Машинное обучение и высоконагруженные системы

Москва, осень 2023

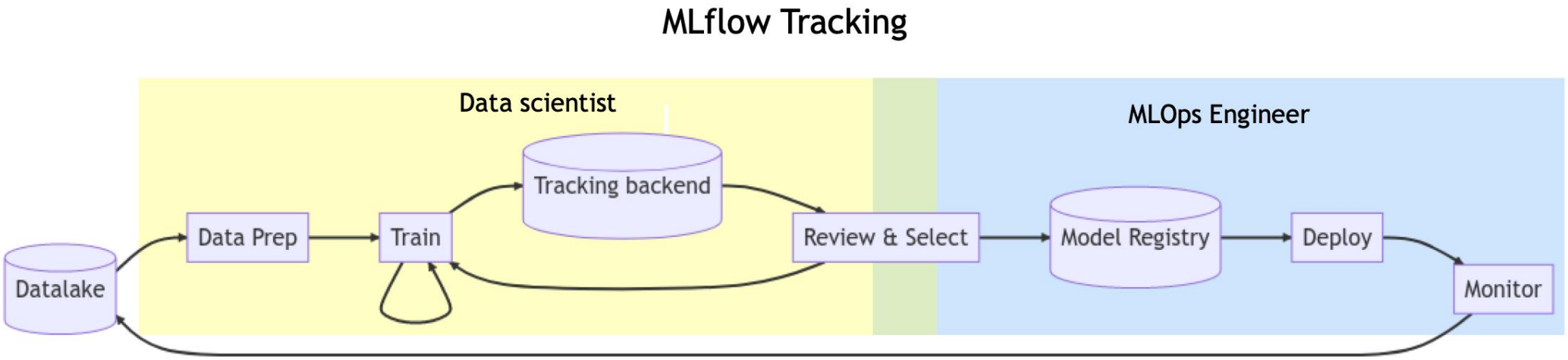
MLOps. Начало

Гаврилова Елизавета,
Senior ML-engineer

Неделя 5. Что будем обсуждать сегодня?

- MLFlow

MLFLow: кто и зачем использует?



MLFlow Tracking: версионирование экспериментов
MLFlow UI: сравнение результатов

MLFlow Models: для деплоя моделей в продуктовую среду
MLFlow Registry: для хранения моделей

**По мнению MLFlow-разработчиков*

mlflow™

Компоненты

Tracking — Версионирование экспериментов: API и UI для сбора данных об экспериментах.

Models — Упаковка моделей в определенном формате и инструменты для их деплоя.

Model Registry — Хранилище моделей с API и UI.

Projects — Упаковка кода в переиспользуемом формате (репозитории на гите или папки с файлами в определенном формате).

Документация

MLFLow Tracking: основные понятия

Run – это запуск data science кода. Каждый **run** записывает следующую информацию:

code version – хэш коммита (при синхронизации с гитом)
start & end time – время начала и конца запуска
source – имя файла или MLFlow Project для запуска (что мы запускаем)
parameters – параметр для запуска
metrics – метрики работы модели

artifacts – выходные файлы запуска (что получаем после отработки run)

Документация

MLFLow Tracking: где хранятся результаты runs?

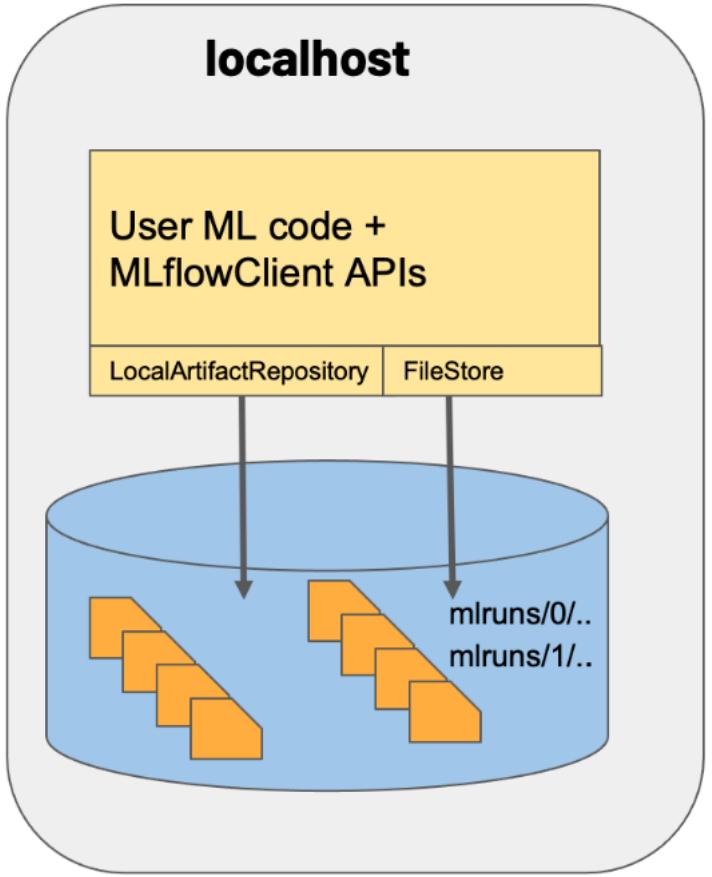
MLFlow поддерживает различные архитектуры MLTracking (т.е. вы можете развернуть MLFlow по-разному, хоть хранить всё в файловой системе).

- Scenario 1: MLflow on localhost
- Scenario 2: MLflow on localhost with SQLite
- Scenario 3: MLflow on localhost with Tracking Server
- Scenario 4: MLflow with remote Tracking Server, backend and artifact stores
- Scenario 5: MLflow Tracking Server enabled with proxied artifact storage access
- Scenario 6: MLflow Tracking Server used exclusively as proxied access host for artifact storage access

MLFLow Tracking: где хранятся результаты runs?

Самый просто сценарий: развернуть всё в локальной файловой системе.

Тогда и результаты экспериментов, и модели будут храниться прямо на компьютере, без использования баз данных и удаленных хранилищ.

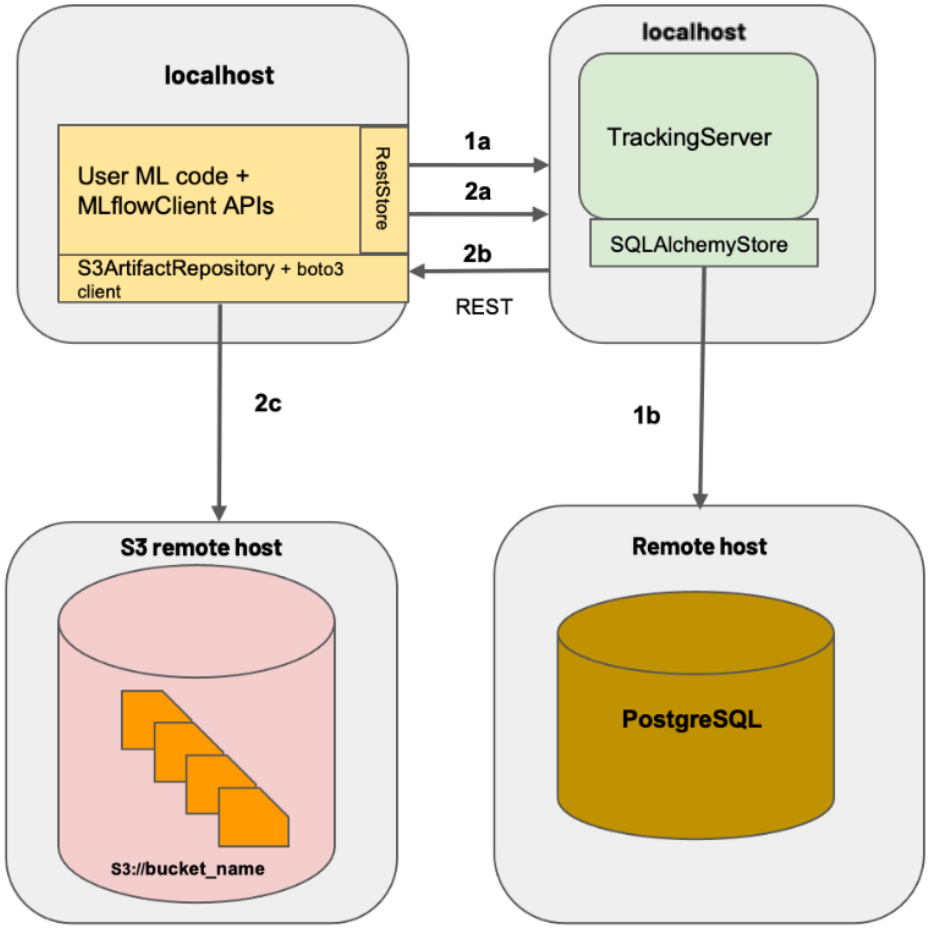


MLFLow Tracking: где хранятся результаты runs?

Scenario 4 modified

Но так как мы имитируем продуктовую среду, мы и разворачивать MLFlow будем не «просто на компьютере».

Мы будем использовать одну из распределённых архитектур, модифицированный тип архитектуры N4.

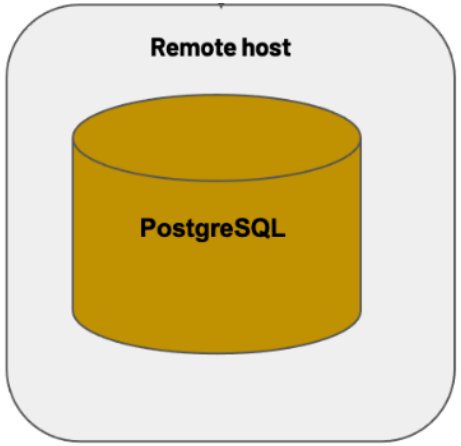


MLFLow Tracking: где хранятся результаты?

Результаты экспериментов(runs) могут храниться:

- ❑ В локальной файловой системе
- ✓ В базе данных

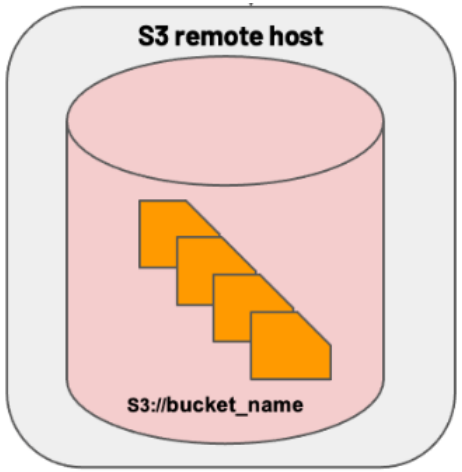
Это хранилище называется **backend store**.



Артефакты могут храниться:

- ❑ В файловых системах
- ✓ На S3
- ❑ На сервере
- ❑ В облачных хранилищах

Это хранилище называется **artifact store**.



MLFLow Projects: основные понятия

Проект – это хранение кода в определенном формате, чтобы другие DS'ы или другие программы могли его запускать.

name – имя проекта
entry points – команды, которые должны запускать в проекте (например, .py или .sh фацлы).
environment – необходимое окружение для запуска проекта (библиотеки и их зависимости).
Поддерживаемые типы окружений: conda, virtualenv, docker.

```
model/
|-- MLmodel
|-- conda.yaml
|-- model.pkl
|-- python_env.yaml
|-- requirements.txt
```

Документация

MLFLow Projects: как запустить

Можно не прописывать конфигурацию проекта, тогда MLFLow будет использовать базовую:

- имя проекта – это имя директории;
- conda.yaml использует в качестве окружения; если такого файла нет, то создается conda.yaml, содержащий только python;
- все .py и .sh файлы в директории – это entry points;
- параметров нет, но их можно задать через CLI или API.

А можно задать конфигурацию проекта через yaml:

```yaml
entry_points:
  main:
    parameters:
      data_file: path
      regularization: {type: float, default: 0.1}
    command: "python train.py -r {regularization} {data_file}"
  validate:
    parameters:
      data_file: path
    command: "python validate.py {data_file}"
```

Документация

MLFLow Projects: как запустить: CLI и API

В командной строке:

```
mlflow run git@github.com:mlflow/mlflow-example.git -P alpha=0.5
```

В коде:

```
mlflow.projects.run()
```

Что потребуется задать:

project uri – путь до директории или ссылка на git
project version – версия проекта (хэш коммита или имя ветки для git)
entry point – главная точка входа
parameters – параметры
deployment mode – способ запуска
environment – окружение для запуска проекта

Документация

MLFLow Models: основные понятия

MLModel – это МЛ-модель,
сконфигурированная с помощью **yaml**-файла, который может содержать:

flavors – форматы, в которых хранится модель
time_created – время создания
signature – описание данных на вход, на выход и параметров модели
input_example – пример данных для обучения (возможные форматы: pd.DataFrame, dict, list, etc.)
mlflow_version – версия mlflow

```
signature:
    inputs: '[{"name": "sepal length (cm)", "type": "double"}, {"name": "sepal width
      (cm)", "type": "double"}, {"name": "petal length (cm)", "type": "double"}, {"name":
      "petal width (cm)", "type": "double"}, {"name": "class", "type": "string", "optional": "true"}]'
    outputs: '[{"type": "integer"}]'
    params: null
```

Документация

MLFLow Models: основные понятия

flavors, описанные в **MLmodel:**

```
time_created: 2018-05-25T17:28:53.35

flavors:
  sklearn:
    sklearn_version: 0.19.1
    pickled_model: model.pkl
  python_function:
    loader_module: mlflow.sklearn
```

Напоминание архитектуры:

```
# Directory written by mlflow.sklearn.save_model(model, "my_model")
my_model/
├── MLmodel
├── model.pkl
├── conda.yaml
├── python_env.yaml
└── requirements.txt
```

Документация

MLFLow Models: что можно делать с моделями

- ❑ Логировать обучение
- ❑ Хранить
- ❑ Оценивать
- ❑ Валидировать
- ❑ Кастомизировать
- ❑ Деплоить

MLFLow Model Registry: основные понятия

Model Registry – это хранилище моделей.

Model – модель
Registered Model – модель, помещенная в хранилище
Model Version – версия модели
Model Stage – версия разработки (*Staging, Production or Archived*)
Annotations and Descriptions – дополнительная информация о моделях
Model Alias – указатель на конкретную версию модели

Прежде чем добавить модель в хранилище, её необходимо логировать с помощью метода **log_model** того или иного **flavor**.

Документация

Давайте поможем Jupyter Notebook стать автоматически обучаемой моделью

Разобрали сегодня

- Что такое MLFlow

ДЗ

- Запустить MLFLow