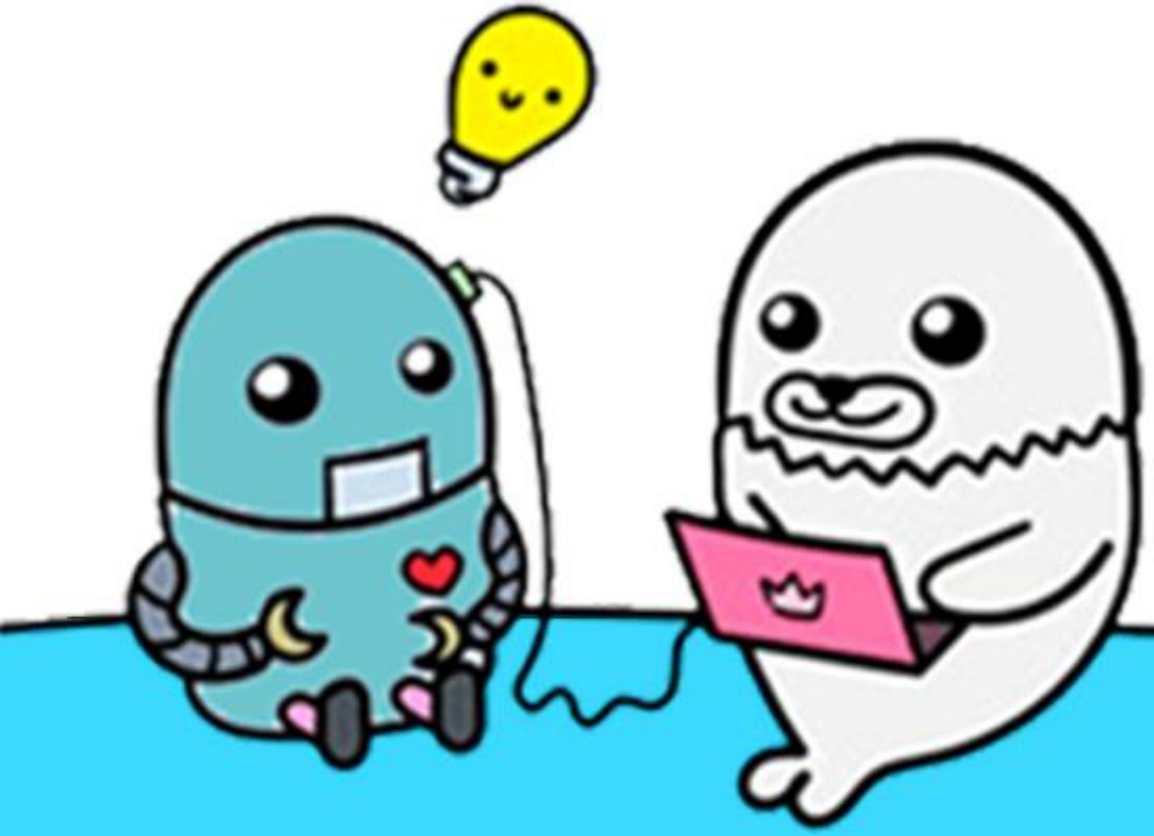
Яндекс Кью

Open Data Science

MLOps и production подход к ML исследованиям



28 марта - 28 мая

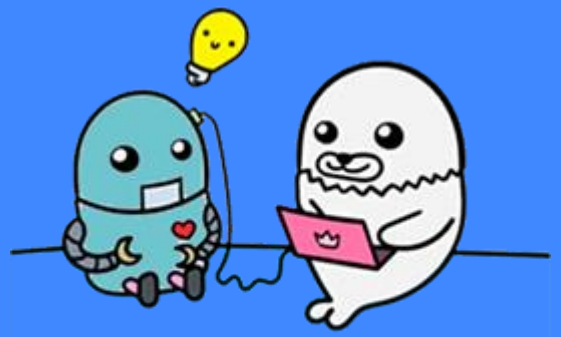
Яндекс Кью

Open Data Science

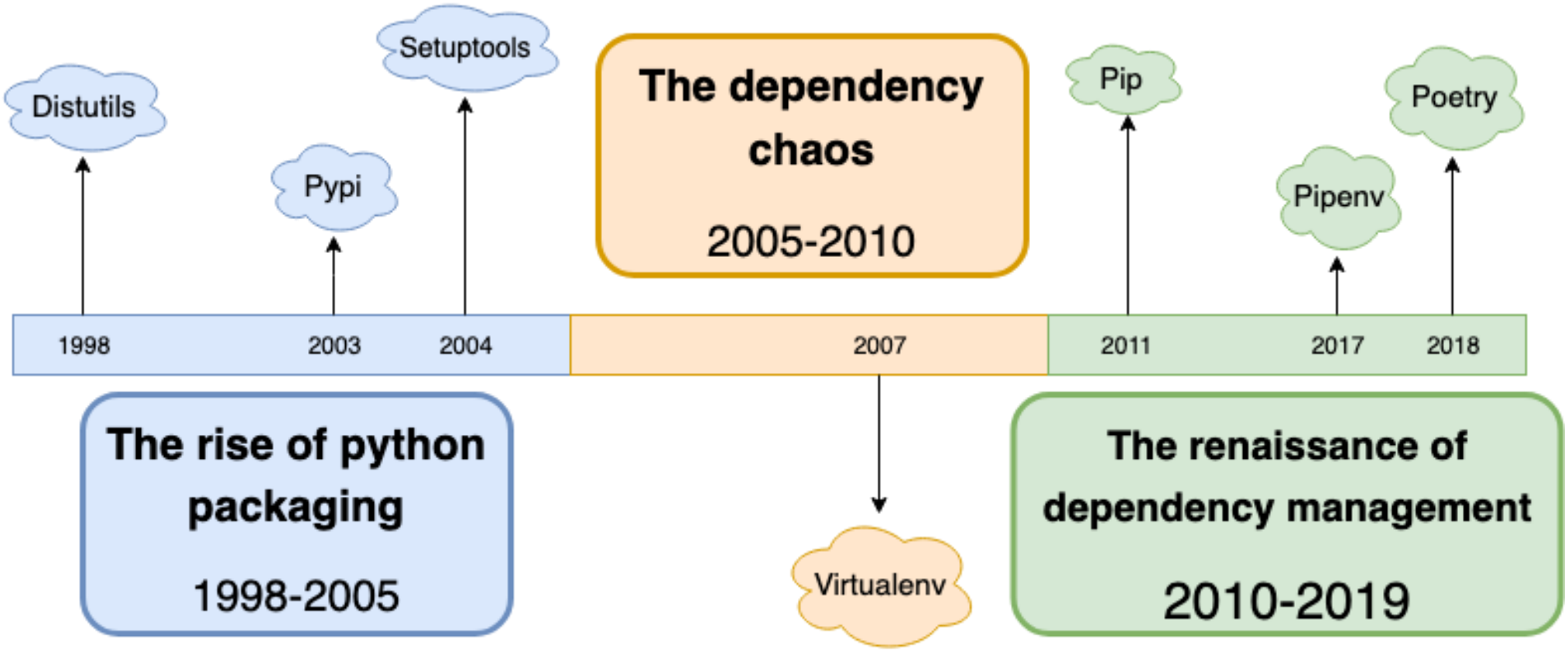
Опыт развёртывания ML сервиса в production на примере проекта FindMyBike

Управление зависимостями и инструменты автоматизации на примере DVC.

Антон Ганичев
Газпромнефть ЦР
Руководитель направления NLP
t.me/pavel_kikin



История развития



Pip

Удаление файлов

- Создаем среду
  - `python -m venv ./venv`
- Активируем
  - `call ./venv/scripts/activate`
- Устанавливаем пакеты
  - `pip install <package_name>`
- Сохраняем зависимости
  - `pip freeze > requirements.txt`
- Устанавливаем зависимости
  - `pip install -r requirements.txt`

Pip

Преимущества

- Менеджер по умолчанию в python

Недостатки

- Отдельные инструменты для изоляции и для установки
- Нет детерминированной сборки
- Невозможность разделения dev и prod среды
- Слабый механизм разрешения конфликтов версий
- Отсутствие механизма упаковки и публикации пакетов
- Обновление зависимостей можно выполнять только вручную

Pipenv

- Создаем среду
  - **`pipenv shell`**
- Устанавливаем зависимость
  - **`pipenv install <package_name>`**
- Фиксация зависимостей
  - **`pipenv lock`**
- Устанавливаем зафиксированных зависимостей
  - **`pipenv install --ignore-pipfile`**

Pipenv

Преимущества

- Объединяет виртуальную среду и установщик
- Разделение prod и dev
- Фиксированная иерархическая структура зависимостей
- Разрешение конфликтов зависимостей
- Возможность удаления пакета вместе с зависимостями
- Трэкинг зависимостей

Недостатки

- Не поддерживает pyproject.toml
- Необходимость поддержки двух файлов для хранения конфигурирования библиотек вроде Black или pytest
- Низкая скорость разрешения зависимостей

Conda

Преимущества

- Фиксированная иерархическая структура зависимостей
- Разрешение конфликтов зависимостей (изолированная установка для каждого пакета)
- Возможность удаления пакета вместе с зависимостями
- Высокая скорость разрешения зависимостей
- Встроенный сборщик пакетов
- Трэкинг зависимостей
- **Установка пакетов с низкоуровневыми зависимостями**
- **Кроссплатформенность. Управление пакетами в любом программном стеке**
- **Изолированный интерпретатор Python в каждой среде**
- **Совместимость среды Conda с pip**

Недостатки

- Тупой resolver
- Сложность сборки пакетов

Poetry

Преимущества

- Фиксированная иерархическая структура зависимостей
- Разрешение конфликтов зависимостей (изолированная установка для каждого пакета)
- Возможность удаления пакета вместе с зависимостями
- Высокая скорость разрешения зависимостей
- Сборка и публикация пакетов одной командой
- Трэкинг зависимостей
- Разделение prod и dev
- Конфигурирование пакетов
- Поддержка pyproject.toml

Недостатки

- Проблемы в случае отсутствия бинарной сборки

Poetry

- Создаем среду в существующем проекте
  - **`poetry init`**

```
$ poetry init

This command will guide you through creating your pyproject.toml config.

Package name [project]:  project_name
Version [0.1.0]:  1.0.0
Description []:  sample project
Author [None, n to skip]:  John Smith <john@example.com>
License []:  MIT
Compatible Python versions [^3.9]:

Would you like to define your main dependencies interactively? (yes/no) [yes] no
Would you like to define your development dependencies interactively? (yes/no) [yes] no
Generated file
```

Poetry

- Создаем среду в существующем проекте
  - **pyproject.toml**

```
[tool.poetry]
name = "project_name"
version = "1.0.0"
description = "sample project"
authors = ["John Smith <john@example.com>"]
license = "MIT"

[tool.poetry.dependencies]
python = "^3.9"

[tool.poetry.dev-dependencies]

[build-system]
requires = ["poetry-core>=1.0.0"]
build-backend = "poetry.core.masonry.api"
```

Poetry

- Один конфигурационный файл для зависимостей и конфигов

```
[tool.black]
line-length = 88
target-version = [ "py37", "py38", "py39",]
exclude = "((.eggs | .git | .pytest_cache | build | dist))"
```

```
[tool.flakehell]
exclude = ["README.rst", "README.md"]
format = "colored"
max_line_length = 88
show_source = true
whitelist = "../../allowlist.txt"
```

```
tool.isort]
default_section = "THIRDPARTY"
known_first_party = "project"
known_django = "django"
sections = "FUTURE,STDLIB,DJANGO,THIRDPARTY,FIRSTPARTY,LOCALFOLDER"
line_length = 120
multi_line_output = 5
include_trailing_comma = true
ensure_newline_before_comments = true
use_parentheses = true


[tool.pytest.ini_options]
DJANGO_SETTINGS_MODULE = "project.settings"
python_files = ["tests.py", "test_*.py", "*_tests.py"]
```

Poetry

- Устанавливаем пакеты
  - `poetry add <package_name>`
  - `poetry add --dev <package_name>`

```
$ poetry add django
Using version ^3.2.3 for Django

Updating dependencies
Resolving dependencies... (0.6s)

Writing lock file

Package operations: 4 installs, 0 updates, 0 removals

  • Installing asgiref (3.3.4)
  • Installing pytz (2021.1)
  • Installing sqlparse (0.4.1)
  • Installing django (3.2.3)
```

Poetry

- Запуск скрипта
  - **`poetry run python <package_name>`**
- Активация среды (опционально)
  - **`poetry shell`**
- Деактивация среды
  - **`poetry exit`**
- Обновление зависимостей
  - **`poetry update`**
- Установка зависимостей
  - **`poetry install`**
- Фиксация зависимостей в poetry.lock
  - **`poetry lock`**

Poetry

- Просмотр дерева зависимостей и обновлений
  - **`poetry show --tree`**
  - **`poetry show --latest`**

```
$ poetry show --tree
requests-toolbelt 0.8.0 A utility belt for advanced users...
└── requests <3.0.0,>=2.0.1
    ├── certifi >=2017.4.17
    ├── chardet >=3.0.2,<3.1.0
    ├── idna >=2.5,<2.7
    └── urllib3 <1.23,>=1.21.1

$ poetry show --latest
pendulum 2.0.4   1.4.5 Python datetimes made easy.
django   1.11.11 2.0.3 A high-level Python Web framework ...
requests 2.18.4  2.18.4 Python HTTP for Humans.
```

Poetry

- Делаем сборку пакета для публикации на Pypi
  - **`poetry build`**

```
$ poetry build
Building project_name (1.0.0)
  - Building sdist
  - Built project_name-1.0.0.tar.gz
  - Building wheel
  - Built project_name-1.0.0-py3-none-any.whl
```

- Публикуем пакет на Pypi
  - **`poetry publish`**

```
$ poetry publish


Publishing poetry (1.0.0) to PyPI
  - Uploading project_name-1.0.0.tar.gz
  - Uploading project_name-1.0.0-py3-none-any.whl
```

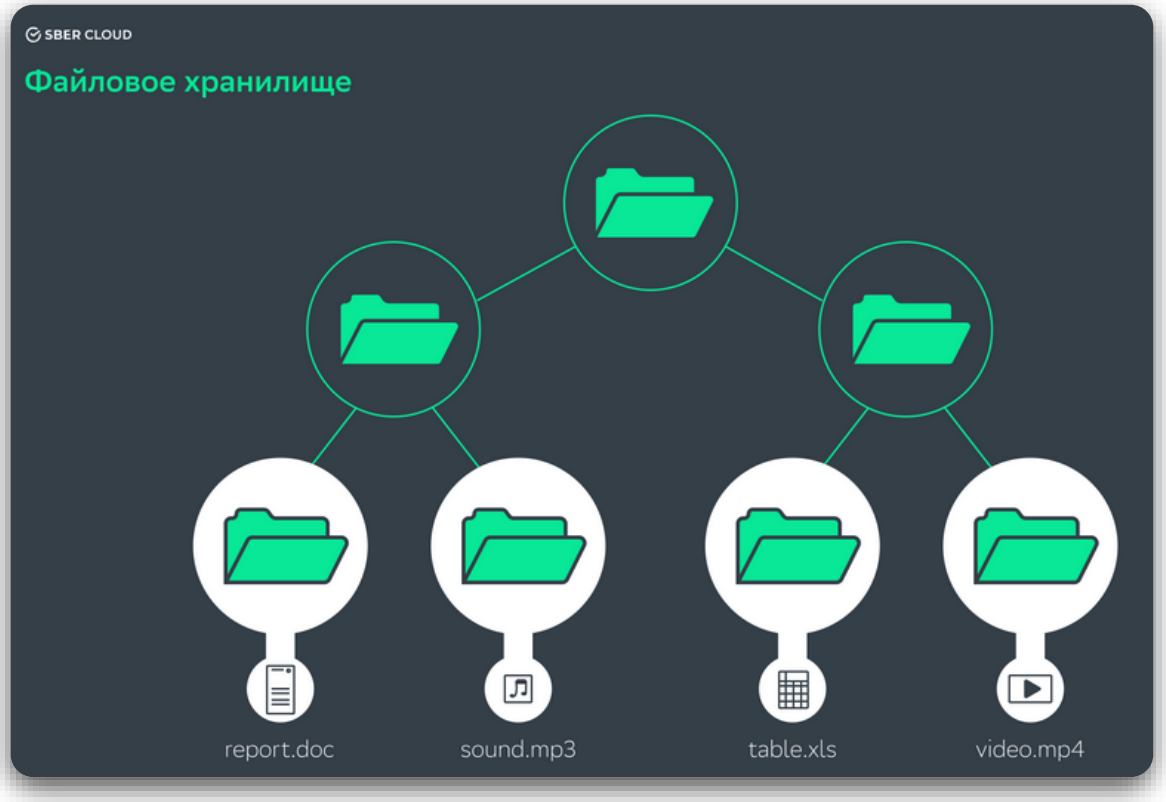
Файловое хранилище

Достоинства:

- привычный для простого пользователя метод хранения данных, не требует высокой квалификации для настройки и использования.

Ограничения:

- не подходит для хранения большого объема данных;
- ограничения на размер файла и длину имени;
- невозможно управлять одновременными подключениями с тысячи компьютеров;
- нужно следить за исчерпанием объема ресурсов.



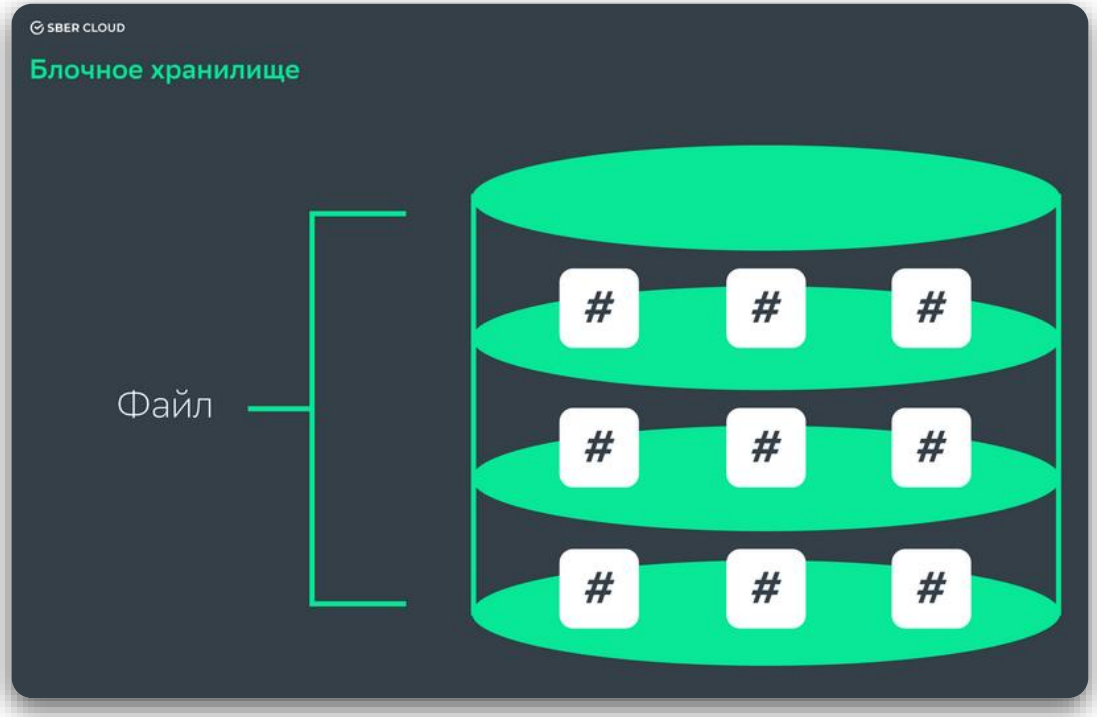
Блочное хранилище

Достоинства:

- высокая скорость передачи данных и производительность.

Ограничения:

- высокая стоимость в сравнении с файловым и объектным хранилищами;
- требуется квалификация, чтобы настроить ПО для работы с блочным хранилищем;
- нужно следить за исчерпанием объема ресурсов.



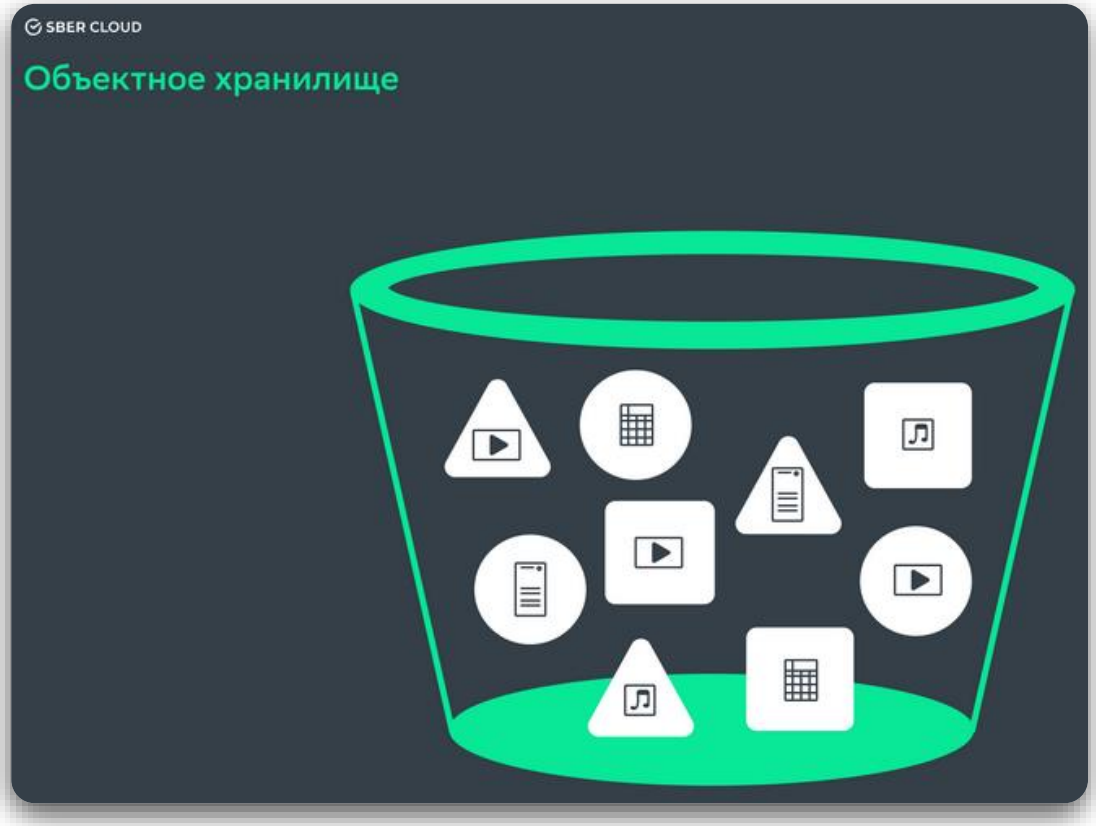
S3 хранилище

Достоинства:

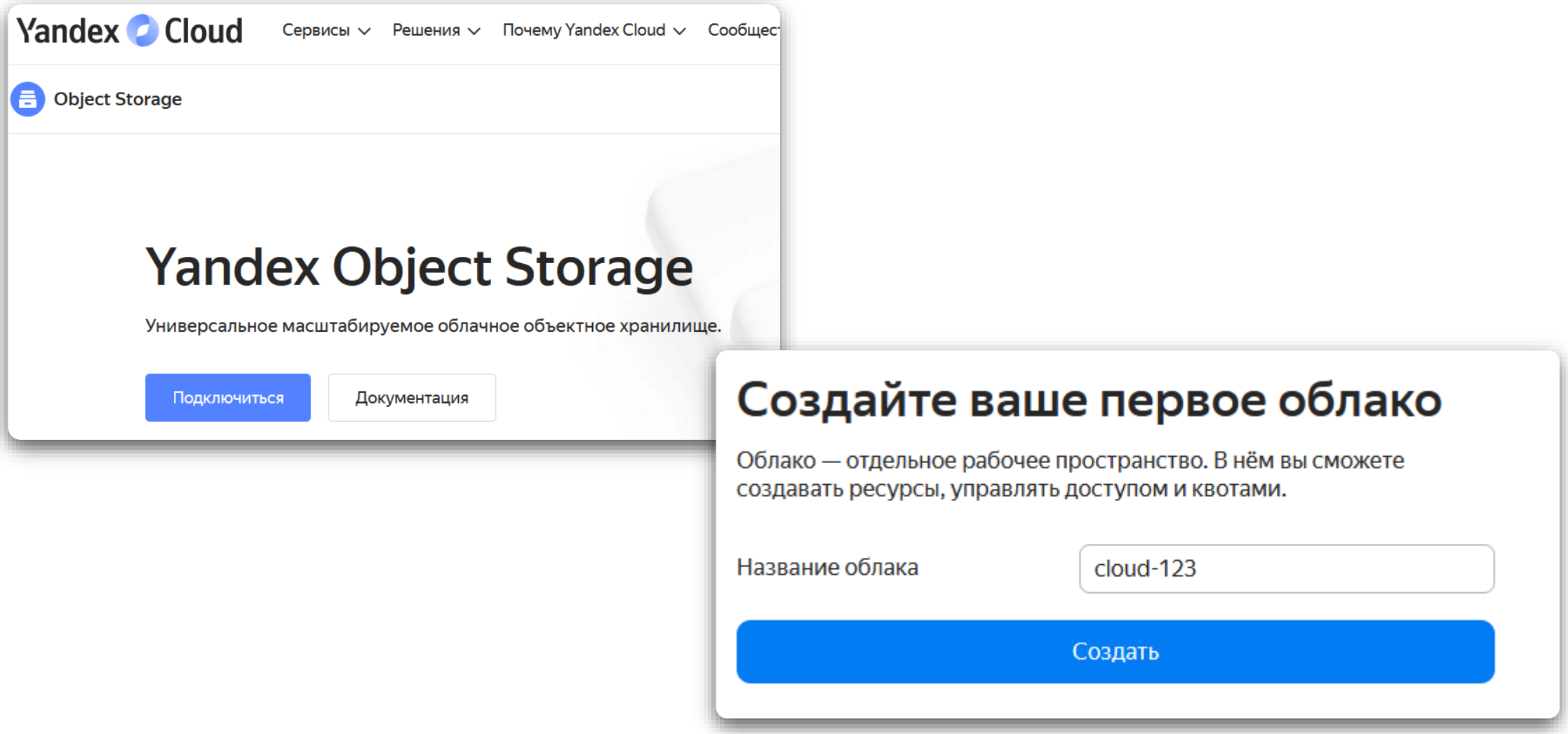
- доступ к данным через HTTP API;
- бесконечная масштабируемость;
- быстрый поиск объектов за счет расширенных метаданных и плоского адресного пространства;
- георепликация (хранение копий объектов на географически распределенных серверах);
- хранение данных любого типа и размера.

Ограничения:

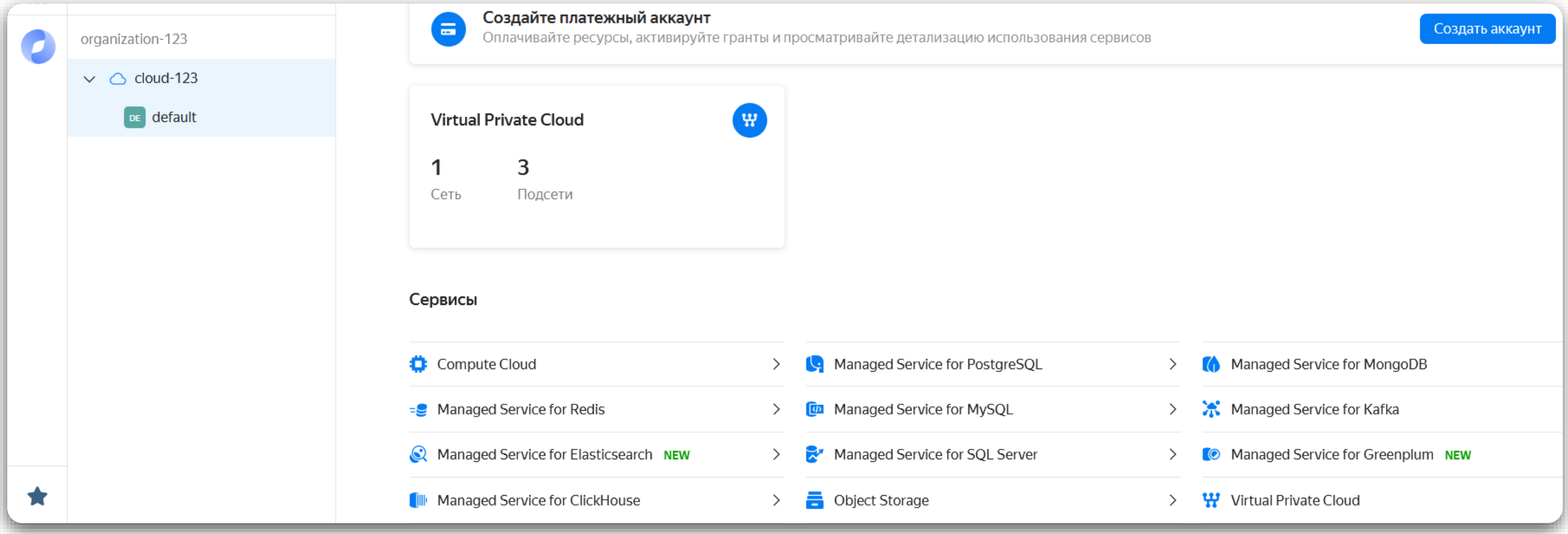
- требуется квалификация для настройки ПО при работе с S3 через API;
- невысокая скорость передачи данных в сравнении с блочным хранилищем.



Настройка S3 хранилища



Настройка S3 хранилища



Настройка S3 хранилища

Создайте ваш первый бакет для хранения объектов

Yandex Object Storage — универсальное масштабируемое решение для хранения данных любого объема. Для доступа к данным можно использовать популярные инструменты для работы с объектными хранилищами — API сервиса совместим с API Amazon S3.

Данные в Object Storage хранятся в виде объектов внутри бакетов. Объект — это пользовательский файл произвольного формата. Бакет — логическая сущность, которая помогает организовать хранение объектов.

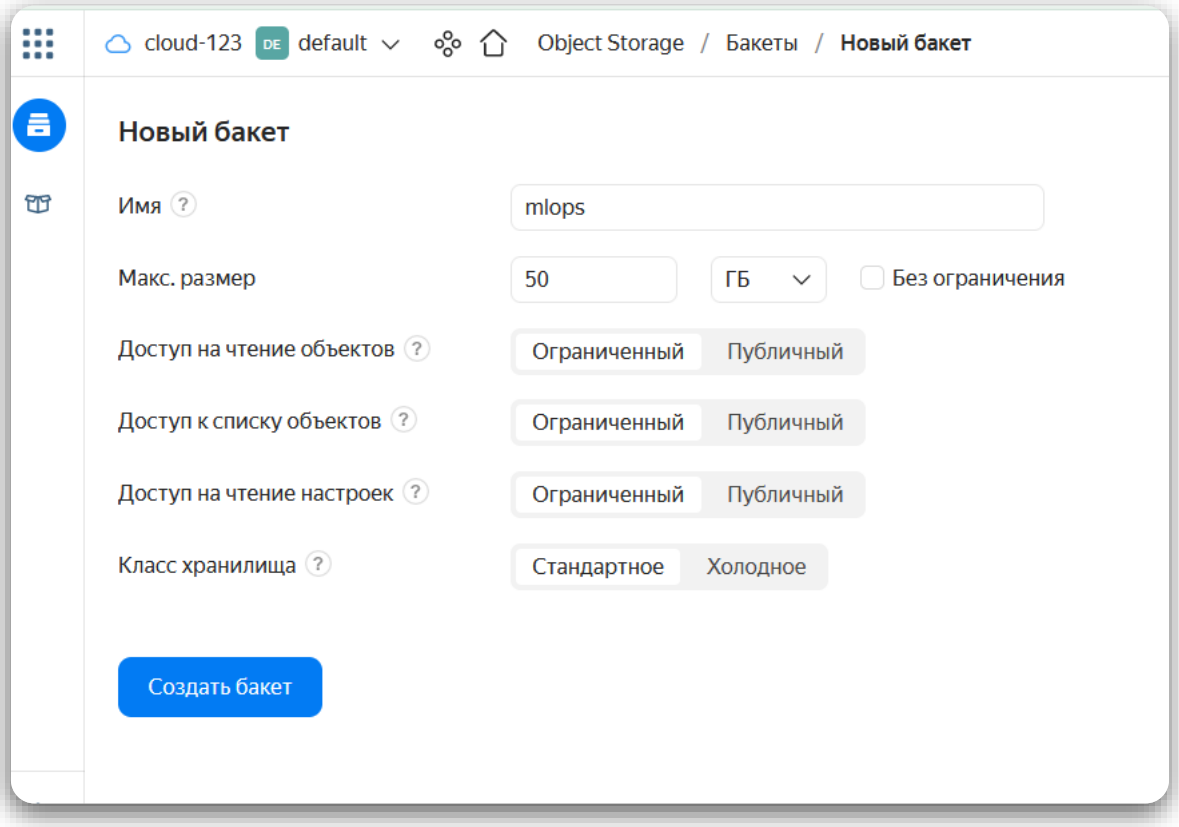
Чтобы сохранить ваши данные в Object Storage, создайте бакет и загрузите в него файлы.

Подробнее о сервисе читайте в документации:

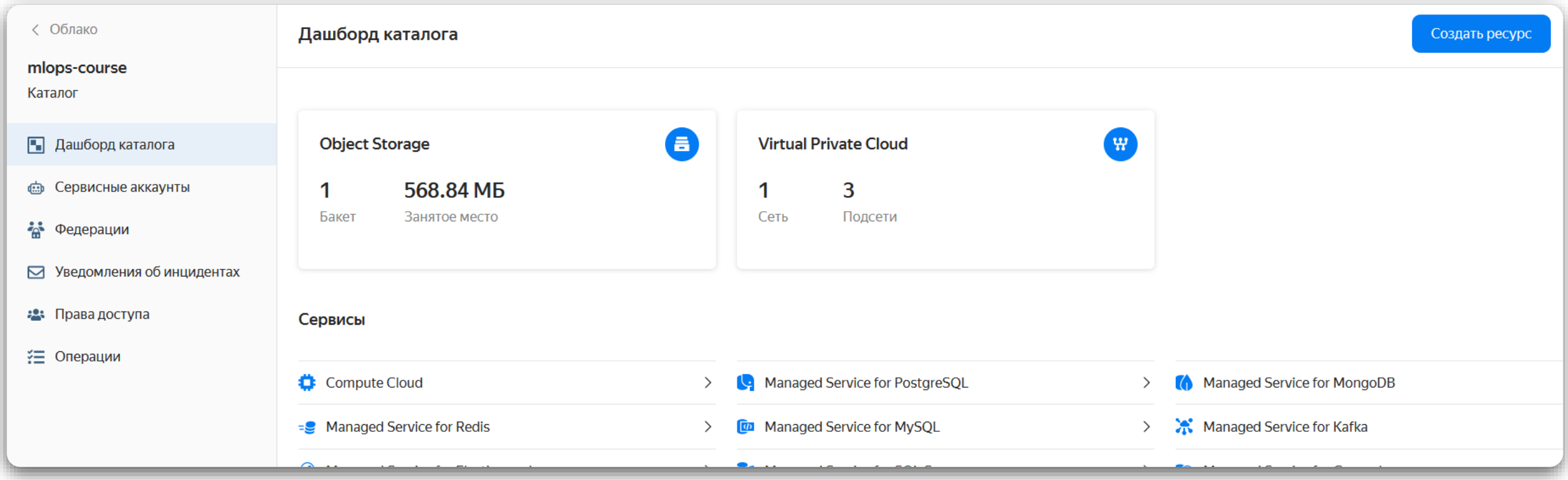
- Начало работы с бакетами и объектами
- Инструменты для работы с Object Storage
- Документация Yandex Object Storage

Создать бакет

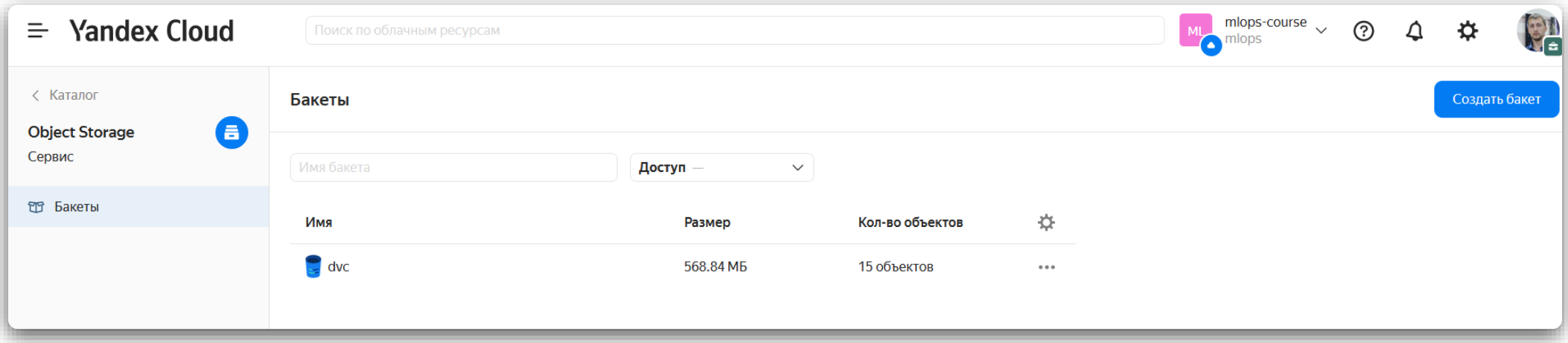
Настройка S3 хранилища



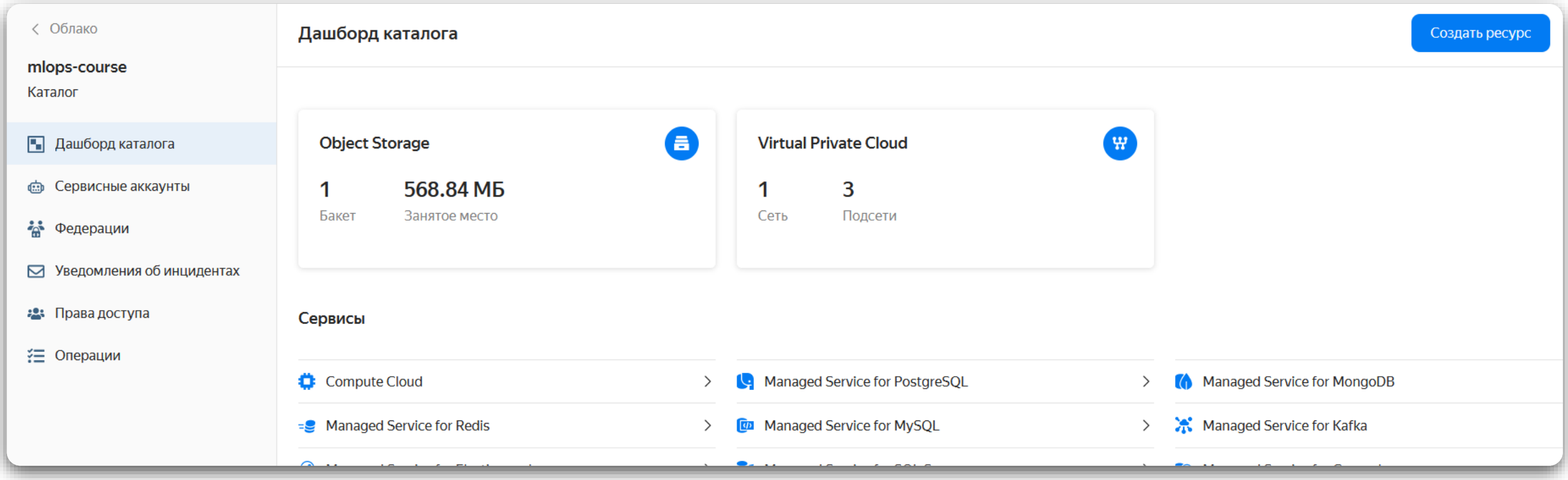
Настройка S3 хранилища



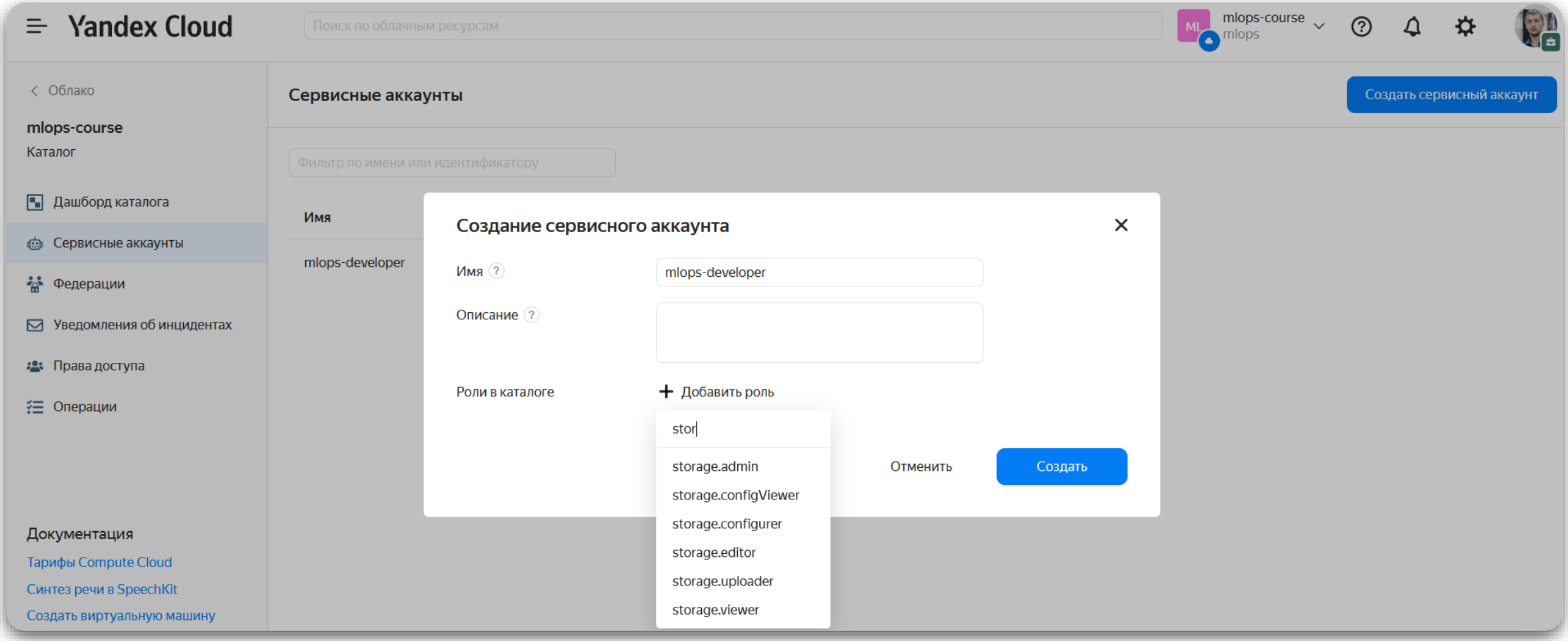
Настройка S3 хранилища



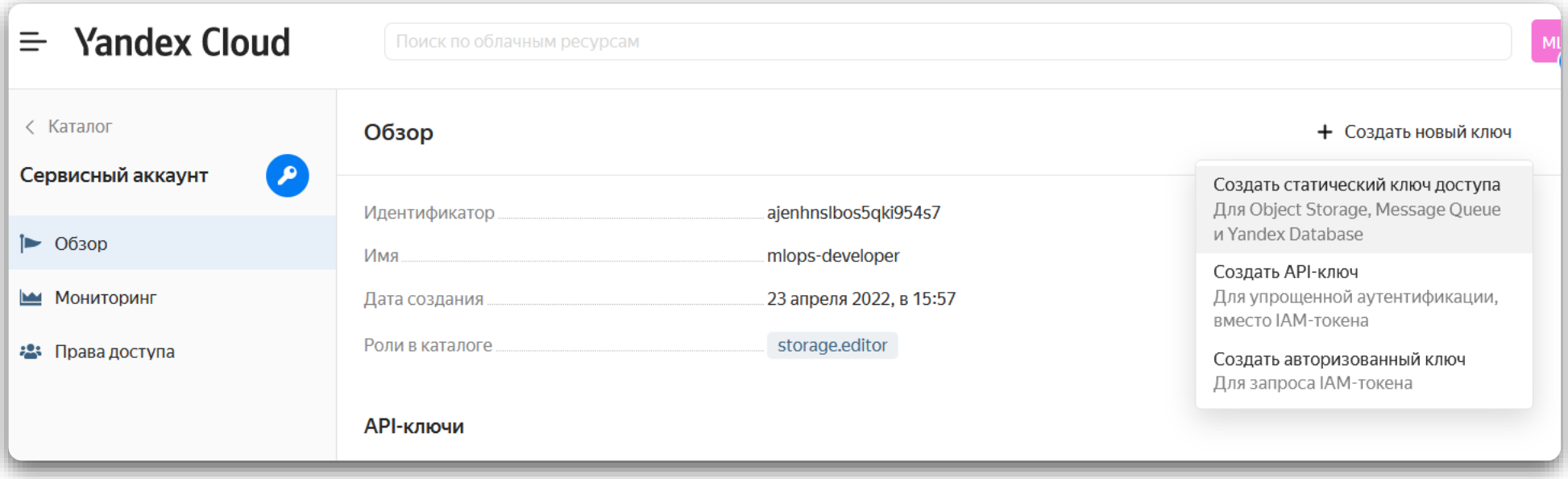
Настройка S3 хранилища



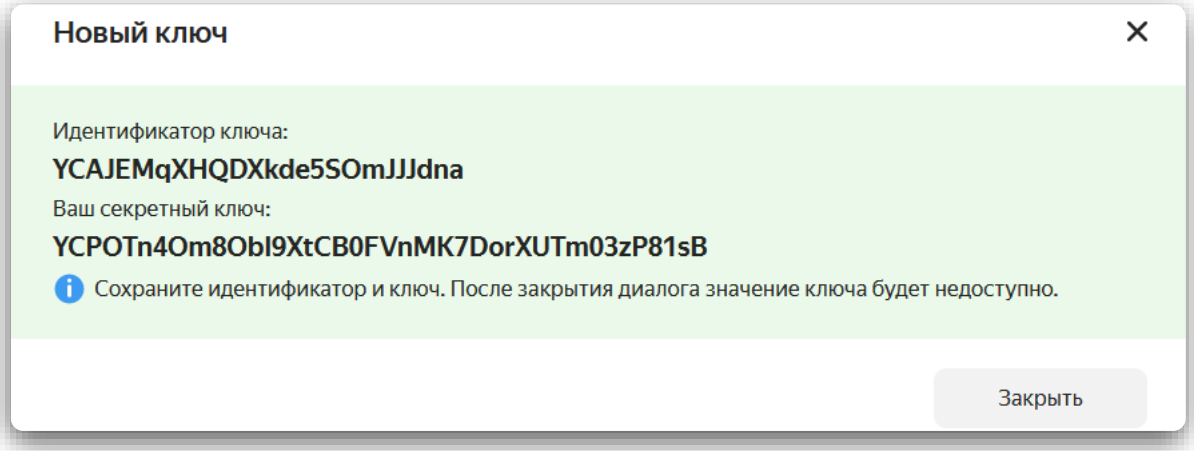
Настройка S3 хранилища



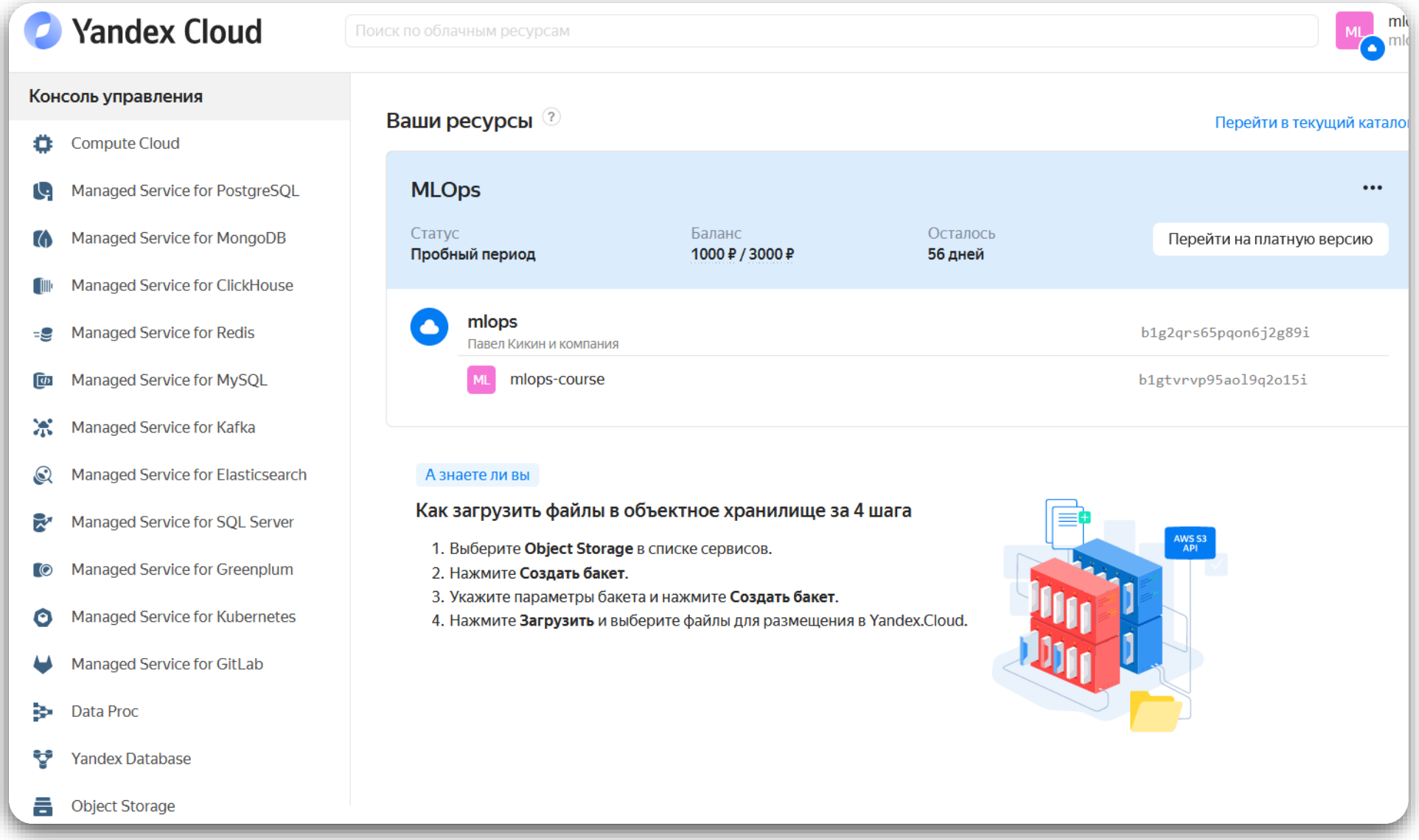
Настройка S3 хранилища



Настройка S3 хранилища

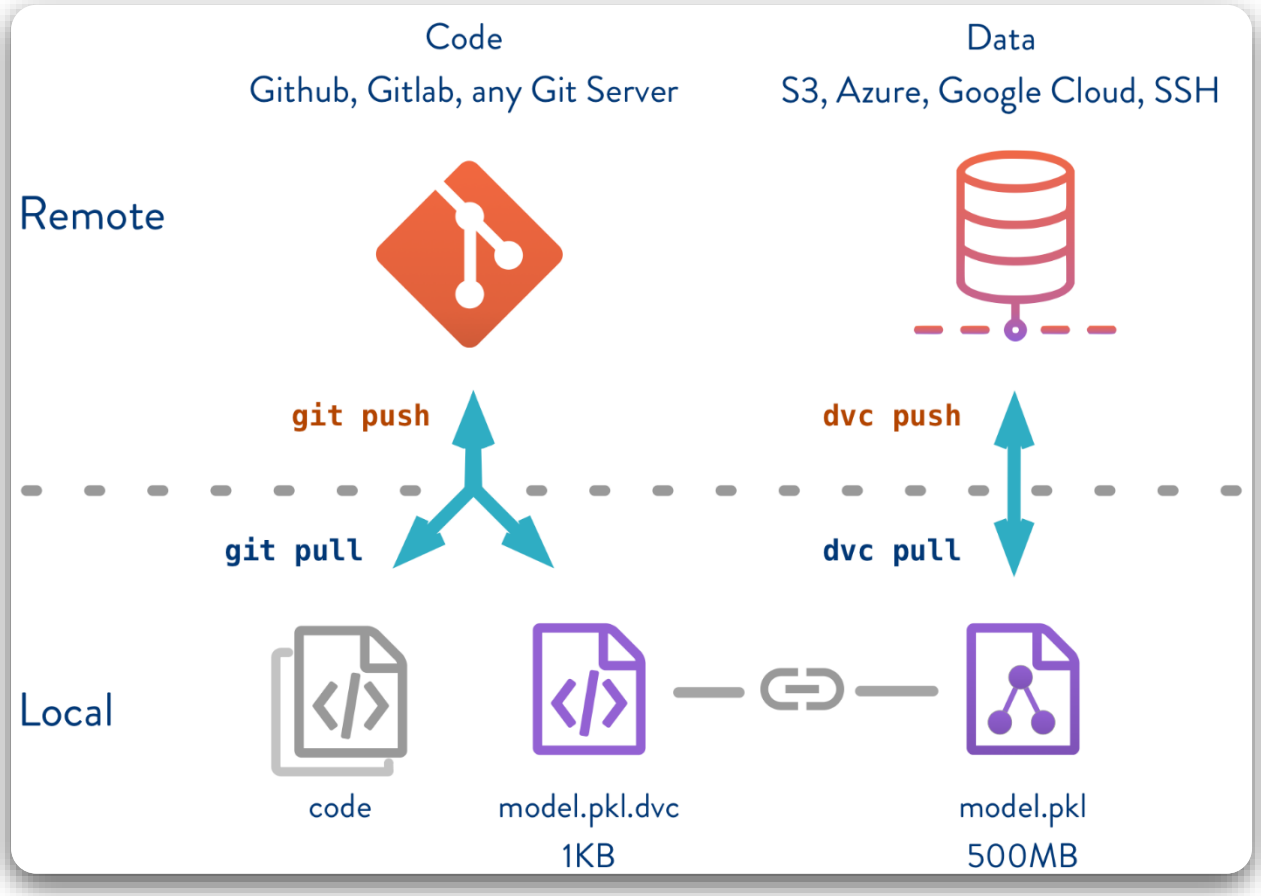


Настройка S3 хранилища



DVC

- Версионирование данных
- Работа со множеством хранилищ (S3, gdrive, local…)
- Работает поверх git
- Обеспечивает работу пайплайнов
- Позволяет контролировать метрики



DVC

- Настраиваем git репозиторий
- Инициализируем DVC

```
$ dvc init
```

- Добавляем файлы в отслеживание

```
$ dvc add data/data.xml
```

- Коммитим в гит

```
$ git add data/data.xml.dvc data/.gitignore
$ git commit -m "Add raw data"
```

DVC

- Добавляем Remote

```
$ dvc remote add -d myremote s3://mystore/path
$ dvc remote modify myremote endpointurl \
                             https://nyc3.digitaloceanspaces.com
```

- Коммитим в git

```
$ git add .dvc/config
$ git commit -m "Configure remote storage"
```

- Пушим в хранилище

```
$ dvc push
```

DVC

- Забираем из хранилища

```
$ dvc pull
```

- Обновляем измененные данные

```
$ dvc add data/data.xml
```

- Пушим изменения

```
$ git commit data/data.xml.dvc -m "Dataset updates"
$ dvc push
```

- Переключаемся между экспериментами

```
$ git checkout <...>
$ dvc checkout
```

DVC

- Настройка пайплайна

```yaml
stages:

  select_region:
    cmd: python src/data/select_region.py data/raw/all_v2.csv data/interim/data_regional.csv 2661
    deps:
      - data/raw/all_v2.csv
      - src/data/select_region.py
    outs:
      - data/interim/data_regional.csv

  clean_data:
    cmd: python src/data/clean_data.py data/interim/data_regional.csv data/interim/data_cleaned.csv
    deps:
      - data/interim/data_regional.csv
      - src/data/clean_data.py
    outs:
      - data/interim/data_cleaned.csv
```

DVC | Что такое

- Воспроизведение пайплайна

```
$ dvc repro
```

- Визуализация DAG

```
$ dvc dag
          +---------+
          | prepare |
          +---------+
               *
               *
               *
        +-----------+
        | featurize |
        +-----------+
               *
               *
               *
         +-------+
         | train |
         +-------+
```

DVC | Что такое

- Запись метрик

```
$ dvc run -n evaluate \
          -d src/evaluate.py -d model.pkl -d data/features \
          -M scores.json \
          --plots-no-cache prc.json \
          --plots-no-cache roc.json \
          python src/evaluate.py model.pkl \
                 data/features scores.json prc.json roc.json
```

- Формат метрик

```
{
  "prc": [
    { "precision": 0.021473008227975116, "recall": 1.0, "threshold": 0.0 },
    ...,
    { "precision": 1.0, "recall": 0.009345794392523364, "threshold": 0.6 }
  ]
}
```

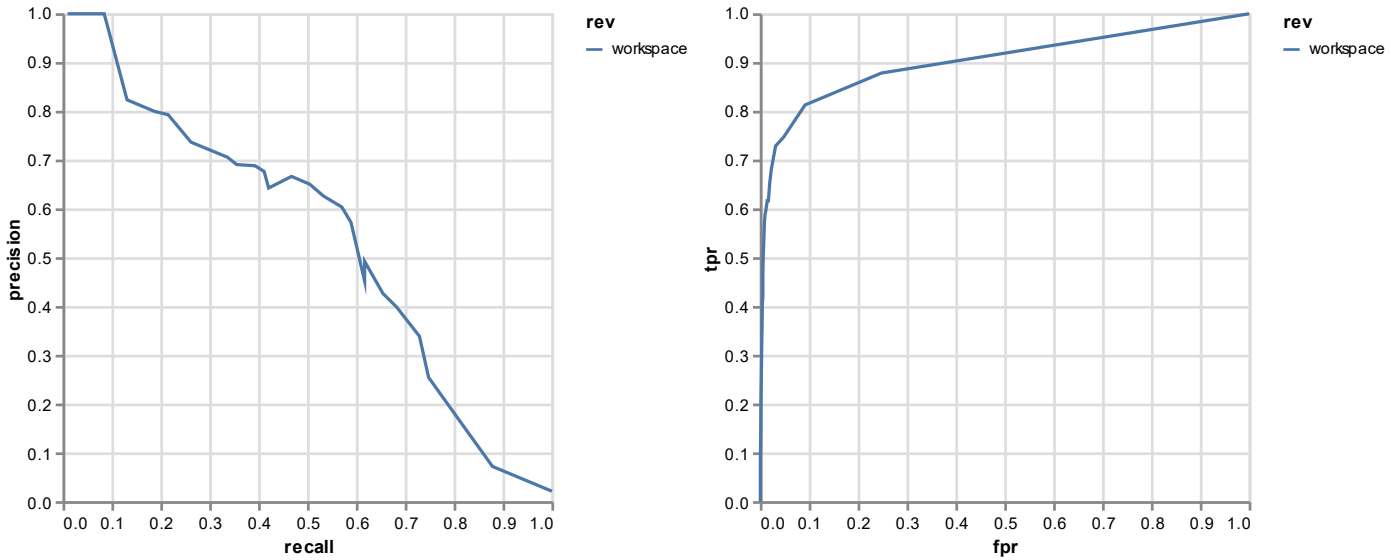
DVC | Что такое

- Вывод метрик

```
$ dvc metrics show
Path         avg_prec    roc_auc
scores.json  0.52048     0.9032
```

- Визуализация метрик

```
$ dvc plots show
file:///Users/dvc/example-get-started/plots.html
```



Яндекс Кью

Open Data Science

Опыт развёртывания ML сервиса в production на примере проекта FindMyBike

Управление зависимостями и инструменты автоматизации на примере DVC.

Антон Ганичев
Газпромнефть ЦР
Руководитель направления NLP
t.me/pavel_kikin