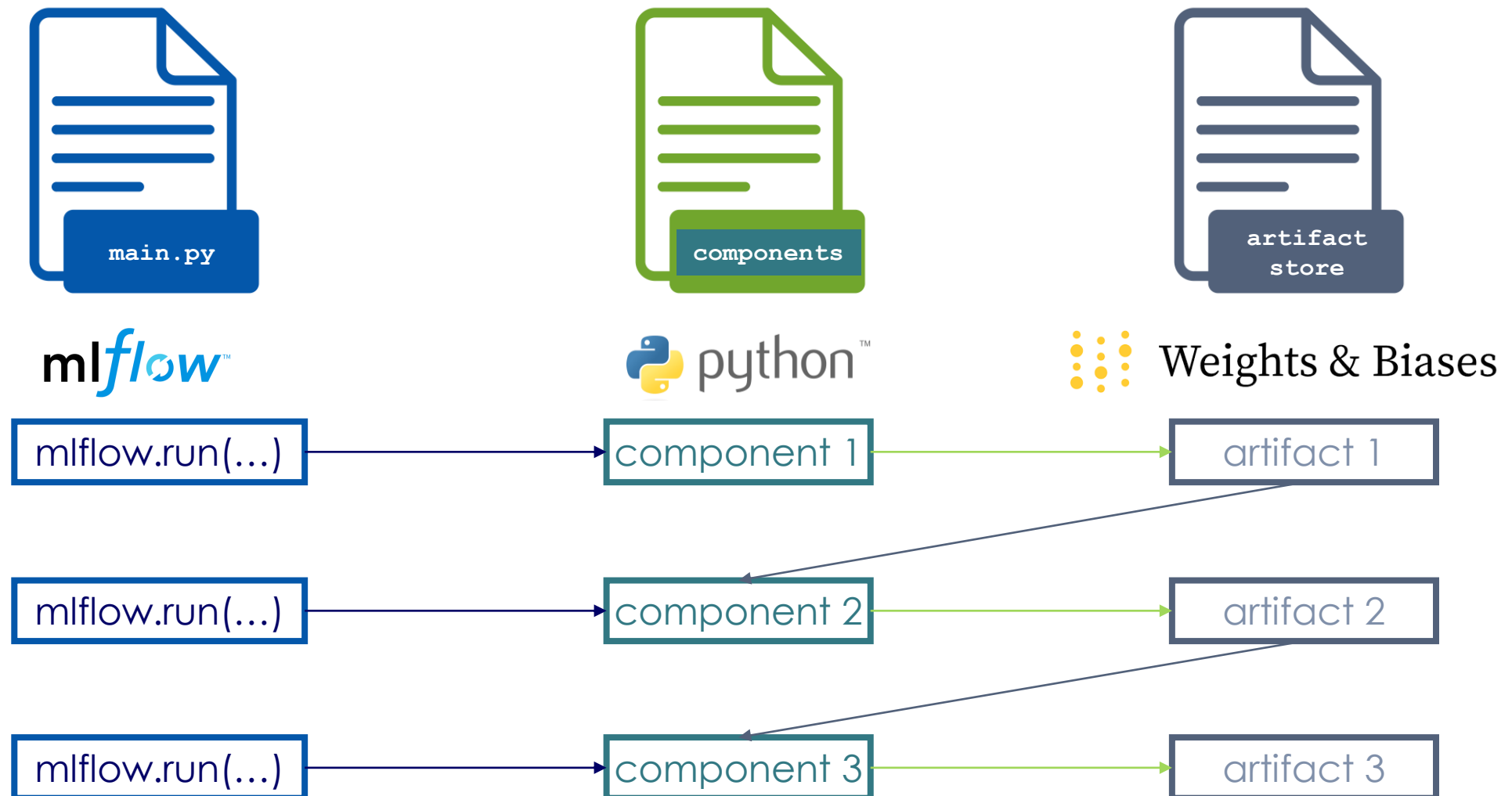


<b>Overview</b>	Open-source platform managing the end-to-end ML lifecycle	Experiment tracking, dataset versioning, and model evaluation
<b>Importance</b>	Facilitates reproducibility, collaboration, and model deployment	Real-time insights, collaboration, and reproducibility in ML projects
<b>Components</b>	Tracking, Projects, Models, and Registry	Experiments, Reports, Artifacts, Tables, Sweeps, Launch, Models...
<b>Business model</b>	Open-sourced	Managed Cloud (Free student)

# Integrating MLflow and WandB ensures your work is trackable, reproducible, and scalable

Dimension	Current (main.py only)	With MLflow	With WandB
Experiment Tracking	Manual, limited, error-prone	Automated, centralized, queryable	Best-in-class, collaborative, real-time
Pipeline Automation	Custom scripting, no standardization	Standardized, modular, reproducible	Still manual, focus on tracking, not flow
Monitoring	Basic logs, hard to compare runs	UI for metrics, basic monitoring	Advanced live metrics, alerts, dashboards
Visualization	Print/logs, no central dashboard	Simple UI, basic charts	Rich dashboards, interactive comparisons
Reproducibility	Depends on discipline, not enforced	Enforced via MLproject, Conda/Docker	Good with artifacts and configs
Collaboration	Manual sharing, hard to track changes	Easier, but limited UI	Team-focused, cloud-based collaboration
Model Registry	Manual versioning, error-prone	Built-in, production-ready registry	Artifacts system, suitable for most projects

# MLflow and WandB integration



# MLflow and WandB integration

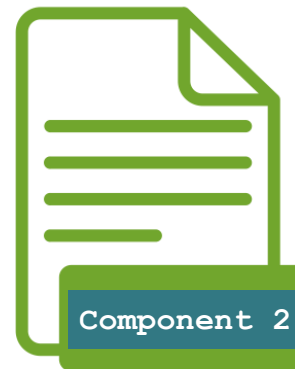
mlflow™ python™



- MLproject
- conda.yml
- config.yalm
- main.py



- MLproject
- conda.yml
- run.py



- MLproject
- conda.yml
- run.py

Argparse()



Weights & Biases

# Canonical Mflow directory

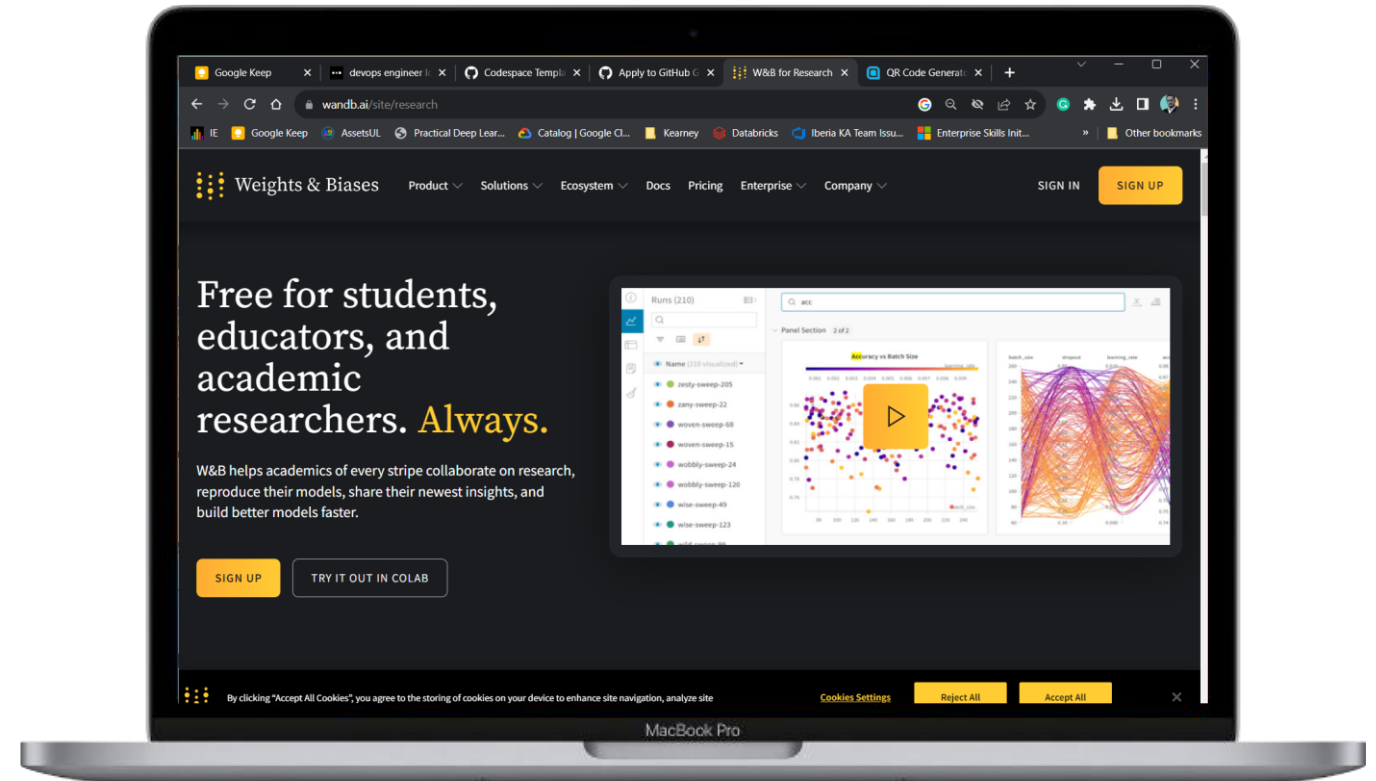
```
.
├── MLproject
├── main.py
├── environemnt.yml
├── conda.yml
└── src
    ├── basic_cleaning
    │   ├── MLproject
    │   ├── conda.yml
    │   └── run.py
    ├── data_check
    │   ├── MLproject
    │   ├── conda.yml
    │   └── run.py
    ├── eda
    │   ├── EDA.ipynb
    │   ├── MLproject
    │   └── conda.yml
    └── train_random_forest
        ├── MLproject
        ├── conda.yml
        └── run.py
```



# Next project steps (25/06)

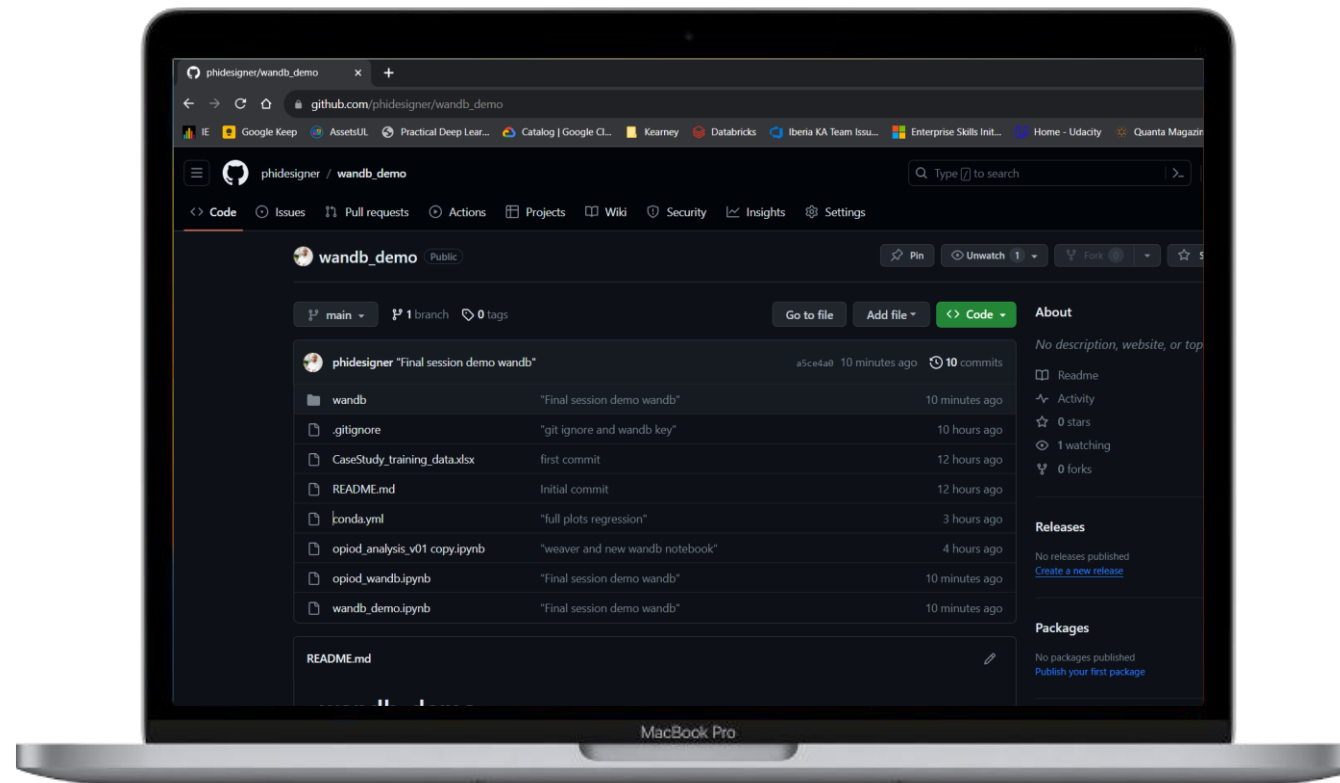
1. Refactor pipeline to use **MLflow and W&B**
2. Add/configure **Hydra** for flexible experimentation
3. Create and push tests; see CI results on **GitHub Actions**
4. Deploy API to <free server>; share the endpoint
5. Wrap model in **FastAPI**; test endpoint locally

# Weights and Biases



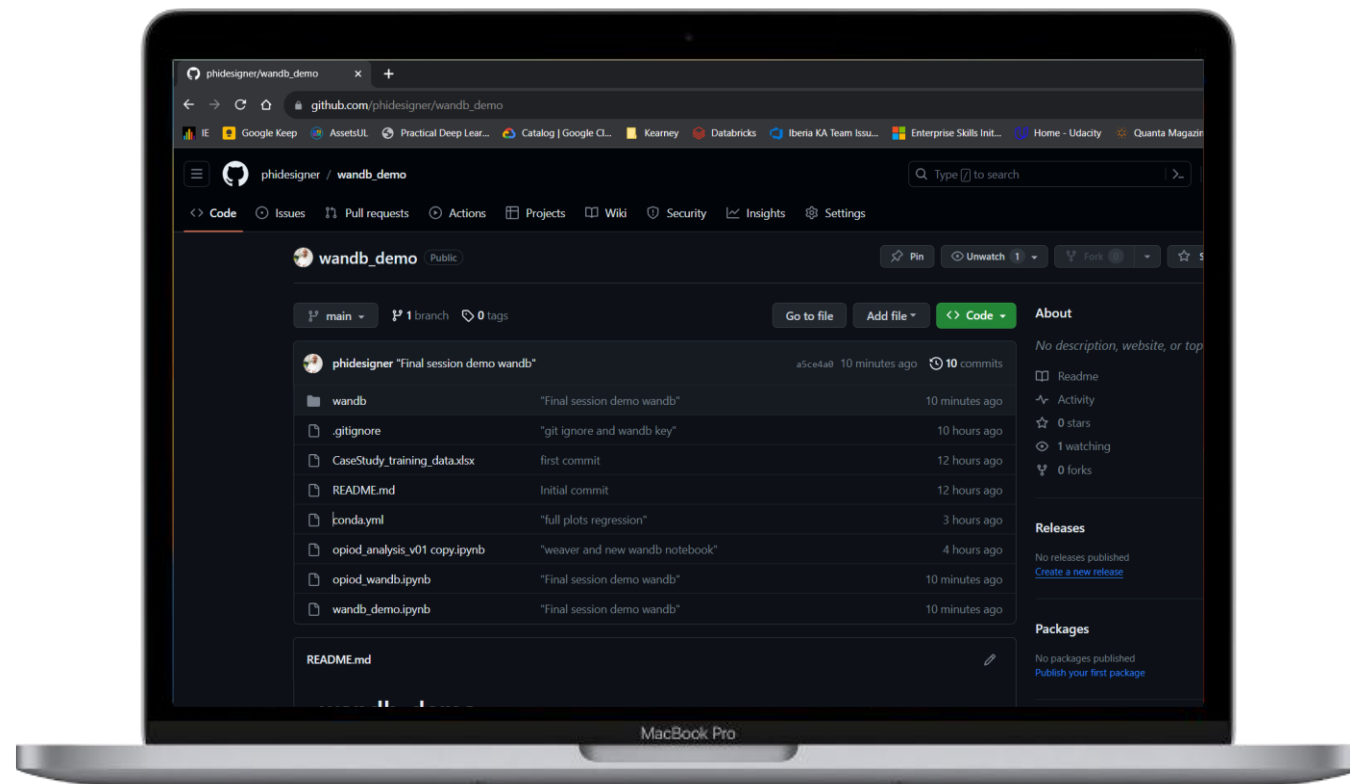
[WandB](https://wandb.ai)

# GitHub repo – WandB demo



[GitHub repo](#)

# GitHub repo – MLflow demo



[GitHub repo](#)

# Mlflow relevant commands

- `conda clean -all`
- `conda env remove -n mlflow-<hash>`
- `mlflow run .`
- `mlflow run . -P <arg name>=<“script”>`
- `mlflow run src/<module>`
- `mlflow ui`



# Mlflow commands in WSL

## # Install Ubuntu

> wsl --install

- # Install conda
- wget  
[https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86\\_64.sh](https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh)
- bash Miniconda3-latest-Linux-x86\_64.sh
- # Install MAMBA
- conda install -n base -c conda-forge mamba
- conda config --set solver libmamba
- # Set MAMBA as solver
- echo 'export  
MLFLOW\_CONDA\_CREATE\_ENV\_CMD=mamba' >>  
~/.bashrc
- source ~/.bashrc

```
File Edit Selection View Go Run Terminal ... main_logging_examples
```

EXPLORER

- OPEN EDITORS
  - ml\_pipe\_sklearn.py ML\_pipeline... 9+
- MAIN\_LOGGING\_EXAMPLES
  - \_\_pycache\_\_
  - .pytest\_cache
  - Examples
  - logs
  - ML\_pipelines\_sklearn
    - conda.yml
    - ml\_pipe\_sklearn.py 9+
    - model\_separated.pkl
    - preprocessing\_pipeline.pkl
  - src
  - tests
  - .env
  - .gitignore
  - 0. main.py
  - environment.yml
  - fail\_pattern\_demo.py
  - logging\_config.yaml
  - logging\_demo.py
  - pytest.ini
  - test\_monkeypatch.py
- OUTLINE
- TIMELINE

ml\_pipe\_sklearn.py 9+ X

ML\_pipelines\_sklearn > ml\_pipe\_sklearn.py > ...

```
Run Cell | Run Below | Debug Cell | You, 2 hours ago | 1 author (You)
# %% Import libraries ----- You, 3 hours ago • Add pipeline basic example ...
1 from sklearn.base import BaseEstimator, TransformerMixin
2 import pandas as pd
3
4 from sklearn.datasets import load_iris
5 from sklearn.model_selection import train_test_split
6 from sklearn.linear_model import LogisticRegression
7 from sklearn.preprocessing import StandardScaler
8 from sklearn.pipeline import Pipeline
9 import pickle
10
Run Cell | Run Above | Debug Cell
# %% Load dataset -----
11 data = load_iris()
12 df = pd.DataFrame(data.data, columns=data.feature_names)
13 df['target'] = data.target
14 print(df.head())
15
Run Cell | Run Above | Debug Cell
# %% Typical data splitting -----
16
17 X = df.drop('target', axis=1)
18 y = df['target']
19
20
21 X_train, X_test, y_train, y_test = train_test_split(
22     X, y, test_size=0.2, random_state=42)
23 print(X_train.shape, X_test.shape)
24
25
Run Cell | Run Above | Debug Cell
# %% Manual preprocessing -----
26
27 scaler = StandardScaler()
28 X_train_scaled = scaler.fit_transform(X_train)
29 X_test_scaled = scaler.transform(X_test)
30 print(X_train_scaled[:5])
31
32
Run Cell | Run Above | Debug Cell
# %% Manual model fitting -----
33
34 model = LogisticRegression()
35 model.fit(X_train_scaled, y_train)
36
37
```

scikit learn

main Launchpad 7 5 9 DVC (Auto) You, 3 hours ago Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Python Type Checking: basic 3.10.17 ('ml\_pipe': conda)

[https://github.com/2025-IE-MLOps-course/main\\_logging\\_examples](https://github.com/2025-IE-MLOps-course/main_logging_examples)