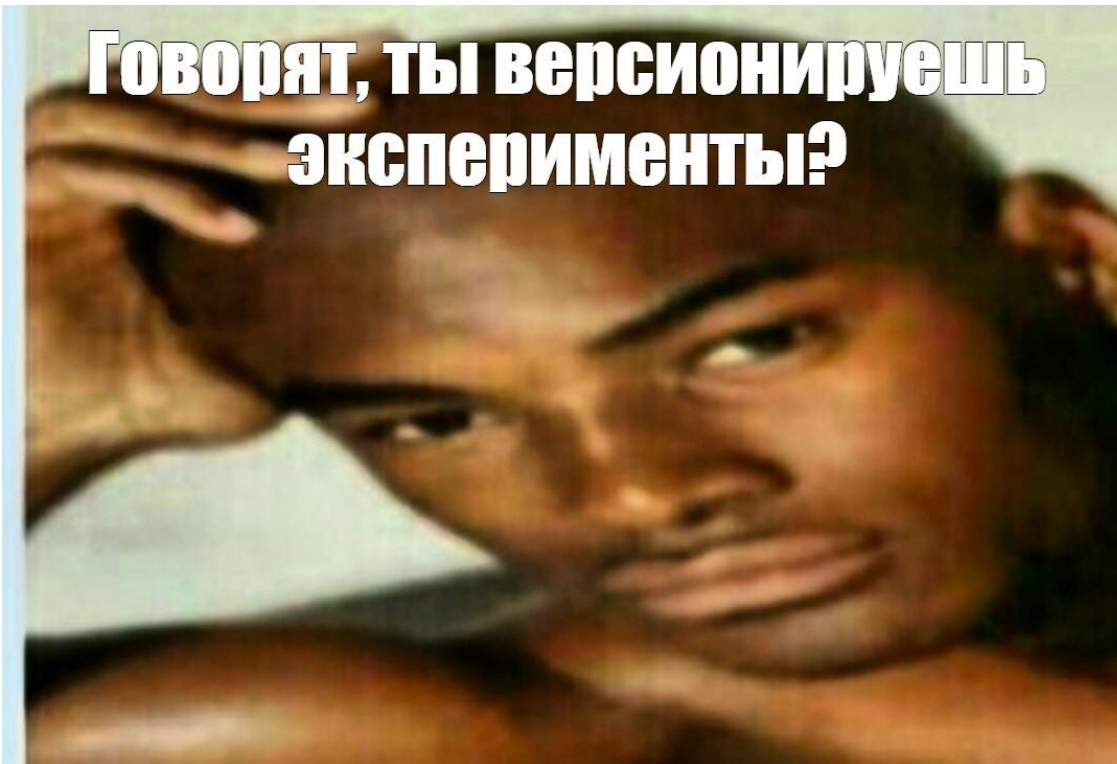


Версионирование данных и моделек



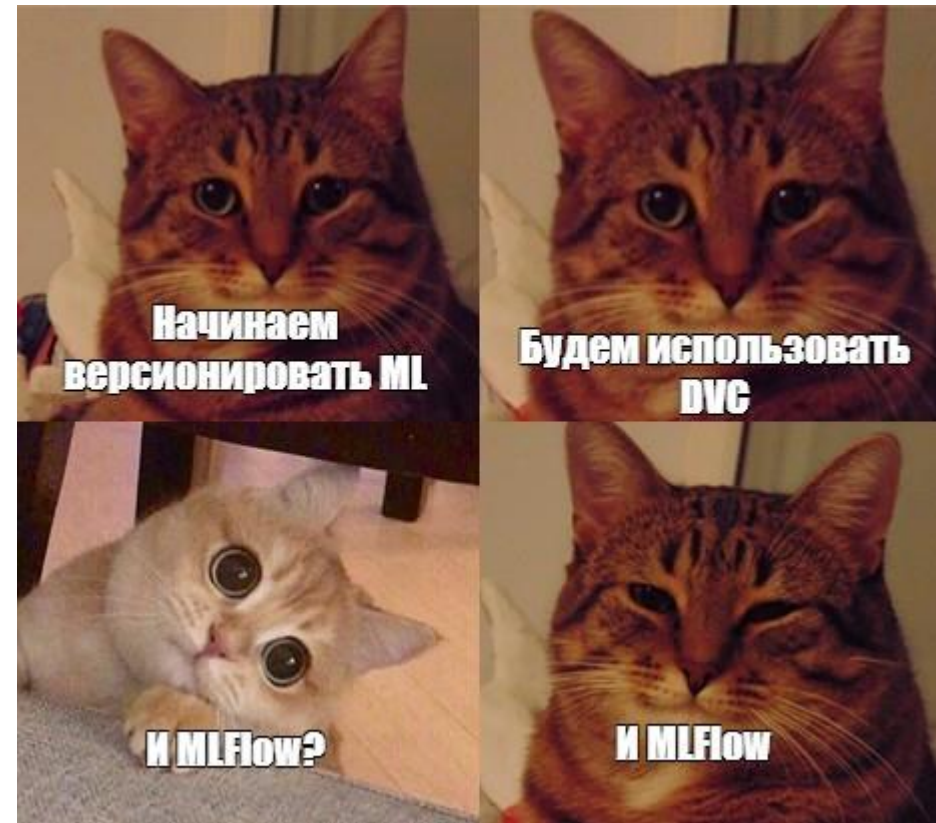
Борисенко Глеб
ФТиАД2021

О чем говорили в прошлый раз

- Что вообще такое и зачем нужны пайплайны
- Примеры типовых пайплайнов
- Dagster

О чем поговорим сегодня

- Зачем версионировать данные и модельки
- Ведение экспериментов
- DVC
- MLFlow



Зачем нужно версионировать данные и модельки

- Хранить данные и модельки – возможность повторно воспользоваться ими
- Версионирование – одна из составляющих для возможности воспроизвести эксперимент.
- Версионировать – означает возможность хранить несколько версий одного и того же объекта данных и переключаться между версиями.

Важный моментик

- Удаление - это специальный маркер, а не фактическое удаление
- Изменение объекта - по сути новый объект

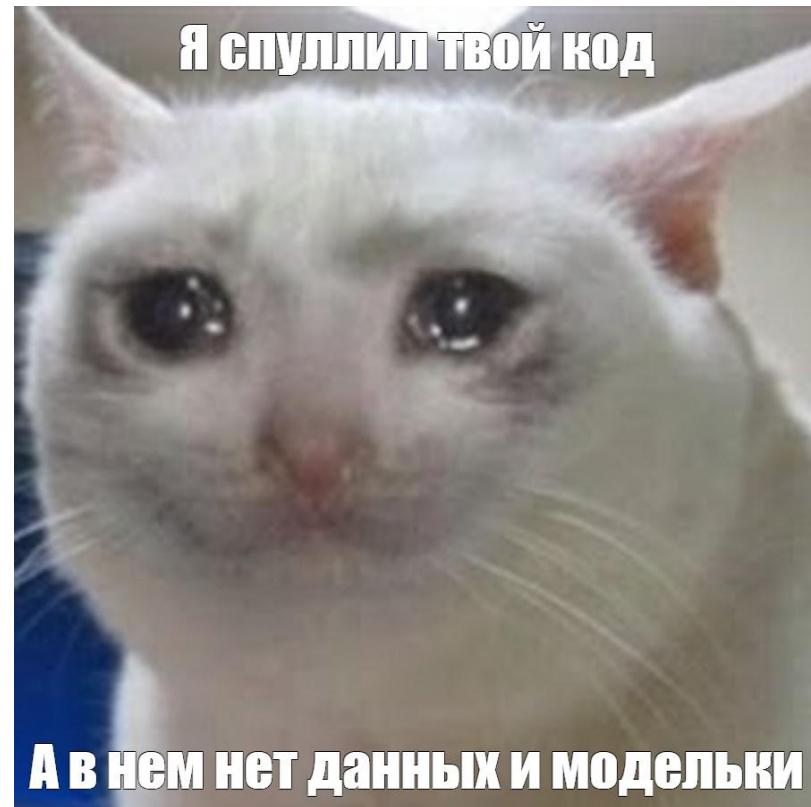
Примеры, когда она нужно

- Делаем экспериментов до задницы – во всем этом нужно суметь разобраться потом:
 - Что делали?
 - Что было на входе?
 - Когда делали?
 - Что получили?
- Моделька переобучилась по расписанию – получилась какаха, нужно откатиться



Командная работа

- Все накидывают на вентилятор экспериментов
- Хочеца делать это КОМАНДНО и не запутаться
- Нужны версии



Версионирование в файловой системе

- git не помощник в версионировании данных
- При решении задачи версионирования в файловой системе хорошо принять решение о разделении данных на чанки
- Некоторые файловые системы имеют встроенные системы версионирования
- Для универсальности – проект DVC

Версионирование данных в базах данных

- Данные меняются - в каждой таблице есть поле версии и признак того что объект удален.
- Изменение - новый объект. Удаление объекта - метка
- Использование триггеров - перегружает и замедляет базу (ок, если это не продакшен)
- Только инкрементальные изменения

Версионирование моделей

- Моделька – по сути, те же самые данные
- Гиперпараметры и веса
- Либо артефакт – файл или набор файлов

Итак, надеюсь, вы поняли

- Вы поняли, зачем нужно версионировать данные и модельки
- Приступим к инструментам



DVC

- Мощщщщщная штука
- По сути, аддон к гиту
- Позволяет версионировать данные с помощью гита
- Храним данные в одном месте – на S3, HDFS, etc., а версионируем метаданные в гите

ОСНОВЫ

- `dvc init` – создает файлы метаданных в `.dvc/`
- `dvc add data/data.json` – типа `git add`
- `git add data/data.xml.dvc data/.gitignore`
- `git commit -m "Add raw data"`

The data, meanwhile, is listed in `.gitignore`.



Под капотом

```
.dvc/cache  
└─ 22  
   └─ a1a2931c8370d3aeedd7183606fd7f
```

outs:

```
- md5: 22a1a2931c8370d3aeedd7183606fd7f  
  path: data.xml
```

Как ~~рулить~~ хранить

- `dvc remote add -d storage s3://mybucket/dvcstore`
- `git add .dvc/config`
- `git commit -m "Configure remote storage«`
- `dvc push`

```
.../dvcstore
└─ 22
    └─ a1a2931c8370d3aeedd7183606fd7f
```

Как скачать и изменить версию

- `git clone gachi_experiment`
- И... ----->

Изменить версию:

- `git checkout`
- `dvc checkout`



Как посмотреть без гита

- `dvc list https://github.com/gigachad/gachi_experiment data`
- `dvc get https://github.com/gigachad/gachi_experiment \`
`data/big_duck`

Есть и python api

Что там еще есть

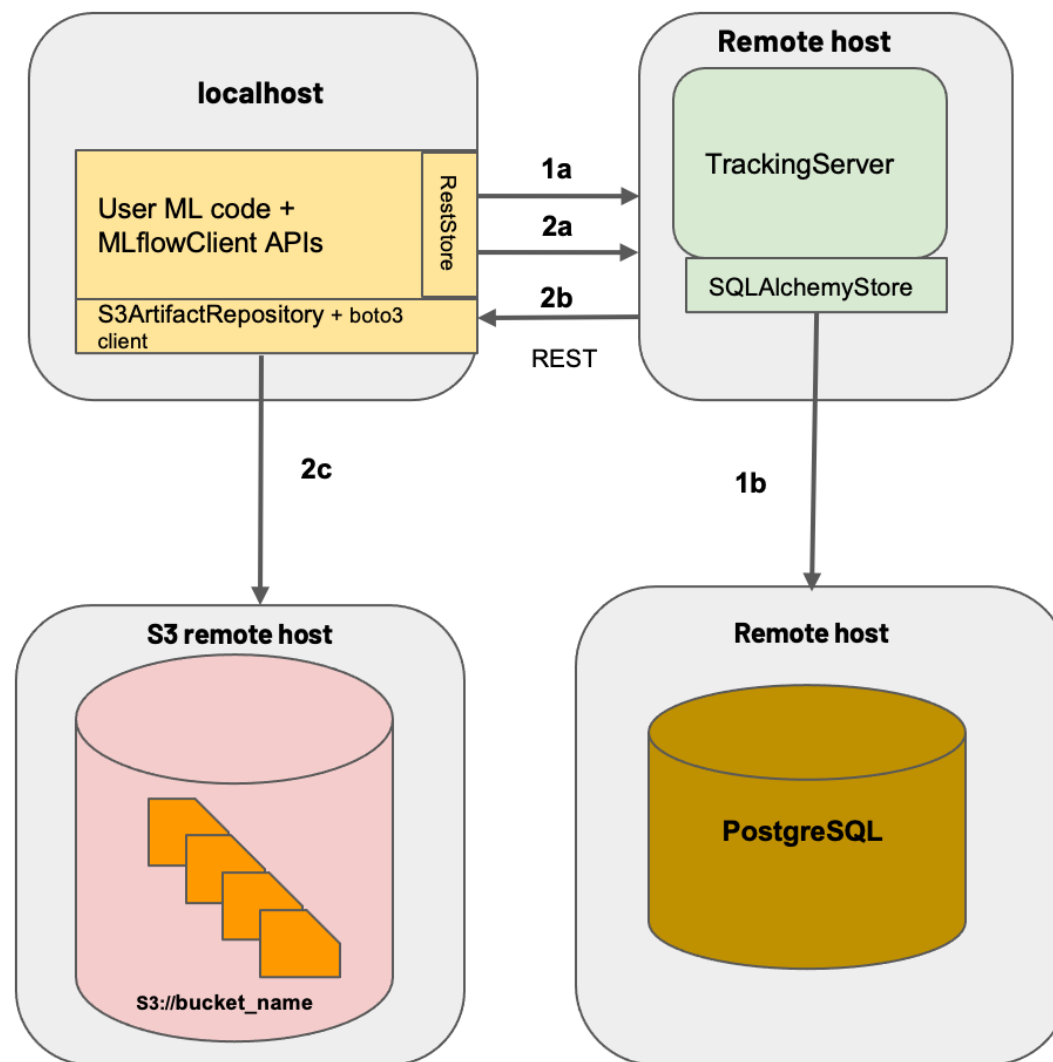
- DVC Pipelines
- DVC Metrics, Plots
- DVC Experiments

MLFlow

- Тоже мощная штука
- Позволяет логировать параметры, графики, модельки, все это хранить
- И все это аккуратно посмотреть в красивом UI



Как чаще всего это используют



MLFlow (Tracking) в питоне

```
mlflow.set_experiment("my-experiment-1")

with mlflow.start_run():

    X, y = load_iris(return_X_y=True)

    params = {"C": 0.1, "random_state": 42}
    mlflow.log_params(params)

    lr = LogisticRegression(**params).fit(X, y)
    y_pred = lr.predict(X)
    mlflow.log_metric("accuracy", accuracy_score(y, y_pred))

    mlflow.sklearn.log_model(lr, artifact_path="models")
    print(f"default artifacts URI: '{mlflow.get_artifact_uri()}'")

from mlflow.tracking import MlflowClient

client = MlflowClient()
client.list_registered_models()
```



MLFlow (Registry) в питоне 2

```
model_name = "nyc-taxi-regressor"
latest_versions = client.get_latest_versions(name=model_name)

for version in latest_versions:
    print(f"version: {version.version}, stage: {version.current_stage}")
```

```
version: 1, stage: Staging
version: 2, stage: Production
version: 4, stage: None
```

```
model_version = 4
new_stage = "Staging"
client.transition_model_version_stage(
    name=model_name,
    version=model_version,
    stage=new_stage,
    archive_existing_versions=False
)
```

```
run_id = "b8904012c84343b5bf8ee72aa8f0f402"
model_uri = f"runs://{run_id}/model"
mlflow.register_model(model_uri=model_uri, name="nyc-taxi-regressor")
```

```
Registered model 'nyc-taxi-regressor' already exists. Creating a new version of this model...
2022/05/19 16:47:17 INFO mlflow.tracking._model_registry.client: Waiting up to 300 seconds for model version to finish creation.
del name: nyc-taxi-regressor, version 4
Created version '4' of model 'nyc-taxi-regressor'.
```



UI 1

	Date	User	Source	Version	Parameters		Metrics		
					alpha	l1_ratio	mae	r2	rmse
<input type="checkbox"/>	2018-06-04 23:00:10	mlflow	train.py	05e956	1	1	0.649	0.04	0.862
<input type="checkbox"/>	2018-06-04 23:00:10	mlflow	train.py	05e956	1	0.5	0.648	0.046	0.859
<input type="checkbox"/>	2018-06-04 23:00:10	mlflow	train.py	05e956	1	0.2	0.628	0.125	0.823
<input type="checkbox"/>	2018-06-04 23:00:09	mlflow	train.py	05e956	1	0	0.619	0.176	0.799
<input type="checkbox"/>	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	1	0.648	0.046	0.859
<input type="checkbox"/>	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0.5	0.628	0.127	0.822
<input type="checkbox"/>	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0.2	0.621	0.171	0.801
<input type="checkbox"/>	2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0	0.615	0.199	0.787
<input type="checkbox"/>	2018-06-04 23:00:09	mlflow	train.py	05e956	0	1	0.578	0.288	0.742
<input type="checkbox"/>	2018-06-04 23:00:09	mlflow	train.py	05e956	0	0.5	0.578	0.288	0.742
<input type="checkbox"/>	2018-06-04 23:00:09	mlflow	train.py	05e956	0	0.2	0.578	0.288	0.742
<input type="checkbox"/>	2018-06-04 23:00:08	mlflow	train.py	05e956	0	0	0.578	0.288	0.742

UI 2

mlflow

[Github](#) [Docs](#)

Run 7c1a0d5c42844dcdb8f5191146925174

Experiment Name: Default

Start Time: 2018-06-04 23:47:22

Source: train.py

Git Commit: 3aa48cffe58b8d9d69f5

User: mlflow

Duration: 145ms

▼ Parameters

Name	Value
alpha	0
l1_ratio	0

▼ Metrics

Name	Value
mae	0.578
r2	0.288
rmse	0.742

► Tags

▼ Artifacts

▼ model

MLmodel

model.pkl


Full Path: /Users/mlflow/mlflow-prototype/mlruns/0/7c1a0d5c42844dcdb8f5191146925174/artifacts/model/MLmodel

Size: 259B

```
artifact_path: model
flavors:
  python_function:
    data: model.pkl
    loader_module: mlflow.sklearn
sklearn:
  pickled_model: model.pkl
  sklearn_version: 0.19.1
run_id: 7c1a0d5c42844dcdb8f5191146925174
utc_time_created: '2018-06-05 06:47:22.757025'
```

24

UI 3



GitHub Docs

Registered Models

search model name

Name	Latest Version	Staging	Production	Last Modified
Model A	Version 1	Version 1	—	2019-10-16 22:51:19
Model B	Version 1	—	—	2019-10-16 22:51:52

< 1 >

Registered Models > Model A > Version 1

Registered At: 2019-10-17 13:38:51

Last Modified: 2019-11-12 09:56:00

▼ Description

Creator:

Source Run: [Run b99a0fc567ae4d32994392c800c0b6ce](#)

Stage: Staging

Transition to → Staging

Transition to → Production

Transition to → Archived

Что там есть еще, но используется реже

- MLFlow Projects – засовывание проекта как бы в пакет
- MLflow Models – помогает деплоить модельки; возможно, используется часто с каким-нибудь Seldon Core или KFServing, но не уверен

Альтернативы

- Neptune (дороже для команды и прикольнее)
- Wandb (Weights and Biases) (еще дороже и прикольнее)
- И другие



DVC и Mlflow можно использовать вместе

- MLFlow – версионирование моделей и параметров
- DVC – версионирование данных
- Ну и Git – версионирование кода



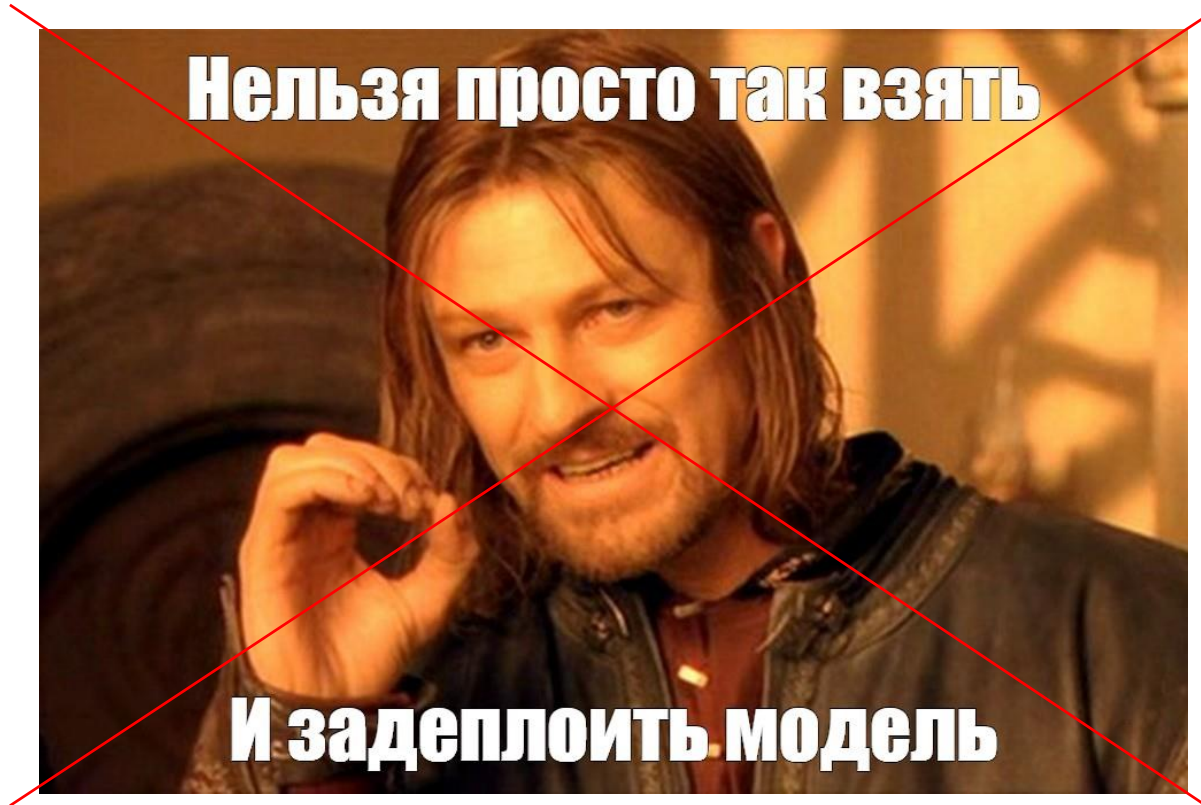
Выводы

- Воспроизводимость эксперимента - это не только гиперпараметры и код, но модельки и данные
- Возможность лучше контролировать результат
- DVC и MLFlow- наши друзья

Что такое Seldon Core

- Seldon Core – фреймворк для создания продакшн микросервисов REST/gRPC из ваших моделей
- Есть wrappers для описания использования вашей модели, по которым генерируется сервис
- Есть свой тип ресурса для кubernetes, что упрощает деплой продакш моделей
- Есть интеграция с существующими моделями (sklearn, tensorflow, etc.)
- Есть куча фич типа канареечного управления, автомасштабирование, логирование и мониторинг из коробки, упрощение A/B тестов и ещё другие.

Что такое Seldon Core (мемом)



<- МОЖНО благодаря Seldon Core

Как выглядит обертка

```
import pickle
class Model:
    def __init__(self):
        self._model = pickle.loads( open("model.pickle", "rb") )

    def predict(self, X):
        output = self._model(X)
        return output
```

```
s2i build . seldonio/seldon-core-s2i-python3:0.18 sklearn_iris:0.1
```

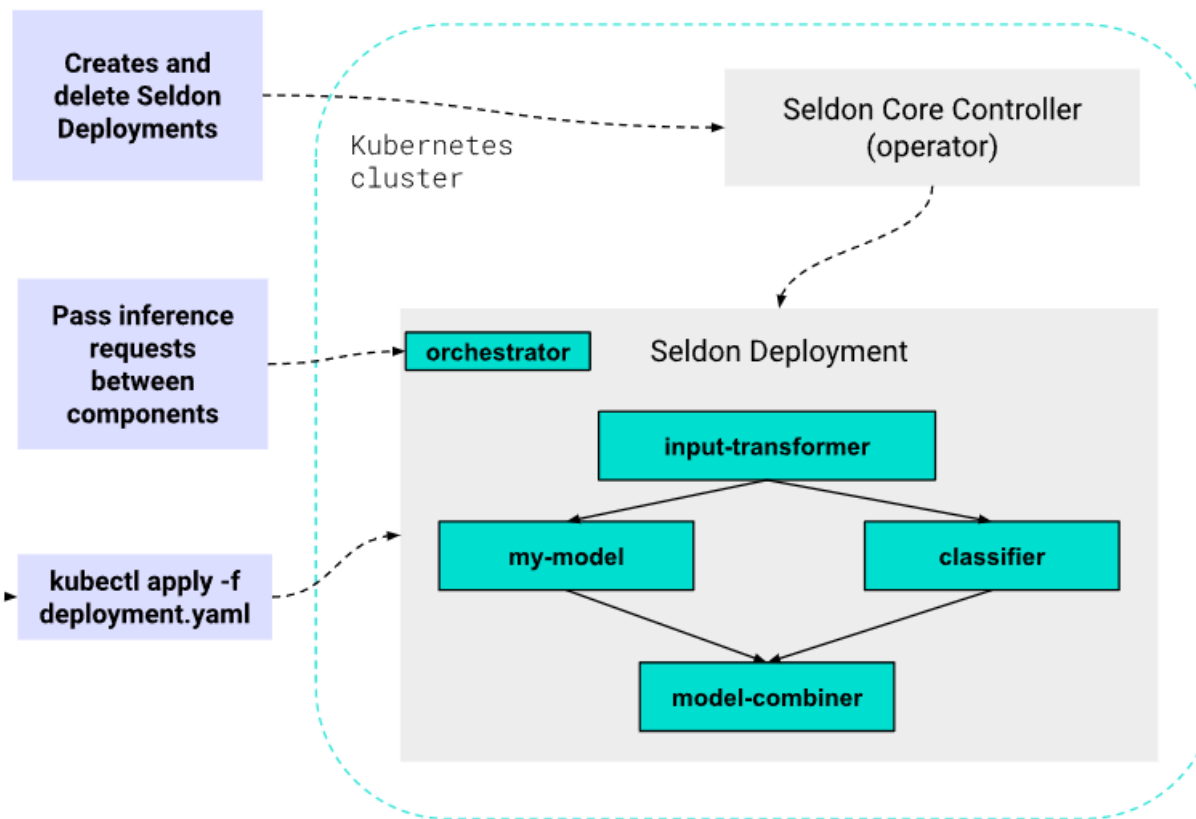

Как выглядит деплой

```
apiVersion: machinelearning.seldon.io/v1
kind: SeldonDeployment
metadata:
  name: iris-model
  namespace: model-namespace
spec:
  name: iris
  predictors:
  - componentSpecs:
    - spec:
        containers:
        - name: classifier
          image: sklearn_iris:0.1
  graph:
    name: classifier
  name: default
  replicas: 1
```

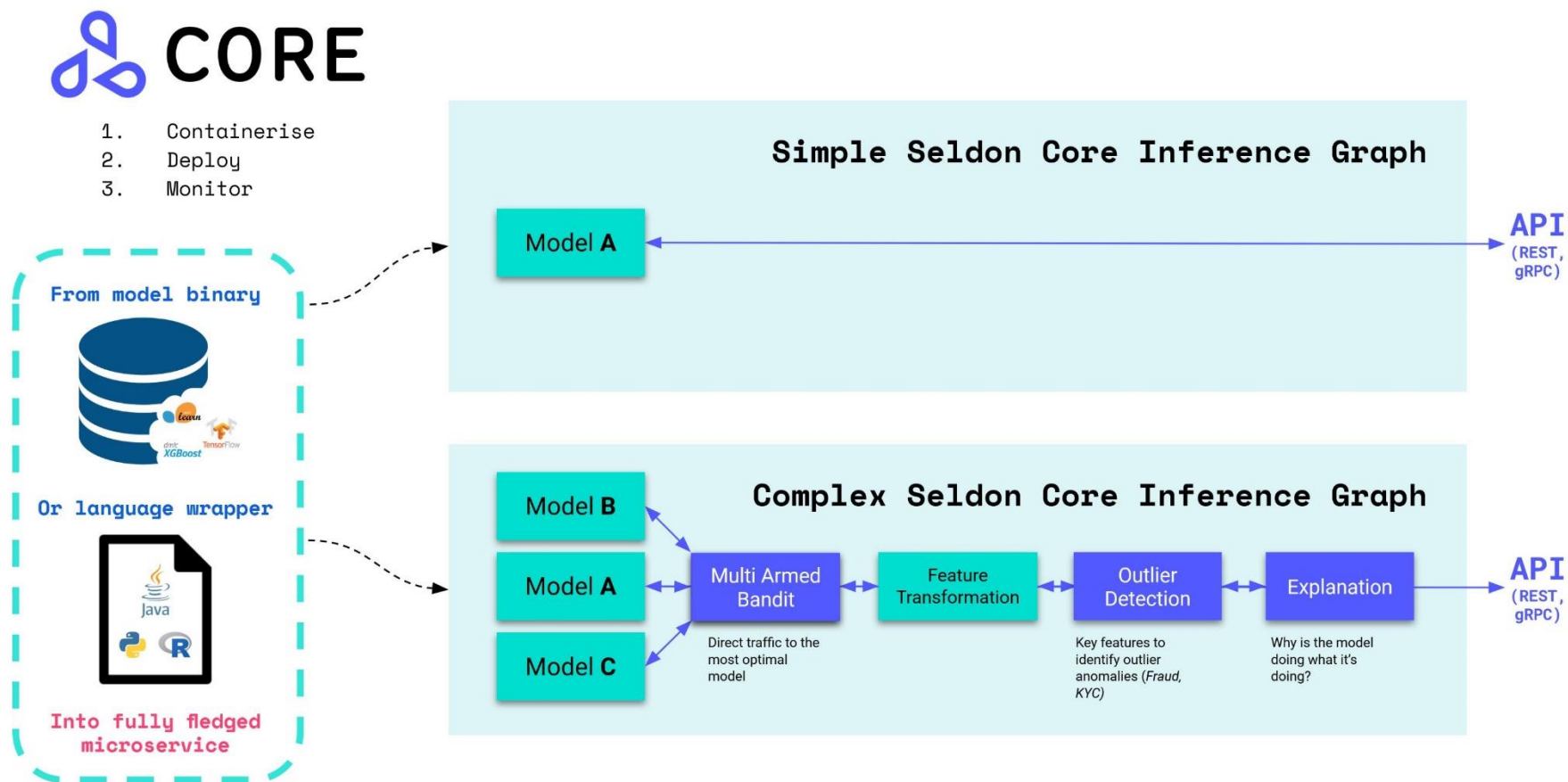
Как выглядит более сложный деплой

```
deployment.yaml

apiVersion: machinelearning.seldon.io/v1
kind: SeldonDeployment
metadata:
  name: example-model
spec:
  name: example
  predictors:
  - componentSpecs:
    - spec:
        containers:
        - image: model:0.1
          name: my-model
        - image: transformer:0.1
          name: input-transformer
        - image: combiner:0.1
          name: model-combiner
  graph:
    name: input-transformer
    type: TRANSFORMER
    children:
    - name: model-combiner
      type: COMBINER
      children:
      - name: my-model
        type: MODEL
      - name: classifier
        implementation: SKLEARN_SERVER
        modelUri: gs://seldon-models/sklearn/iris
  name: default
  replicas: 1
```



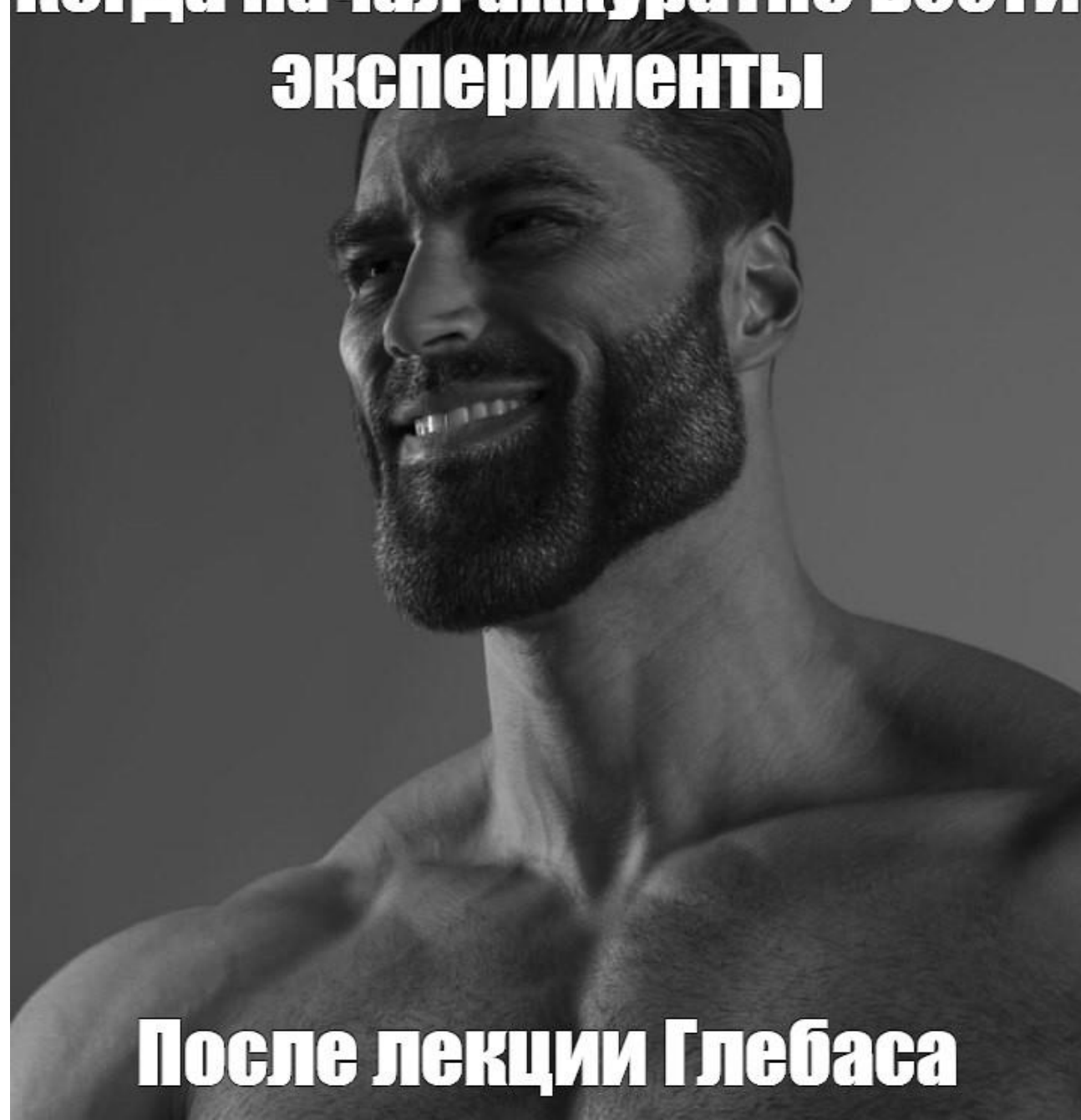
Варианты использования



Смотрите в следующей серии

- Мониторинг моделей
- Grafana + VictoriaMetrics

**Когда начал аккуратно вести
эксперименты**



После лекции Глебаса