Basic_mlflow

MACHINE LEARNING OPERATIONS

UNLIMITED

Presented by Asst. Prof. Dr. Tuchsanai Ploysuwan

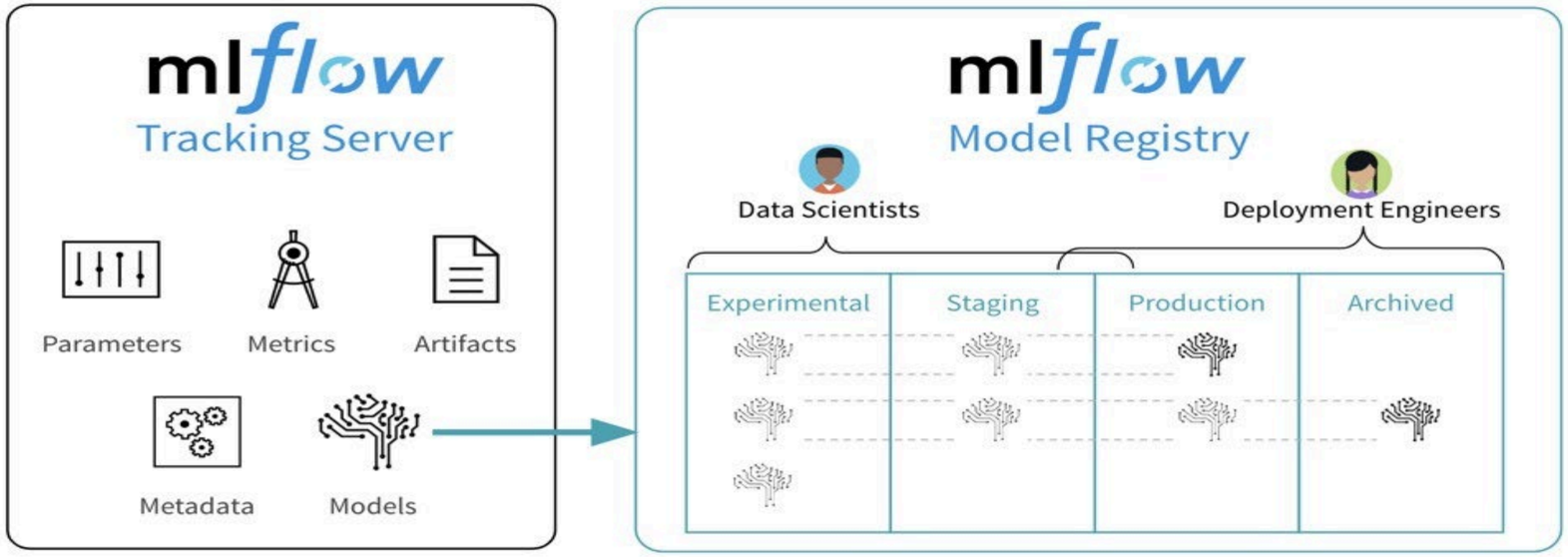
🔍 MLflow คืออะไร?

MLflow เป็น Open Source Platform สำหรับจัดการ Machine Learning Lifecycle ทั้งหมด พัฒนาโดย Databricks และเปิดตัวครั้งแรกในปี 2018 ปัจจุบันเป็นหนึ่งในเครื่องมือ MLOps ที่ได้รับความนิยมมากที่สุด โดยมีจุดเด่นคือความยืดหยุ่นสูง รองรับหลายภาษาโปรแกรม (Python, R, Java, REST API) และสามารถทำงานร่วมกับ ML Framework ยอดนิยมได้ทุกตัว

ปัญหาที่ MLflow แก้ไข

ในการพัฒนา Machine Learning แบบดั้งเดิม Data Scientists มักประสบปัญหาหลายประการ:

1. **Reproducibility Crisis**: ไม่สามารถทำซ้ำผลการทดลองได้เพราะลืมว่าใช้ hyperparameters อะไร ใช้ data version ไหน หรือใช้ code version ใด
2. **Experiment Chaos**: มี notebook หลายสิบไฟล์ที่ชื่อคล้ายกัน เช่น `model_v1.ipynb`, `model_v2_final.ipynb`, `model_v2_final_real.ipynb` จนไม่รู้ว่าอันไหนดีที่สุด
3. **Collaboration Difficulty**: ทีมงานไม่สามารถแชร์และเปรียบเทียบผลการทดลองได้อย่างมีประสิทธิภาพ
4. **Deployment Gap**: การนำ Model ไป Deploy ต้องเขียน code ใหม่เกือบทั้งหมด



mlflow TRACKING	mlflow PROJECTS	mlflow MODEL REGISTRY	mlflow MODELS
Record and query experiments: code, data, config, and results.	Package data science code in a format that enables reproducible runs on many platforms	Store, annotate, and manage models in a central repository	Deploy machine learning models in diverse serving environments

องค์ประกอบหลัก 4 ส่วนของ MLflow Platform

MLflow ประกอบด้วย 4 Components ที่ทำงานร่วมกันครอบคลุม ML Lifecycle ทั้งหมด:

📋 **ตารางสรุป MLflow Components**

Component	Icon	Tagline	หน้าที่หลัก	ใช้เมื่อไหร่?
Tracking	📊	"บันทึกทุกการทดลอง"	บันทึก Parameters, Metrics, Artifacts	ตอน Train/Experiment
Projects	📦	"รันที่ไหนก็ได้ผลเหมือนกัน"	Package code ให้ reproduce ได้	ตอนแชร์ code กับทีม
Models	🤖	"Train ครั้งเดียว Deploy ได้ทุกที่"	Package model ให้ deploy ได้หลายที่	ตอน Deploy model
Registry	📋	"จัดการ Model เหมือน Git"	Version control สำหรับ models	ตอนจัดการ Production

หัวข้อ	📊 Tracking	📦 Projects	🤖 Models	📋 Registry
บันทึกอะไร?	Parameters, Metrics, Artifacts	Code, Dependencies, Entry points	Trained model, Model format	Model versions, Stages
Output	Runs, Experiments	Reproducible package	Deployable model	Versioned model store
ไฟล์สำคัญ	`mlruns/` folder	`MLproject`, `conda.yaml`	`MLmodel`, `model.pkl`	Registry database
คำสั่งหลัก	`mlflow.log_*()`	`mlflow run`	`mlflow.*.log_model()`	`mlflow.register_model()`
UI	Experiment UI	-	Model UI	Registry UI

สถานการณ์	Component ที่ใช้	เหตุผล
กำลังทดลอง hyperparameters หลายค่า	📊 **Tracking**	บันทึกและเปรียบเทียบผลได้
ต้องการให้เพื่อนร่วมทีมรัน code เดียวกัน	📦 **Projects**	มี dependencies ครบ รันได้ผลเหมือนกัน
จะนำ model ไป deploy บน server	🤖 **Models**	Package model ให้พร้อม serve
มี model หลาย versions ต้องจัดการ	📋 **Registry**	ติดตาม version, จัดการ Production
ทำทุกอย่างครบวงจร	**ใช้ทั้ง 4**	ครอบคลุม ML Lifecycle

💡 **ตัวอย่าง Use Case จริง**

ขั้นตอน	สิ่งที่ทำ	Component
1	Data Scientist ทดลอง model หลายแบบ บันทึก accuracy แต่ละแบบ	📊 Tracking
2	เลือก model ที่ดีที่สุด แพ็คโค้ดให้ทีม ML Engineer	📦 Projects
3	ML Engineer นำ model ไป deploy เป็น REST API	🤖 Models
4	เมื่อมี model ใหม่ที่ดีกว่า เปลี่ยน Production version	📋 Registry

📊 รายละเอียด MLflow Tracking

📊 1. MLflow Tracking (เน้นใน Lab นี้)

> "บันทึกทุกการทดลอง เปรียบเทียบได้ทุกผลลัพธ์"

MLflow Tracking คือระบบบันทึกและติดตามการทดลอง Machine Learning ช่วยให้คุณไม่ต้องจำว่าใช้ hyperparameters อะไร ได้ผลเท่าไหร่

สิ่งที่บันทึกได้:

ประเภท	คำอธิบาย	ตัวอย่าง
Parameters	ค่าที่ตั้งก่อนทดลอง (Input)	`learning_rate=0.001`, `epochs=100`
Metrics	ค่าที่วัดได้จากการทดลอง (Output)	`accuracy=0.95`, `loss=0.05`
Artifacts	ไฟล์ที่สร้างจากการทดลอง	`model.pkl`, `confusion_matrix.png`
Tags	ข้อมูลเพิ่มเติมสำหรับจัดกลุ่ม	`developer=john`, `env=production`

ตัวอย่างการใช้งาน:

```python
import mlflow

mlflow.set_experiment("my-experiment")

with mlflow.start_run():
    mlflow.log_param("learning_rate", 0.001)    # บันทึก Parameter
    mlflow.log_metric("accuracy", 0.95)         # บันทึก Metric
    mlflow.log_artifact("model.pkl")            # บันทึก Artifact
```

📦 รายละเอียด MLflow Projects

📦 2. MLflow Projects

> "เขียนครั้งเดียว รันที่ไหนก็ได้ผลเหมือนกัน"

MLflow Projects คือ standard format สำหรับ packaging ML code ให้สามารถ reproduce ได้

องค์ประกอบหลัก:

```
my_project/
├── MLproject              # ไฟล์กำหนดค่า entry points
├── conda.yaml             # Dependencies (conda)
├── requirements.txt       # Dependencies (pip)
└── train.py               # โค้ดหลัก
```

ตัวอย่าง MLproject file:

```yaml
name: my_ml_project

conda_env: conda.yaml

entry_points:
  main:
    parameters:
      learning_rate: {type: float, default: 0.01}
      epochs: {type: int, default: 100}
    command: "python train.py --lr {learning_rate} --epochs {epochs}"
```

การรัน Project:

```bash
# รันจาก local
mlflow run ./my_project -P learning_rate=0.001

# รันจาก GitHub
mlflow run git@github.com:user/repo.git -P epochs=50
```

📦 รายละเอียด MLflow Models

📦 3. MLflow Models

> "Train ครั้งเดียว Deploy ได้ทุกที่"

MLflow Models คือ standard format สำหรับ packaging trained model ให้ deploy ได้หลาย platforms

Model Flavors ที่รองรับ:

Category	Flavors
Traditional ML	sklearn, xgboost, lightgbm, catboost
Deep Learning	pytorch, tensorflow, keras
Big Data	spark, h2o
Other	onnx, prophet, statsmodels, custom

Deployment Targets:

Target	คำอธิบาย
Local	รัน prediction บนเครื่อง
REST API	Deploy เป็น REST endpoint
Docker	Package เป็น Docker container
AWS SageMaker	Deploy บน Amazon SageMaker
Azure ML	Deploy บน Azure Machine Learning
Spark	รันบน Apache Spark cluster

ตัวอย่างการบันทึกและ Deploy Model:

```
# บันทึก Model
import mlflow.sklearn
mlflow.sklearn.log_model(model, "my_model")

# Deploy เป็น REST API
# mlflow models serve -m runs:/<run_id>/my_model -p 5000
```

📋 รายละเอียด MLflow Model Registry

> "จัดการ Model เหมือนจัดการ Code ด้วย Git"

Model Registry คือ centralized store สำหรับจัดการ model versioning และ lifecycle เปรียบเทียบได้กับ Git ที่ใช้จัดการ source code

---

🤔 ทำไมต้องมี Model Registry?

เปรียบเทียบ: ไม่มี vs มี Model Registry

คำถามที่พบบ่อย	😰 ไม่มี Model Registry	😊 มี Model Registry
"Model ที่รันบน Production คือไฟล์ไหน?"	ไม่รู้... น่าจะเป็น `model_final_v2_new.pkl` ?	`FraudDetector` version 3, stage: Production ✅
"Model นี้ train ด้วย hyperparameters อะไร?"	จำไม่ได้... ลองหาใน notebook ดู	คลิกดู lineage → Run `abc123`, lr=0.001, epochs=100 ✅
"Model เมื่อวานดีกว่า rollback ได้ไหม?"	ไม่ได้... ลบไฟล์เก่าไปแล้ว 😭	ได้เลย! เปลี่ยน version 2 เป็น Production ✅
"ใครเป็นคน deploy model นี้? เมื่อไหร่?"	ไม่รู้... ไม่มี log	`john@company.com`, 2024-01-15 14:30:00 ✅
"Model version ไหนดีที่สุด?"	ต้องเปิด notebook เทียบเอง	ดูใน Registry UI เปรียบเทียบ metrics ได้เลย ✅
"ทีม QA จะทดสอบ model ใหม่ได้ยังไง?"	ส่งไฟล์ให้ทาง Slack/Email	เปลี่ยน stage เป็น Staging แล้วแจ้งทีม ✅

🔄 เปรียบเทียบ Model Registry กับ Git

แนวคิด	Git (Code)	Model Registry (Model)
Repository	Git Repository	Registered Model
Version	Commit	Model Version
Branch/Tag	main, develop, v1.0	Production, Staging, Archived
History	Commit History	Version History
Rollback	`git checkout <commit>`	Transition to previous version
Collaboration	Pull Request	Stage Transition + Approval
Metadata	Commit Message	Description, Tags, Lineage

```
┌──────────────────────────────────────────────────────────────┐
│                    Git vs Model Registry                     │
├──────────────────────────────────────────────────────────────┤
│           Git                  │          Model Registry     │
├────────────────────────────────┼─────────────────────────────┤
│                                │                             │
│  Repository: my-app            │  Registered Model: FraudDetector │
│  ├── commit abc123 (v1.0)      │  ├── Version 1 (Archived)   │
│  ├── commit def456 (v1.1)      │  ├── Version 2 (Staging)    │
│  └── commit ghi789 (latest)    │  └── Version 3 (Production) ← Current │
│          │                     │          │                  │
│          ▼                     │          ▼                  │
│  Branches:                     │  Stages:                    │
│  • main (production)           │  • Production (ใช้งานจริง)  │
│  • develop (testing)           │  • Staging (ทดสอบ)          │
│  • feature/* (development)     │  • Archived (เก็บถาวร)      │
│                                │                             │
└────────────────────────────────┴─────────────────────────────┘
```

🎯 Model Stages อธิบายละเอียด

Model Registry มี 4 Stages สำหรับจัดการ lifecycle ของ model:

```
┌──────────────────────────────────────────────────────────────┐
│                     Model Lifecycle Stages                   │
│                                                              │
│  ┌──────────────┐                                            │
│  │    None      │  ← Model ที่เพิ่งลงทะเบียน ยังไม่ได้กำหนด stage │
│  │ (ค่าเริ่มต้น) │                                            │
│  └──────────────┘                                            │
│         │                                                    │
│         │ เริ่มทดสอบ                                         │
│         ▼                                                    │
│  ┌──────────────┐                                            │
│  │   Staging    │  ← Model กำลังถูกทดสอบ (QA, Integration Test) │
│  │  (ทดสอบ)     │    เช่น ทดสอบกับข้อมูลจริง, ทดสอบ performance │
│  └──────────────┘                                            │
│         │                                                    │
│         │ ผ่านการทดสอบ ✅                                    │
│         ▼                                                    │
│  ┌──────────────┐                                            │
│  │ Production   │  ← Model ที่ใช้งานจริง serve ให้ users     │
│  │ (ใช้งานจริง) │    ควรมีแค่ 1 version ต่อ 1 Registered Model │
│  └──────────────┘                                            │
│         │                                                    │
│         │ มี version ใหม่มาแทน / เลิกใช้                     │
│         ▼                                                    │
│  ┌──────────────┐                                            │
│  │   Archived   │  ← Model ที่เลิกใช้แล้ว เก็บไว้อ้างอิง/rollback │
│  │ (เก็บถาวร)   │    ไม่ลบทิ้ง เผื่อต้องการกลับมาใช้         │
│  └──────────────┘                                            │
│                                                              │
└──────────────────────────────────────────────────────────────┘
```

สรุป Stage แต่ละประเภท:

Stage	สัญลักษณ์	คำอธิบาย	ใครใช้?
None	🆕	เพิ่งลงทะเบียน ยังไม่พร้อม	Data Scientist (กำลังพัฒนา)
Staging	🧪	กำลังทดสอบ อาจมีหลาย versions	QA Team, ML Engineer
Production	🚀	ใช้งานจริง ควรมี 1 version	End Users, Applications
Archived	📦	เลิกใช้แล้ว เก็บไว้อ้างอิง	ไม่มีใครใช้ (backup)

📊 **ตัวอย่าง Workflow จริง**

สมมติทีมพัฒนา Fraud Detection Model:

```
┌──────────────────────────────────────────────────────────────────────┐
│                    Fraud Detection Model - Version History           │
├──────────────────────────────────────────────────────────────────────┤
│                                                                      │
│  📋 Registered Model: "FraudDetector"                                │
│                                                                      │
│  ┌──────────────────────────────────────────────────────────────┐   │
│  │ Version 1                                                    │   │
│  │ ├── Stage: Archived 📦                                       │   │
│  │ ├── Created: 2024-01-01 by john@company.com                  │   │
│  │ ├── Source: runs:/abc111/model                               │   │
│  │ ├── Metrics: accuracy=0.85, f1=0.82                          │   │
│  │ └── Note: "Baseline model with LogisticRegression"           │   │
│  └──────────────────────────────────────────────────────────────┘   │
│                            │                                         │
│                            │ ถูกแทนที่โดย Version 2                  │
│                            ▼                                         │
│  ┌──────────────────────────────────────────────────────────────┐   │
│  │ Version 2                                                    │   │
│  │ ├── Stage: Archived 📦                                       │   │
│  │ ├── Created: 2024-01-15 by jane@company.com                  │   │
│  │ ├── Source: runs:/abc222/model                               │   │
│  │ ├── Metrics: accuracy=0.91, f1=0.89                          │   │
│  │ └── Note: "Improved with RandomForest + feature engineering" │   │
│  └──────────────────────────────────────────────────────────────┘   │
│                            │                                         │
│                            │ ถูกแทนที่โดย Version 3                  │
│                            ▼                                         │
│  ┌──────────────────────────────────────────────────────────────┐   │
│  │ Version 3                                                 🚀 │   │
│  │ ├── Stage: Production                                        │   │
│  │ ├── Created: 2024-02-01 by jane@company.com                  │   │
│  │ ├── Source: runs:/abc333/model                               │   │
│  │ ├── Metrics: accuracy=0.95, f1=0.93                          │   │
│  │ └── Note: "XGBoost with hyperparameter tuning - BEST"        │   │
│  └──────────────────────────────────────────────────────────────┘   │
│                            │                                         │
│                            │ กำลังพัฒนา Version 4                    │
│                            ▼                                         │
│  ┌──────────────────────────────────────────────────────────────┐   │
│  │ Version 4                                                 🧪 │   │
│  │ ├── Stage: Staging                                           │   │
│  │ ├── Created: 2024-02-15 by john@company.com                  │   │
│  │ ├── Source: runs:/abc444/model                               │   │
│  │ ├── Metrics: accuracy=0.96, f1=0.94                          │   │
│  │ └── Note: "Testing Deep Learning approach"                   │   │
│  └──────────────────────────────────────────────────────────────┘   │
│                                                                      │
└──────────────────────────────────────────────────────────────────────┘
```

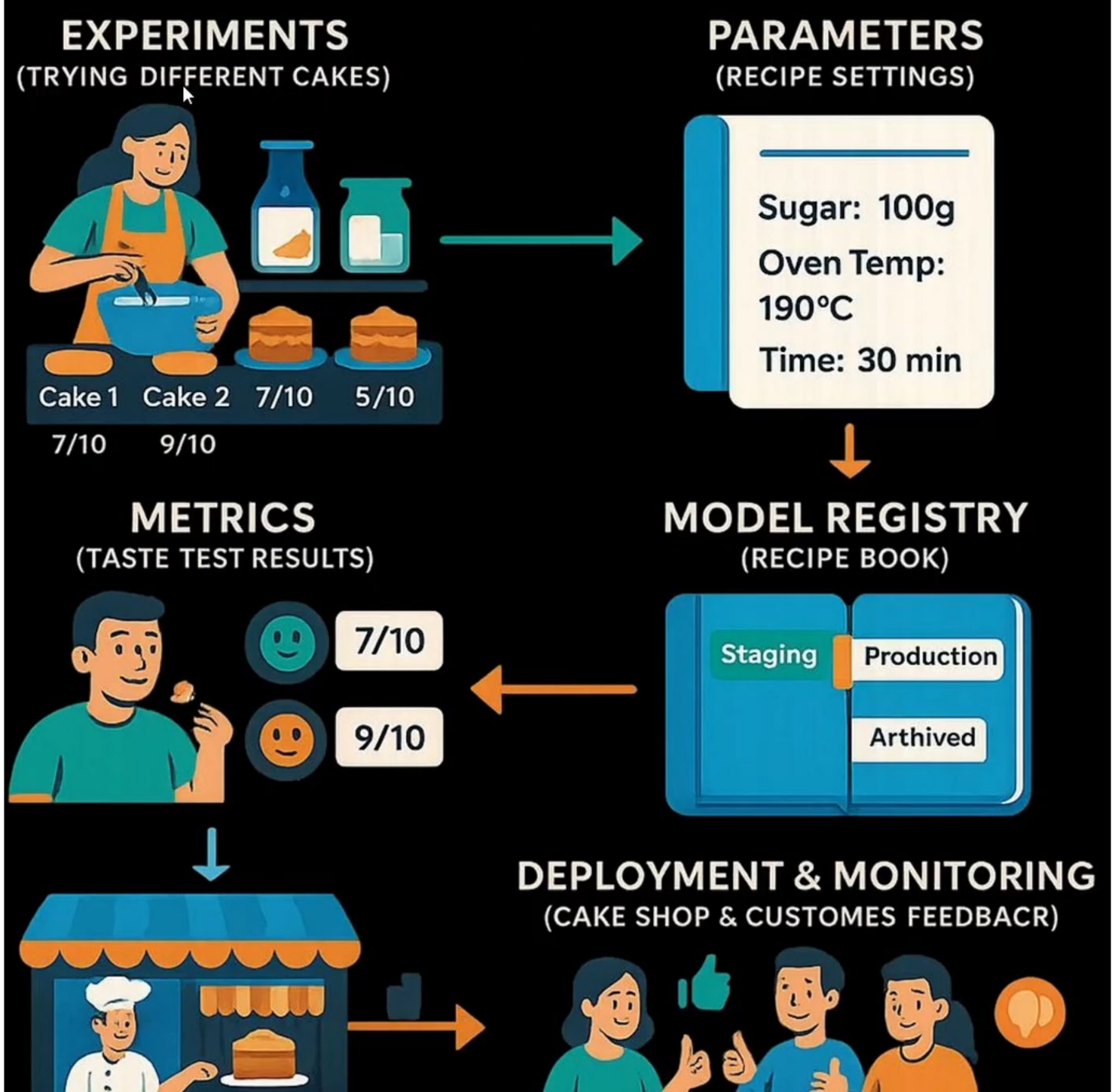
Workflow อธิบาย:

```
วันที่ 1: John สร้าง Version 1 (Baseline)
    → ลงทะเบียนเป็น FraudDetector Version 1
    → ทดสอบผ่าน → เปลี่ยนเป็น Production

วันที่ 15: Jane สร้าง Version 2 (ดีกว่า)
    → ลงทะเบียนเป็น FraudDetector Version 2
    → ทดสอบใน Staging
    → ผ่านการทดสอบ → เปลี่ยนเป็น Production
    → Version 1 ถูกเปลี่ยนเป็น Archived

วันที่ 30: Jane สร้าง Version 3 (ดีที่สุด)
    → ลงทะเบียน → ทดสอบ → Production
    → Version 2 → Archived

วันที่ 45: John กำลังทดสอบ Version 4
    → อยู่ใน Staging รอทดสอบ
    → ถ้าผ่าน จะขึ้น Production แทน Version 3
```

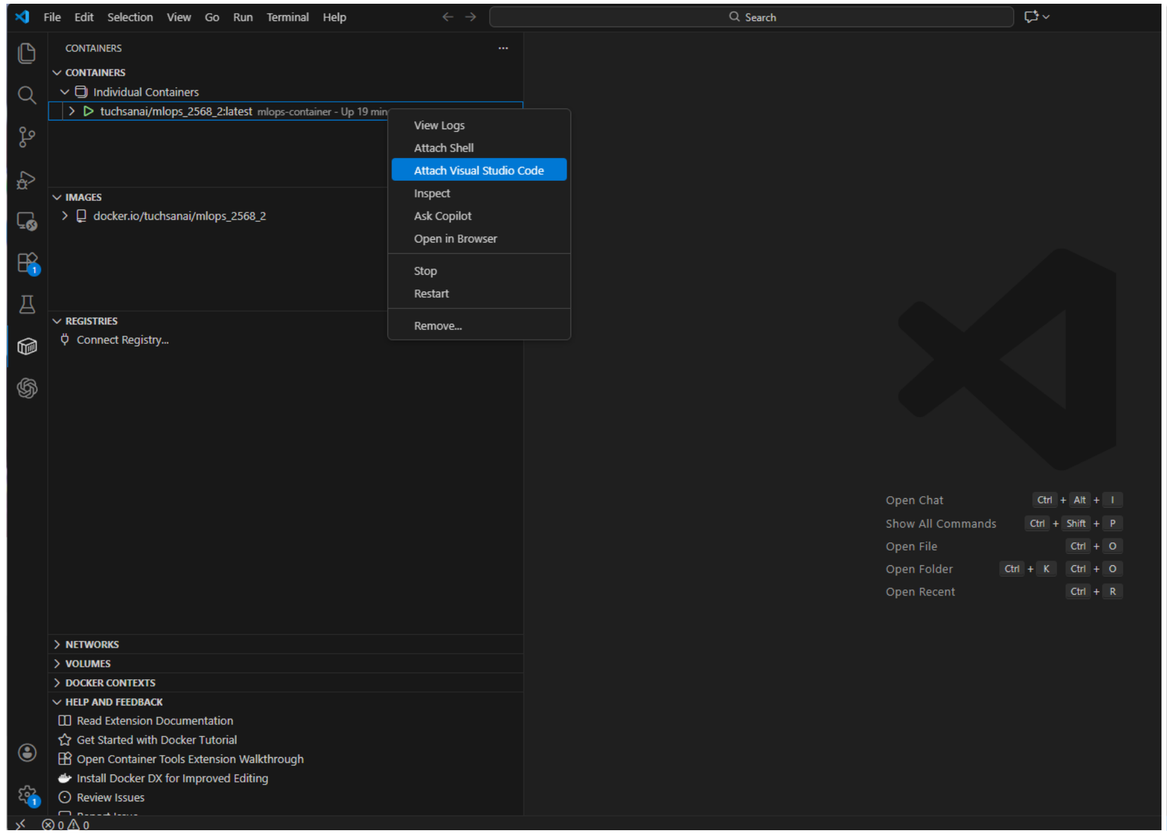

EXPERIMENTS
(TRYING DIFFERENT CAKES)
Cake 1
Cake 2
7/10
5/10
7/10
9/10
PARAMETERS
(RECIPE SETTINGS)
Sugar: 100g
Oven Temp: 190°C
Time: 30 min
MODEL REGISTRY
(RECIPE BOOK)
Staging
Production
Arthived
METRICS
(TASTE TEST RESULTS)
7/10
9/10
DEPLOYMENT & MONITORING
(CAKE SHOP & CUSTOMES FEEDBACR)

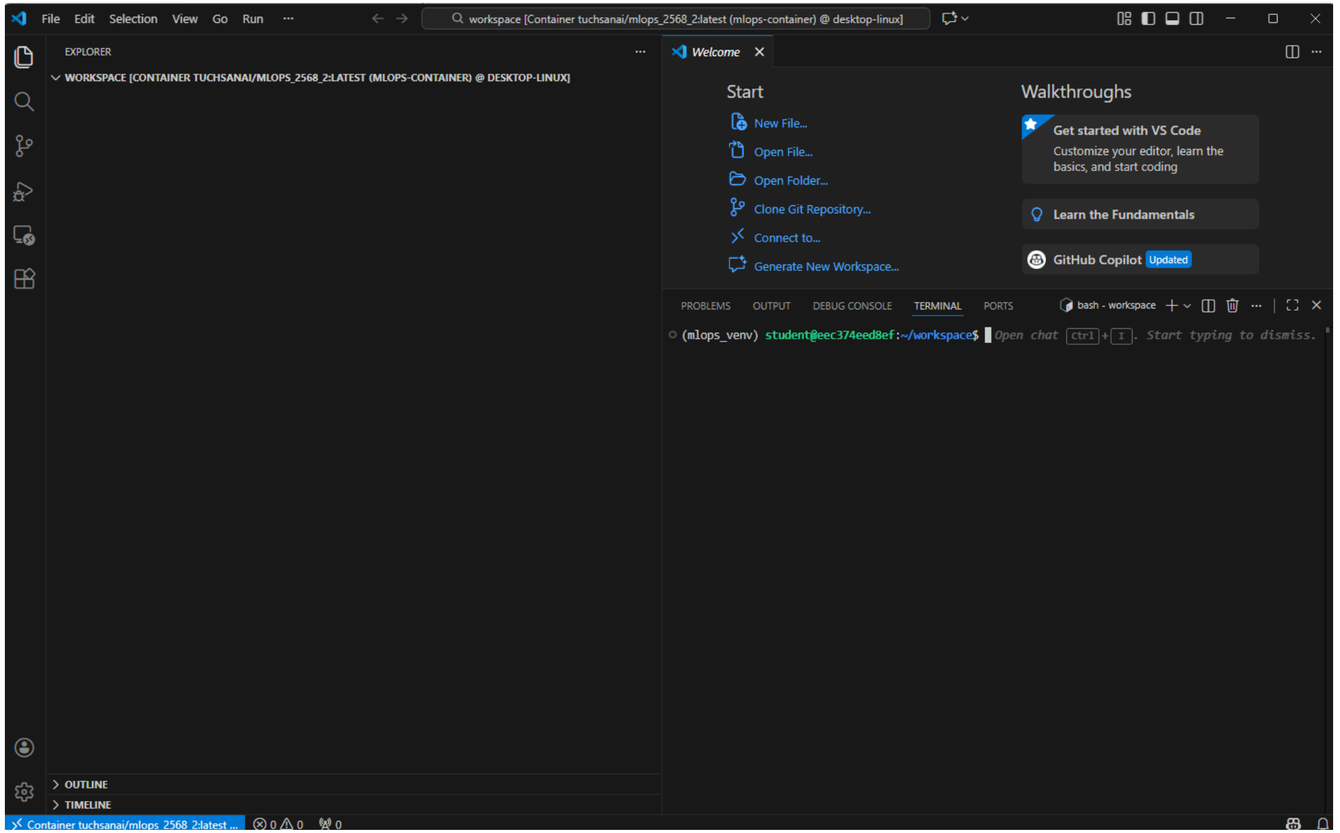
Model registration and versioning

Installing MLflow

All LAB Run in Dev Container

```
docker run -d  --name mlops-container tuchsanai/mlops_2568_2:latest
```





🚀 ขั้นตอนการทำ Lab

ขั้นตอนที่ 1: สร้างโฟลเดอร์สำหรับ Lab

```
# สร้างโฟลเดอร์สำหรับเก็บไฟล์ Lab
mkdir -p mlflowserver-lab

# เข้าไปในโฟลเดอร์
cd mlflowserver-lab
```

ขั้นตอนที่ 1: สร้าง Virtual Environment

> 💡 **ทำไมต้องใช้ Virtual Environment?**
>
> Virtual Environment ช่วยแยก Package ของแต่ละโปรเจกต์ออกจากกัน ป้องกันปัญหา Package Version ขัดแย้งกัน

```
# 1.1 สร้าง Virtual Environment ชื่อ .venv
python -m venv .venv

# 1.2 เปิดใช้งาน Virtual Environment
source .venv/bin/activate

# 1.3 อัพเดท pip ให้เป็นเวอร์ชันล่าสุด
python -m pip install --upgrade pip
```

✅ **ตรวจสอบ:** เมื่อเปิดใช้งานสำเร็จ จะเห็น `(.venv)` นำหน้า prompt

---

ขั้นตอนที่ 2: ติดตั้ง Package ที่จำเป็น

```
pip install mlflow[extras] scikit-learn pandas numpy
```

รายละเอียด Package:

Package	หน้าที่
`mlflow[extras]`	MLflow เวอร์ชันสมบูรณ์ รวม Model Serving, SQL Storage และ Dependencies ทั้งหมด
`scikit-learn`	Library สำหรับ Machine Learning
`pandas`	จัดการข้อมูลแบบ DataFrame
`numpy`	คำนวณทางคณิตศาสตร์

ขั้นตอนที่ 4: เปิดใช้งาน MLflow Tracking Server

> ⚠️ **สำคัญ:** ให้นักศึกษา **เลือกทำเพียงแบบเดียว** (แบบที่ 1 หรือ แบบที่ 2) ไม่ต้องทำทั้ง 2 แบบ

แบบ	ข้อดี	ข้อเสีย	เหมาะสำหรับ
แบบที่ 1: In-Memory	เริ่มต้นเร็ว ไม่ต้องตั้งค่า	ข้อมูลหายเมื่อปิด Server	ทดลองใช้งานเร็วๆ
แบบที่ 2: Persistent	ข้อมูลไม่หาย	ต้องสร้างโฟลเดอร์เพิ่ม	ใช้งานจริง ⭐

📌 แบบที่ 1: In-Memory (สำหรับทดลองใช้งานเร็ว)

```
mlflow server --host 127.0.0.1 --port 8080
```

ข้อดี: เริ่มต้นได้เร็ว ไม่ต้องตั้งค่าอะไรเพิ่ม
ข้อเสีย: ข้อมูลหายเมื่อปิด Server

ผลลัพธ์ที่คาดหวัง:

```
INFO:     Started server process [28550]
INFO:     Application startup complete.
Uvicorn running on http://127.0.0.1:8080 (Press CTRL+C to quit)
```

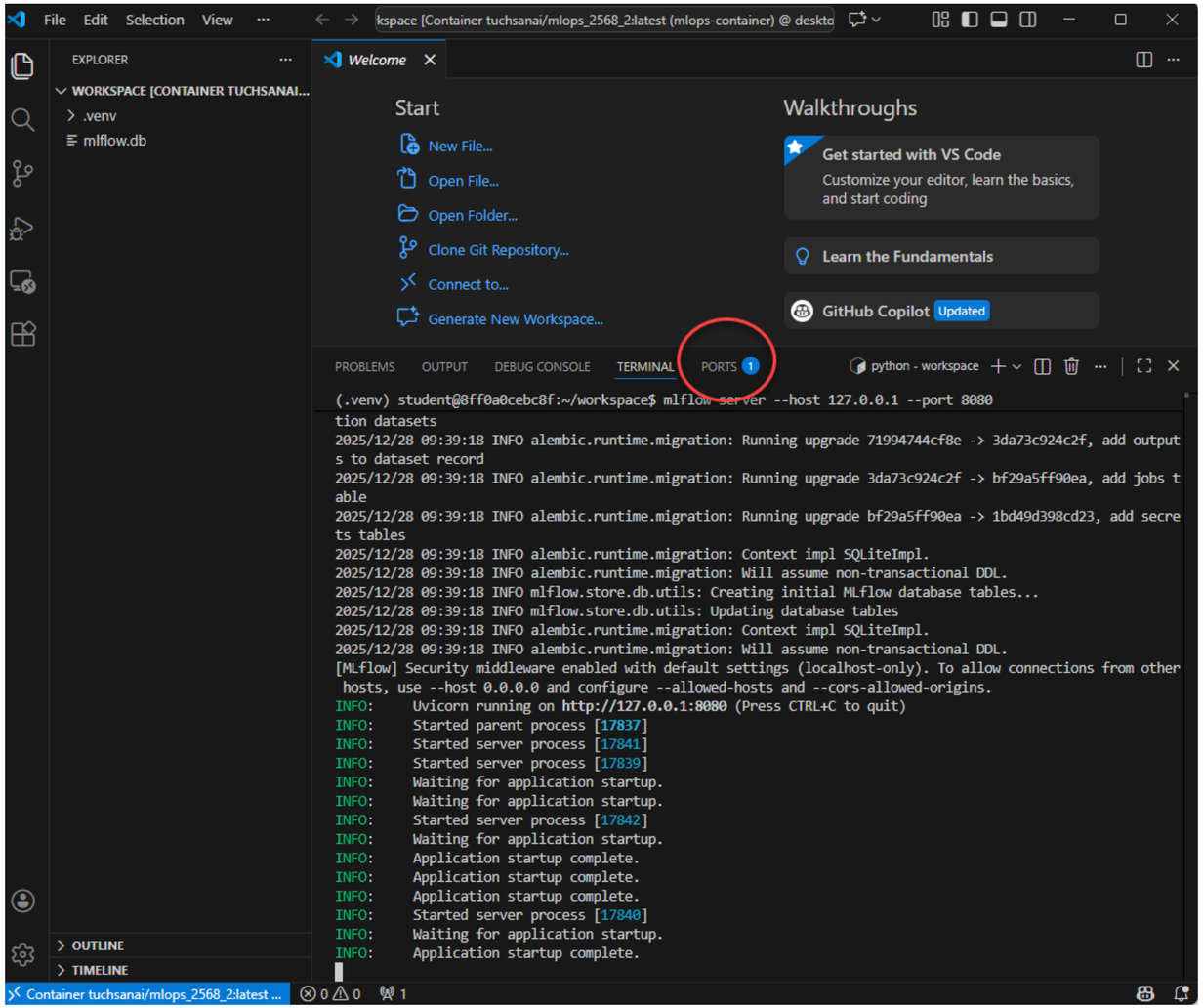
📌 แบบที่ 2: Persistent Storage (แนะนำสำหรับใช้งานจริง) ⭐

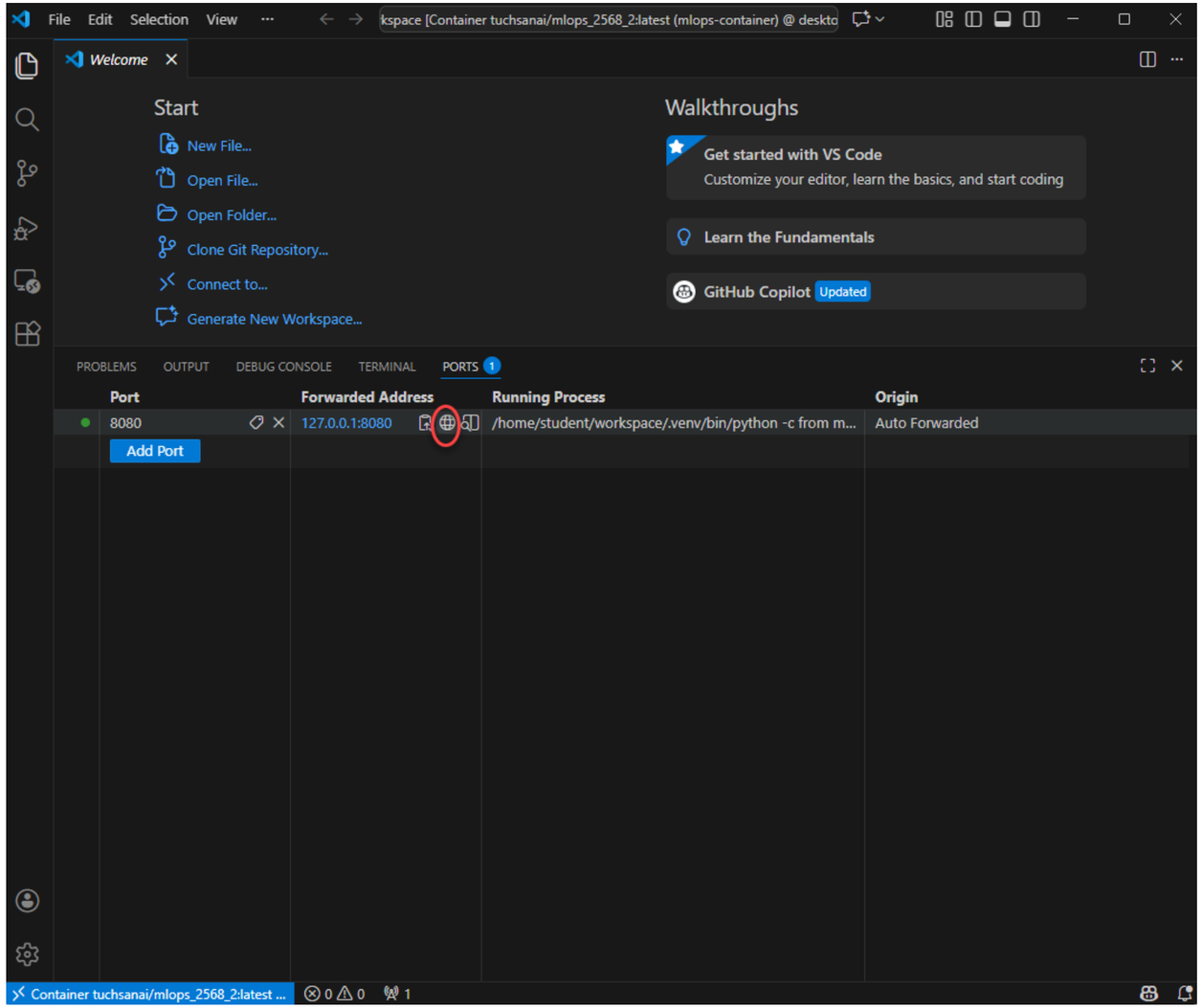
```
# 3.1 สร้างโฟลเดอร์เก็บข้อมูล
mkdir -p mlruns_db mlartifacts

# 3.2 เปิด Server พร้อม SQLite Backend
mlflow server \
  --host 127.0.0.1 --port 8080 \
  --backend-store-uri sqlite:///mlruns_db/mlflow.db \
  --artifacts-destination ./mlartifacts \
  --serve-artifacts
```

🗂️ อธิบายโครงสร้างไฟล์:

```
โปรเจกต์ของคุณ/
├── mlruns_db/
│   └── mlflow.db          # ฐานข้อมูล SQLite เก็บ Experiment และ Run Metadata
├── mlartifacts/           # โฟลเดอร์เก็บ Model Files และ Artifacts
└── .venv/                 # Virtual Environment
```


File Edit Selection View
kspace [Container tuchsanai/mlops_2568_2:latest (mlops-container) @ deskto
EXPLORER
WORKSPACE [CONTAINER TUCHSANAI...
.venv
mlflow.db
Welcome
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Generate New Workspace...
Walkthroughs
Get started with VS Code
Customize your editor, learn the basics, and start coding
Learn the Fundamentals
GitHub Copilot Updated
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1
python - workspace
(.venv) student@8ff0a0cebc8f:~/workspace$ mlflow server --host 127.0.0.1 --port 8080
tion datasets
2025/12/28 09:39:18 INFO alembic.runtime.migration: Running upgrade 71994744cf8e -> 3da73c924c2f, add output
s to dataset record
2025/12/28 09:39:18 INFO alembic.runtime.migration: Running upgrade 3da73c924c2f -> bf29a5ff90ea, add jobs t
able
2025/12/28 09:39:18 INFO alembic.runtime.migration: Running upgrade bf29a5ff90ea -> 1bd49d398cd23, add secre
ts tables
2025/12/28 09:39:18 INFO alembic.runtime.migration: Context impl SQLiteImpl.
2025/12/28 09:39:18 INFO alembic.runtime.migration: Will assume non-transactional DDL.
2025/12/28 09:39:18 INFO mlflow.store.db.utils: Creating initial MLflow database tables...
2025/12/28 09:39:18 INFO mlflow.store.db.utils: Updating database tables
2025/12/28 09:39:18 INFO alembic.runtime.migration: Context impl SQLiteImpl.
2025/12/28 09:39:18 INFO alembic.runtime.migration: Will assume non-transactional DDL.
[MLflow] Security middleware enabled with default settings (localhost-only). To allow connections from other
hosts, use --host 0.0.0.0 and configure --allowed-hosts and --cors-allowed-origins.
INFO: Uvicorn running on http://127.0.0.1:8080 (Press CTRL+C to quit)
INFO: Started parent process [17837]
INFO: Started server process [17841]
INFO: Started server process [17839]
INFO: Waiting for application startup.
INFO: Waiting for application startup.
INFO: Started server process [17842]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Application startup complete.
INFO: Application startup complete.
INFO: Started server process [17840]
INFO: Waiting for application startup.
INFO: Application startup complete.
OUTLINE
TIMELINE
Container tuchsanai/mlops_2568_2:latest ...


File Edit Selection View
kspace [Container tuchsanai/mlops_2568_2:latest (mlops-container) @ deskto
Welcome
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Generate New Workspace...
Walkthroughs
Get started with VS Code
Customize your editor, learn the basics, and start coding
Learn the Fundamentals
GitHub Copilot Updated
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1
Port
Forwarded Address
Running Process
Origin
8080
127.0.0.1:8080
/home/student/workspace/.venv/bin/python -c from m...
Auto Forwarded
Add Port
Container tuchsanai/mlops_2568_2:latest ...

127.0.0.1:8080

mlflow 3.8.1

GitHub Docs

+ New

- Home
- Experiments
- Models
- Prompts

Welcome to MLflow

Information about UI telemetry
MLflow collects usage data to improve the product. To confirm your preferences, please visit the settings page in the navigation sidebar. To learn more about what data is collected, please visit the documentation.

Get started

Log traces
Trace LLM applications for debugging and monitoring.

Run evaluation
Iterate on quality with offline evaluations and comparisons.

Train models
Track experiments, parameters, and metrics throughout training.

Register prompts
Manage prompt updates and collaborate across teams.

Experiments

View all

Name	Time created	Last modified	Description	Tags
Default	12/28/2025, 04:39:18 PM	12/28/2025, 04:39:18 PM	-	

Discover new features

View all

MLflow MCP server
Connect your coding assistants and AI applications to MLflow and automatically analyze your experiments and traces.

Optimize prompts
Access the state-of-the-art prompt optimization algorithms such as MIPROv2, GEPA, through MLflow Prompt Registry.

Agents-as-a-judge
Leverage agents as a judge to perform deep trace analysis and improve your evaluation accuracy.

Dataset tracking
Track dataset lineage and versions and effectively drive the quality improvement loop.

notebook_v1
Logistic Reg
C = 1
solver = 'lbfgs"
Dataset
V1
notebook_v2
Logistic Reg
C = 0.1
solver = 'saga"
Dataset
V1
notebook_v3
XGBoost
max_depth = 3
solver = 'saga"
Dataset
V1
notebook_v4
XGBoost
max_depth = 3
solver = 'saga"
Dataset
V2
Additional feature f10
Kathy

notebook_v5

Random Forest
criterion = ‘gini’

Dataset
V10

notebook_v6

Random Forest
criterion =
‘entropy’

Dataset
V10

notebook_v7

XGBoost
max_depth = 4
learning_rate=0.01

Dataset
V10

notebook_v8

XGBoost
max_depth = 3
learning_rate=0.01

Dataset
V20

Additional feature f10

Venkat

Run_id	Date	User	Model	Params	Dataset	F1 Score ('macro')	Recall (Class=1)	Accuracy
1	1-Jul-24	Kathy	Logistic Reg	C=1, Solver='lbfgs'	V1	0.75	0.62	0.71
2	2-Jul-24	Kathy	Logistic Reg	C=1, Solver='saga'	V1	0.78	0.58	0.62
3	6-Jul-24	Kathy	XGBoost	max_depth=3,solver='saga'	V1	0.82	0.72	0.75
4	9-Jul-24	Kathy	XGBoost	max_depth=3,solver='saga'	V2	0.87	0.74	0.77
5	28-Jun-24	Venkat	Random Forest	criterion='gini'	V10	0.71	0.87	0.82
6	2-Jul-24	Venkat	Random Forest	criterion='entropy'	V10	0.67	0.84	0.81
7	10-Jul-24	Venkat	XGBoost	max_depth=4, lr=0.01	V10	0.83	0.92	0.79
8	11-Jul-24	Venkat	XGBoost	max_depth=3, lr=0.01	V20	0.82	0.89	0.84

Here's a simple command to run MLflow with its tracking server:

create directory

```
mkdir -p MLflow  # -p flag ensures no error if directory already exists
cd MLflow
```

Run MLflow UI with Docker

```
docker run -d \
  -p 5000:5000 \
  -v "$(pwd):/mlflow" \
  ghcr.io/mlflow/mlflow:latest \
  mlflow ui \
  --backend-store-uri file:///mlflow \
  --host 0.0.0.0
```

Verify It Works:

After running, check

```
http://ip_address:5000
```

34.126.111.16:5000

mlflow 2.20.3 Experiments Models GitHub Docs

Experiments

Search Experiments

Default

Default Provide Feedback Add Description Share

Runs Evaluation Experimental Traces

metrics.rmse < 1 and params.model = "tree" Time created State: Active New run

Datasets Sort: Created Columns Group by

Run Name	Created	Dataset	Duration	Source	Models

No runs logged

No runs have been logged yet. Learn more about how to create ML model training runs in this experiment.

Model registration and versioning