MLOps с MLflow и Airflow

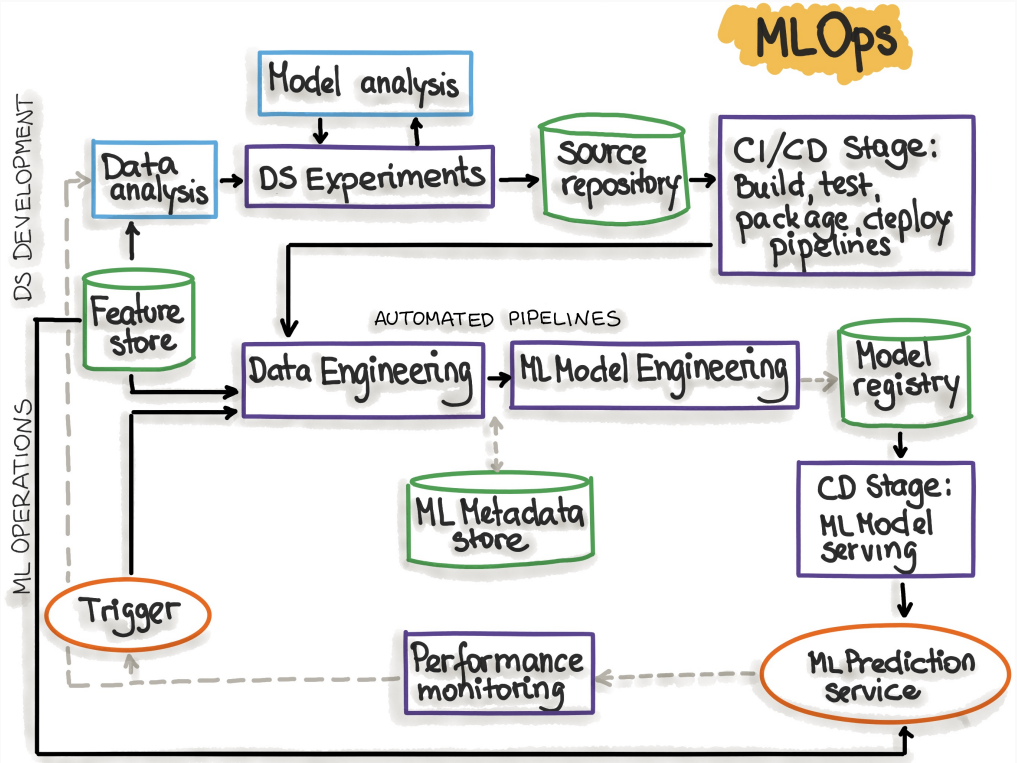
Практическое руководство по развертыванию ML-моделей

ML поступашки. ML Hard

Введение

Почему MLOps?

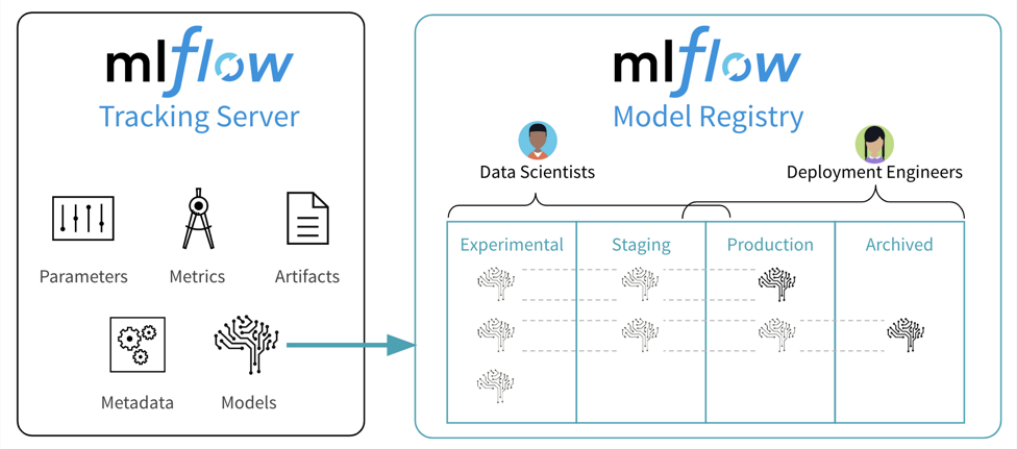
- **Проблема**: ML-модели сложно развертывать и поддерживать.
- **Решение**: MLOps — подход к автоматизации жизненного цикла ML-моделей.
- **Инструменты**: Kubernetes, MLflow, Airflow.



MLflow: Tracking

MLflow Tracking

- **Что это?** Система для логирования параметров, метрик и артефактов.
- **Зачем?** Для воспроизводимости экспериментов.
- **Пример**: Логирование параметров и метрик модели.



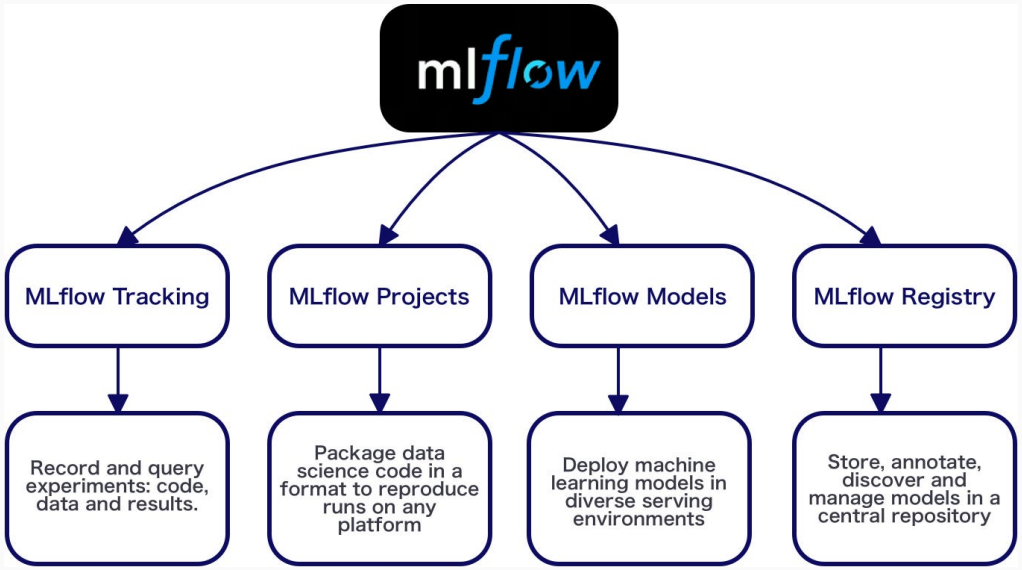
Пример трекинга эксперимента

```
1 import mlflow
2
3 with mlflow.start_run():
4     mlflow.log_param("n_estimators", 100)
5     mlflow.log_metric("accuracy", 0.95)
6     mlflow.sklearn.log_model(model, "model")
```

MLflow: Projects

MLflow Projects

- **Что это?** Упаковка кода для воспроизводимости.
- **Зачем?** Для упрощения запуска экспериментов.
- **Пример**: Определение проекта через ‘MLproject‘.



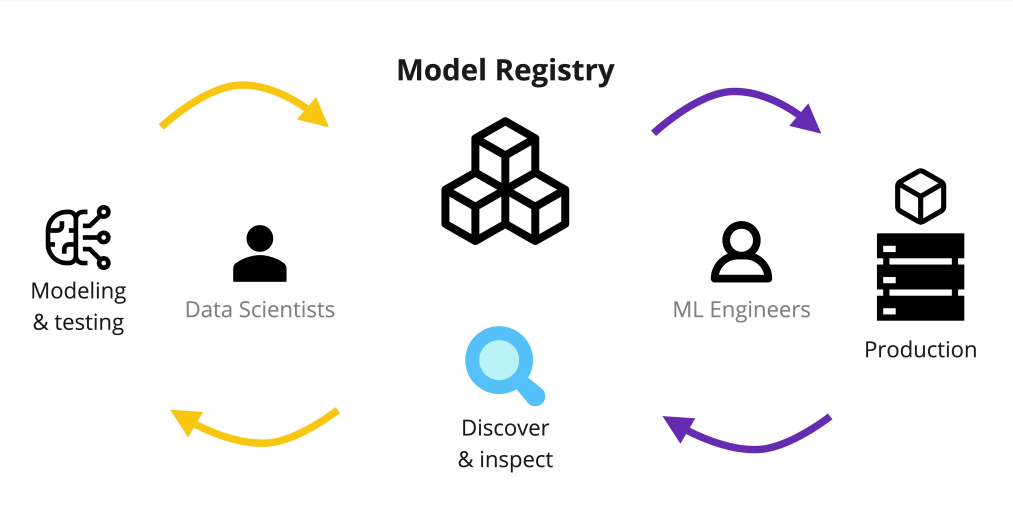
Пример MLproject

```
name: Iris Classification
conda_env: conda.yaml
entry_points:
  main:
    parameters:
      n_estimators: {type: int, default: 100}
    command: "python train.py --n_estimators {
        n_estimators}"
```

MLflow: Model Registry

MLflow Model Registry

- **Что это?** Централизованное управление версиями моделей.
- **Зачем?** Для контроля за жизненным циклом моделей.
- **Пример**: Регистрация модели в реестре.



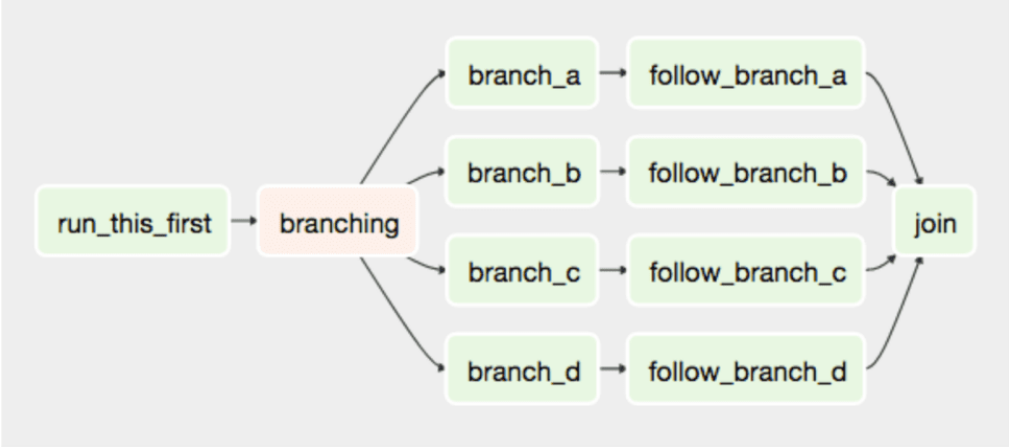
Пример регистрации модели

```
import mlflow

#
mlflow.sklearn.log_model(model, "model")

#
mlflow.register_model(
    "runs:/<RUN_ID>/model",
    "IrisClassifier"
)
```

Airflow

Airflow: Оркестрация пайплайнов

- **DAG**: Определение задач и зависимостей.
- **Операторы**: Выполнение задач (Python, Kubernetes, Bash).
- **Мониторинг**: Визуализация выполнения задач.



Пример DAG для ML-пайплайна

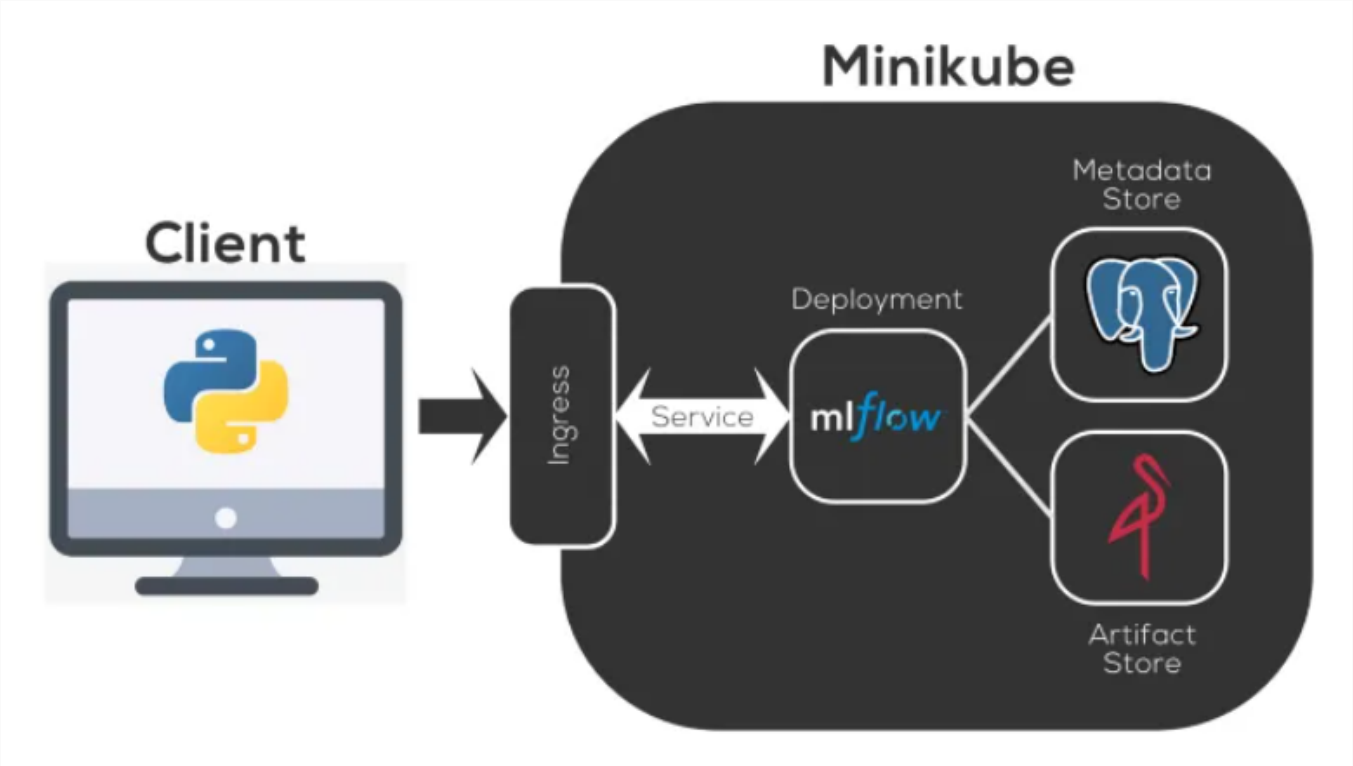
- Задачи: обучение, тестирование, деплой.
- Зависимости: обучение $\rightarrow$ тестирование $\rightarrow$ деплой.

```
1  from airflow import DAG
2  from airflow.operators.python_operator import
       PythonOperator
3
4  default_args = {
5      'owner': 'airflow',
6      'start_date': datetime(2023, 1, 1)
7  }
8
9  with DAG('ml_pipeline', default_args=default_args,
10           schedule_interval='@daily') as dag:
11     train_task = PythonOperator(
12         task_id='train_model',
13         python_callable=train_model
14     )
```

Практическое решение

Пайплайн развертывания модели

1. Обучение модели с логированием в MLflow.
2. Тестирование модели и регистрация в Model Registry.
3. Деплой модели в Kubernetes через Airflow.
4. Мониторинг и автоматическое переобучение.



Преимущества решения

- **Автоматизация**: Уменьшение ручного труда.
- **Масштабируемость**: Использование Kubernetes.
- **Воспроизводимость**: Логирование экспериментов.
- **Мониторинг**: Контроль качества модели.

Заключение

Рекомендации

- Используйте MLflow для трекинга и управления моделями.
- Оркестрируйте пайплайны с Airflow.
- Развертывайте модели в Kubernetes для масштабируемости.
- Настройте мониторинг и автоматическое переобучение.

Вопросы?

Спасибо за внимание!