Ml flow

1. `.env` → stores credentials and configs safely.
2. `Dockerfile.mlflow` → builds MLflow image with required libs.
3. `docker-compose.yml` → orchestrates Postgres, MinIO, bucket creation, and MLflow.

1️⃣ .env

Put this file in your project root (`~/mlflow-minio/.env`):

```
None
# Postgres
PG_USER=mlflow
PG_PASSWORD=mlflow
PG_DATABASE=mlflow
PG_PORT=5432

# MinIO
MINIO_ROOT_USER=minio
MINIO_ROOT_PASSWORD=minio123
MINIO_PORT=9000
MINIO_CONSOLE_PORT=9001
MLFLOW_BUCKET_NAME=mlflow

# MLflow
MLFLOW_PORT=5000
```

2️⃣ Dockerfile.mlflow

This goes in the root (same place as docker-compose.yml):

```
None
FROM python:3.10-slim

# Install system dependencies for psycopg2
RUN apt-get update && apt-get install -y --no-install-recommends
\
    gcc libpq-dev curl && \
    rm -rf /var/lib/apt/lists/*

# Install MLflow and required libraries
RUN pip install --no-cache-dir mlflow psycopg2-binary boto3

# Default command (overridden in docker-compose.yml)
CMD ["mlflow", "server"]
```

3️⃣ docker-compose.yml

This is the full setup with **healthchecks + bucket auto-creation + networks**:

```
None
version: "3.9"

services:
  postgres:
    image: postgres:15
    container_name: mlflow_postgres
    restart: always
    environment:
      POSTGRES_USER: ${PG_USER}
      POSTGRES_PASSWORD: ${PG_PASSWORD}
      POSTGRES_DB: ${PG_DATABASE}
    volumes:
      - ./db_data:/var/lib/postgresql/data
    ports:
```

```
      - "5432:5432"
    networks:
      - backend
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U ${PG_USER}"]
      interval: 5s
      retries: 5

  minio:
    image: minio/minio:latest
    container_name: mlflow_minio
    restart: always
    command: server /data --console-address ":9001"
    environment:
      MINIO_ROOT_USER: ${MINIO_ROOT_USER}
      MINIO_ROOT_PASSWORD: ${MINIO_ROOT_PASSWORD}
    volumes:
      - ./minio_data:/data
    ports:
      - "${MINIO_PORT}:9000"
      - "${MINIO_CONSOLE_PORT}:9001"
    networks:
      - frontend
      - backend
    healthcheck:
      test: ["CMD", "curl", "-f",
"http://localhost:9000/minio/health/live"]
      interval: 30s
      retries: 5

  create_buckets:
    image: minio/mc
    container_name: mlflow_create_buckets
    depends_on:
      - minio
    entrypoint: >
```

```
      /bin/sh -c '
      sleep 10;
      mc alias set s3 http://minio:9000 ${MINIO_ROOT_USER}
${MINIO_ROOT_PASSWORD};
      mc mb s3/${MLFLOW_BUCKET_NAME} || echo "Bucket exists";
      mc policy set download s3/${MLFLOW_BUCKET_NAME};
      exit 0;'
    networks:
      - backend

  mlflow:
    build:
      context: .
      dockerfile: Dockerfile.mlflow
    container_name: mlflow_server
    restart: always
    depends_on:
      - postgres
      - minio
      - create_buckets
    ports:
      - "${MLFLOW_PORT}:5000"
    networks:
      - frontend
      - backend
    environment:
      AWS_ACCESS_KEY_ID: ${MINIO_ROOT_USER}
      AWS_SECRET_ACCESS_KEY: ${MINIO_ROOT_PASSWORD}
      MLFLOW_S3_ENDPOINT_URL: http://minio:9000
      MLFLOW_S3_IGNORE_TLS: true
    command: >
      mlflow server
      --backend-store-uri
postgresql+psycopg2://${PG_USER}:${PG_PASSWORD}@postgres:${PG_POR
T}/${PG_DATABASE}
      --default-artifact-root s3://${MLFLOW_BUCKET_NAME}
```

```
      --host 0.0.0.0
      --port 5000
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:5000/"]
      interval: 30s
      retries: 5

volumes:
  db_data:
  minio_data:

networks:
  frontend:
  backend:
```

4️⃣ How to Start Everything

1. Save all three files (`.env`, `Dockerfile.mlflow`, `docker-compose.yml`).
2. Build and start the services:

```shell
docker compose up -d --build
```

3. Check logs:

```shell
docker compose logs -f
```

4. Verify services:
    - MLflow UI → http://localhost:5000
    - MinIO Console → http://localhost:9001 (login: `minio/minio123`)
    - Postgres running on `localhost:5432`

Great 🚀 — since you already did `docker compose up`, let's go step by step to verify **MinIO, MLflow, and Postgres**.

1️⃣ Verify MLflow Server

Check if the tracking server is running:

```shell
docker ps | grep mlflow_server
```

You should see a container named `mlflow_server` with port `5000->5000`.

Then open the UI in your browser:
👉 http://localhost:5000

If it loads, your MLflow tracking server is working ✅

2️⃣ Verify MinIO

Check if the container is running:

```shell
docker ps | grep mlflow_minio
```

Then open the **MinIO Console**:
👉 http://localhost:9001

- Username: `minio`
- Password: `minio123`

You should see a bucket called **mlflow** (created by `create_buckets` service).

👉 To check via CLI inside container:

```shell
docker exec -it mlflow_minio sh
```

Then:

```shell
mc alias set local http://localhost:9000 minio minio123
mc ls local
```

That should show the `mlflow/` bucket.

3️⃣ Verify Postgres

Check if container is running:

```shell
docker ps | grep mlflow_postgres
```

Then connect to Postgres:

```shell
docker exec -it mlflow_postgres psql -U mlflow -d mlflow
```

Inside the psql shell, run:

```SQL
\dt
```

At first, you may see no tables. After you log some runs in MLflow, tables like `experiments`, `runs`, `metrics`, etc. will appear.

Exit with:

```SQL
\q
```

4️⃣ Quick End-to-End Test

- This will create an experiment in **Postgres**.
- Artifacts (if logged) will go into **MinIO**.
- You'll see the run in the MLflow UI.

Testing

1. Creare an folder is called sample file and keep the training and testing script
2. Run this training script 1st (this will have the environment for minio) - this will create the exp and model in minio and the ml flow - verify
3. After this regecter the model
4. Take the run id
5. Using this run id run the testing script - it will pull the model and give the test result
6. We got all the models in mino and we got the exp and model in ml floe, and postgre data we got,
7. In this same location (sample file) we got the irismode, test data (inside test data the have x test and y text)
8. In ml flow folder we got the db data and mino

Look like this : -

```
vikram@edgefast:~/mlflow$ ls
db_data  docker-compose.yml  Dockerfile.mlflow  minio_data  sample_file
—----------------------------------
vikram@edgefast:~/mlflow/sample_file$ cat training.py
import mlflow
import mlflow.sklearn
import os
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import joblib

# --- MLflow setup ---
mlflow.set_tracking_uri("http://192.168.1.200:5000")
mlflow.set_experiment("Iris_Classification")

# Required for artifact handling (MinIO)
os.environ["AWS_ACCESS_KEY_ID"] = "minio"
os.environ["AWS_SECRET_ACCESS_KEY"] = "minio123"
os.environ["MLFLOW_S3_ENDPOINT_URL"] = "http://192.168.1.200:9000"

def train():
    # Load dataset
```

```
iris = load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.Series(iris.target, name="target")

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Save test data for testing script
os.makedirs("test_data", exist_ok=True)
X_test.to_csv("test_data/X_test.csv", index=False)
y_test.to_csv("test_data/y_test.csv", index=False)

# Start MLflow run
with mlflow.start_run() as run:
    print(f"Started MLflow Run ID: {run.info.run_id}")

    # Train model
    model = LogisticRegression(max_iter=200)
    model.fit(X_train, y_train)

    # Log parameters and metrics
    mlflow.log_param("model_type", "LogisticRegression")
```

```
        mlflow.log_param("max_iter", 200)
        accuracy = model.score(X_test, y_test)
        mlflow.log_metric("test_accuracy", accuracy)

        # Save and log model
        joblib.dump(model, "iris_model.pkl")
        mlflow.log_artifact("iris_model.pkl")

        # Log model using MLflow model registry format
        mlflow.sklearn.log_model(model, artifact_path="model")

        print(f"Model logged to MLflow with run_id={run.info.run_id}")
        print(f"Test accuracy: {accuracy:.4f}")

if __name__ == "__main__":
    train()

—----------------------------------------

vikram@edgefast:~/mlflow/sample_file$ cat test
test_data/  testing.py
vikram@edgefast:~/mlflow/sample_file$ cat testing.py
import mlflow
import os
```

```
import pandas as pd
from sklearn.metrics import accuracy_score, classification_report

# --- MLflow setup ---
mlflow.set_tracking_uri("http://192.168.1.200:5000")

# Required for artifact handling (MinIO)
os.environ["AWS_ACCESS_KEY_ID"] = "minio"
os.environ["AWS_SECRET_ACCESS_KEY"] = "minio123"
os.environ["MLFLOW_S3_ENDPOINT_URL"] = "http://192.168.1.200:9000"

def test_latest_model():
    experiment_name = "Iris_Classification"
    client = mlflow.tracking.MlflowClient()

    # Get experiment
    experiment = client.get_experiment_by_name(experiment_name)
    if experiment is None:
        print(f"Experiment {experiment_name} not found!")
        return

    # Get latest run
    runs = client.search_runs(
        experiment_ids=[experiment.experiment_id],
```

```
    order_by=["attributes.start_time DESC"],
    max_results=1,
)
if not runs:
    print("No runs found!")
    return

run = runs[0]
run_id = run.info.run_id
print(f"Using latest run_id: {run_id}")

# Load model
logged_model_uri = f"runs:/{run_id}/model"
try:
    model = mlflow.sklearn.load_model(logged_model_uri)
except Exception as e:
    print(f"Error loading model: {e}")
    return

# Load test data
try:
    X_test = pd.read_csv("test_data/X_test.csv")
    y_test = pd.read_csv("test_data/y_test.csv")
except FileNotFoundError:
```

```
        print("Error: test data not found. Run training first!")
        return

    # Evaluate
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)

    print("\n--- Test Results ---")
    print(f"Test Accuracy: {acc:.4f}")
    print("\nClassification Report:")
    print(report)
    print("----------------------")

if __name__ == "__main__":
    test_latest_model()
vikram@edgefast:~/mlflow/sample_file$
—----------------------
```

Afer running the training you will see the models are in mino and the ml flow after that you run the testing script udin the run id thats it - make sure the postgress is accessible in the pgadmin software and also it has that datas are there

Also check inside the container the datas are available test need to add hera

---

1. Find your Postgres container

To confirm:

```shell
docker ps --format "table {{.Names}}\t{{.Image}}\t{{.Status}}"
```

Look for the container running `postgres`.

From your `docker ps` output, your **Postgres container name is**:

```
mlflow_postgres
```

Use: (not use)

```shell
sudo docker exec -it mlflow_postgres psql -U $PG_USER -d $PG_DATABASE
```

If your `.env` has:

```
PG_USER=mlflow

PG_DATABASE=mlflow
```

then run: (use this)

```shell
sudo docker exec -it mlflow_postgres psql -U mlflow -d mlflow
```

Once inside psql

- List tables:

```sql
\dt
```

- Check experiments:

```sql
SELECT * FROM experiments;
```

- Check runs:

```sql
SELECT run_uuid, experiment_id, status, start_time FROM runs
LIMIT 10;
```

- Check schema of `runs` table:

```sql
\d runs
```

```
sudo mkdir mlflow
  699  ls
  700  cd mlflow/
  701  ls
  702  sudo nano .env
  703  sudo nano Dockerfile.mlflow
  704  sudo nano docker-compose.yml
  705  ls
  706  ll
  707  docker compose up -d --build
  708  sudo docker compose up -d --build
  709  docker compose logs -f
  710  sudo docker compose logs -f
  711  sudo docker exec -it mlflow_minio sh
  712  docker ps | grep mlflow_postgres
  713  sudo docker ps | grep mlflow_postgres
  714  docker exec -it mlflow_postgres psql -U mlflow -d mlflow
  715  sudo docker exec -it mlflow_postgres psql -U mlflow -d mlflow
  716  ls
  717  cat docker-compose.yml
  718  ls
  719  ll
  720  cd minio_data/
  721  ls
  722  ls mlflow/
  723  cd mlflow/
  724  ls
  725  cd ..
  726  ls
  727  cd db_data/
  728  sudo ls db_data/
  729  ls
  730  ll
  731  ls
  732  sudo mkdir sample_file
  733  ls
  734  cd sample_file/
  735  ls
  736  sudo nano training.py
  737  python training.py
  738  sudo chown -R vikram:vikram ~/mlflow/sample_file
  739  python training.py
  740  ls
  741  cd test_data/
```

```
  742  ls
  743  cd ..
  744  sudo docker exec -it mlflow_postgres psql -U mlflow -d mlflow
  745  ls
  746  sudo nano testing.py
  747  ls
  748  python3 testing.py
  749  history
vikram@edgefast:~/mlflow/sample_file$
```