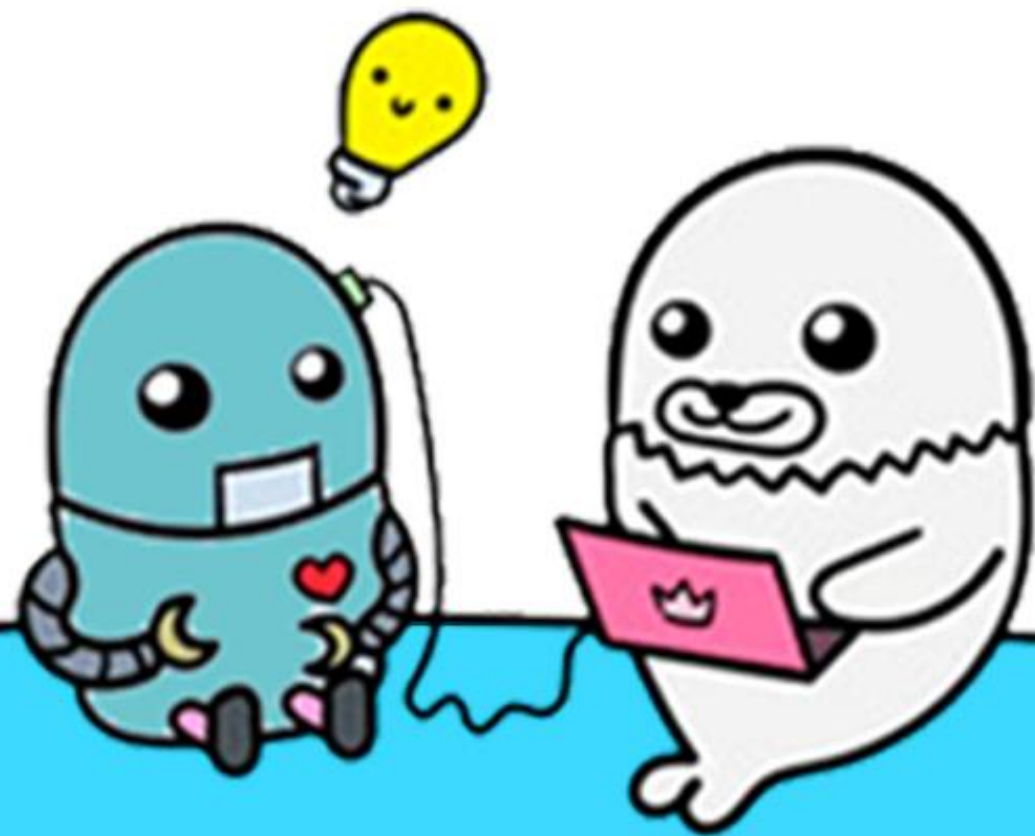


MLOps и production подход к ML исследованиям

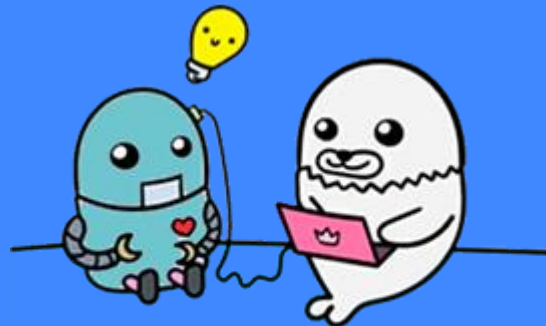


28 марта - 28 мая

Опыт развёртывания ML сервиса в production на примере проекта FindMyBike

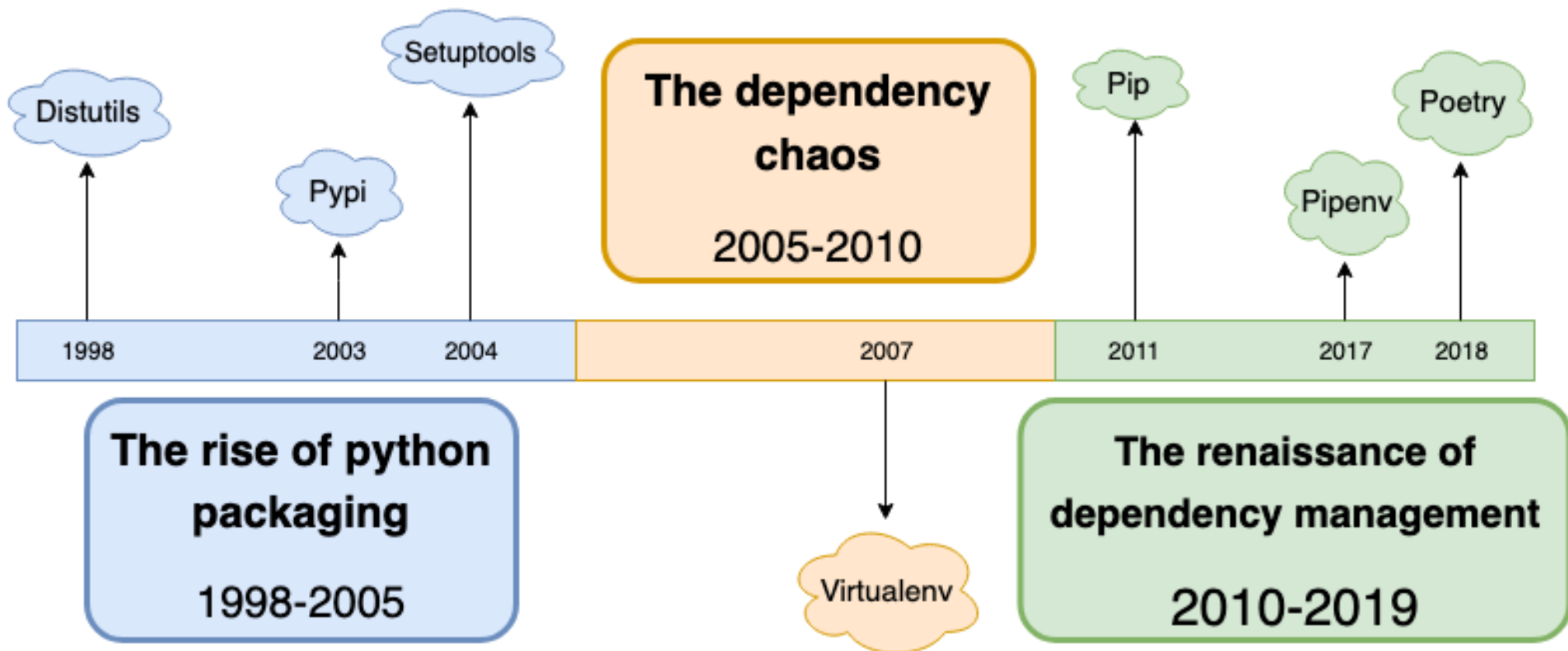
Управление зависимостями и инструменты автоматизации на примере DVC.

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
- Трэйкинг 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



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
```



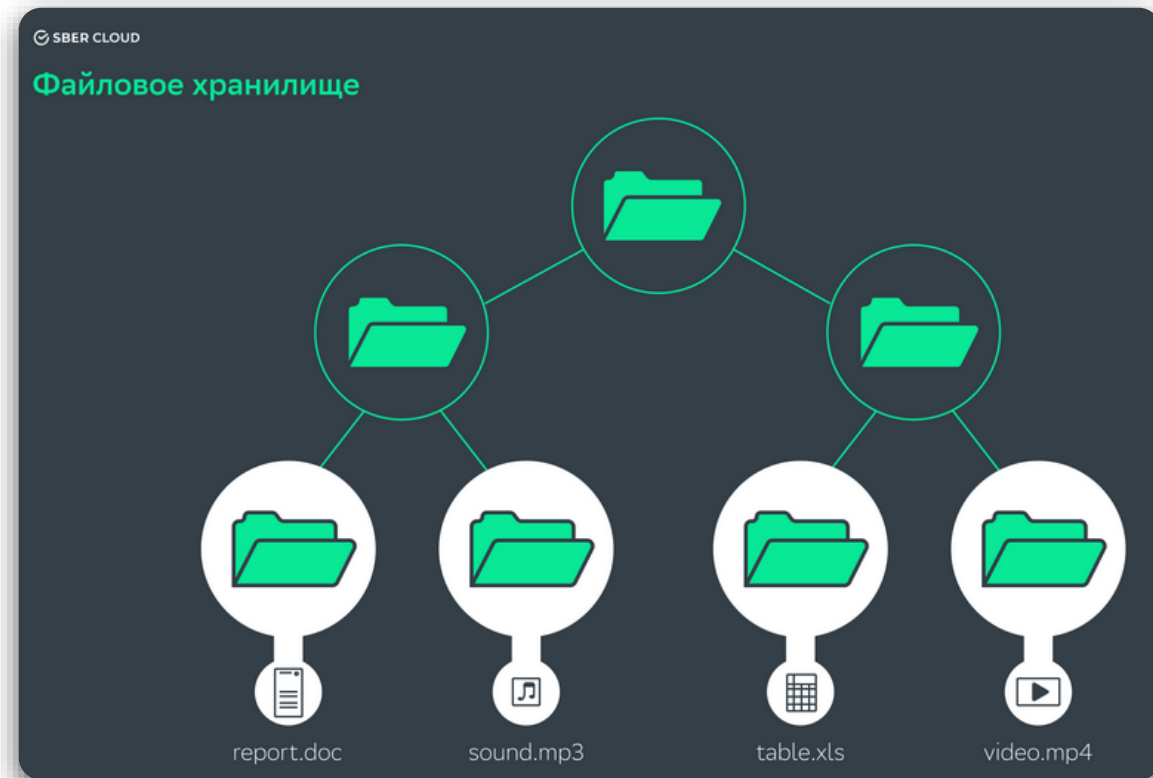

Файловое хранилище

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.





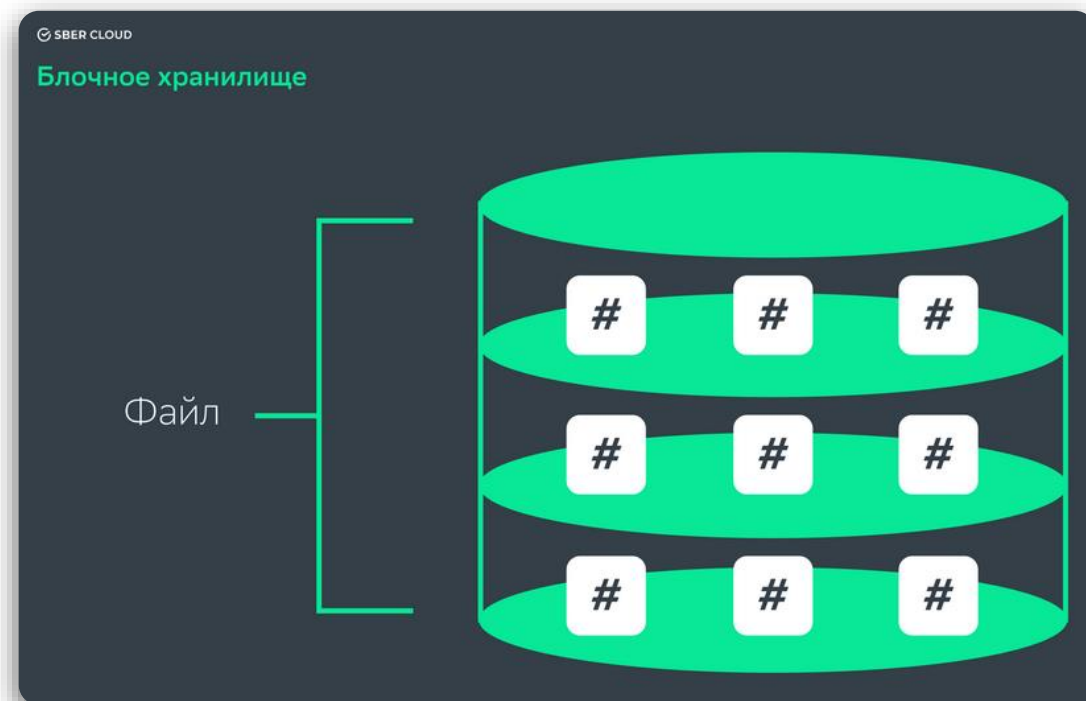
Блочное хранилище

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.





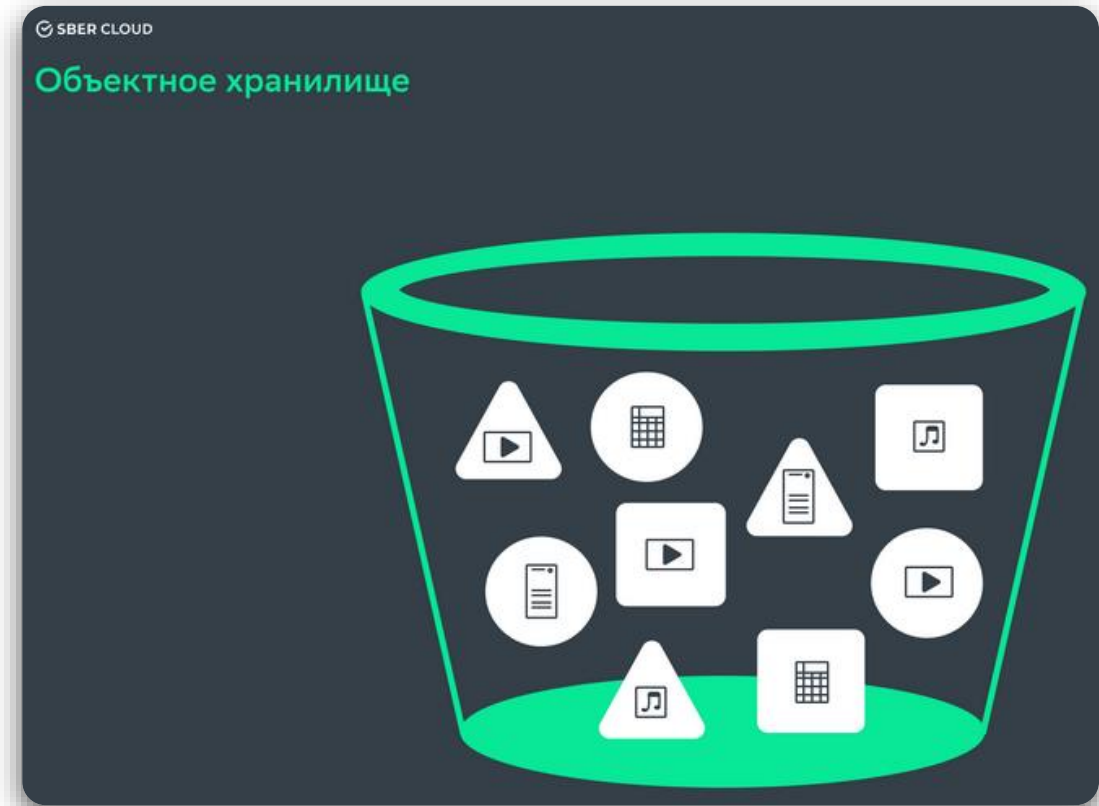
S3 хранилище

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.

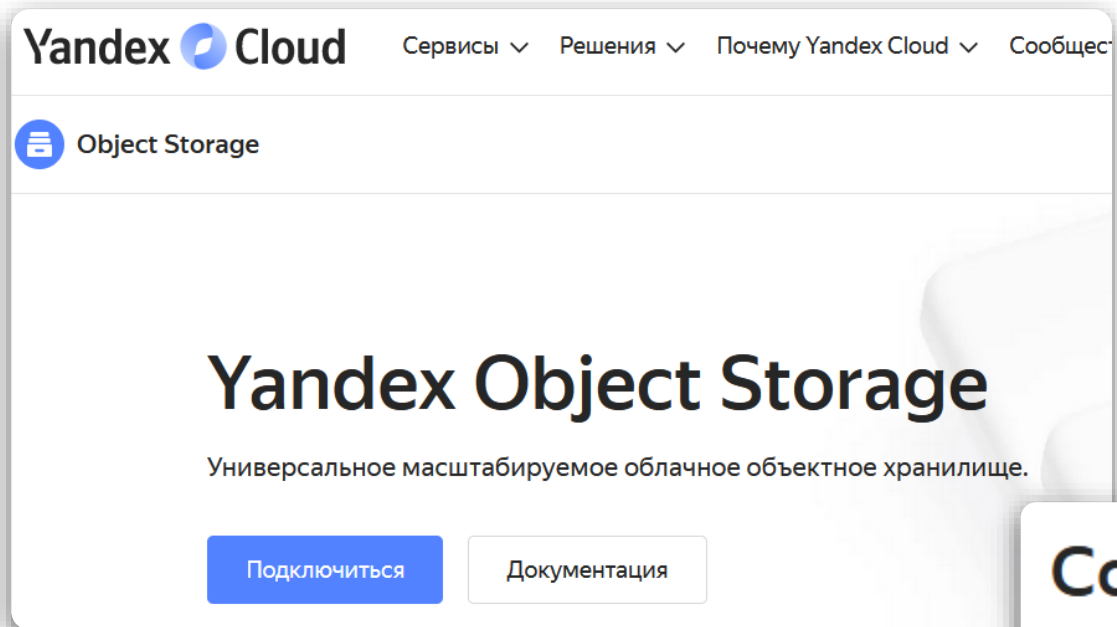
Ограничения:

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





Настройка S3 хранилища



Создайте ваше первое облако

Облако — отдельное рабочее пространство. В нём вы сможете создавать ресурсы, управлять доступом и квотами.

Название облака

Создать



Настройка S3 хранилища

organization-123

cloud-123

default

★

Создайте платёжный аккаунт

Оплачивайте ресурсы, активируйте гранты и просматривайте детализацию использования сервисов

Создать аккаунт

Virtual Private Cloud

1
Сеть

3
Подсети

Сервисы

Compute Cloud	>	Managed Service for PostgreSQL	>	Managed Service for MongoDB
Managed Service for Redis	>	Managed Service for MySQL	>	Managed Service for Kafka
Managed Service for Elasticsearch NEW	>	Managed Service for SQL Server	>	Managed Service for Greenplum NEW
Managed Service for ClickHouse	>	Object Storage	>	Virtual Private Cloud



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

cloud-123 DE default

Object Storage / Бакеты / Новый бакет

Новый бакет

Имя ?

mlops

Макс. размер

50

ГБ

☐ Без ограничения

Доступ на чтение объектов ?

Ограниченный

Публичный

Доступ к списку объектов ?

Ограниченный

Публичный

Доступ на чтение настроек ?

Ограниченный

Публичный

Класс хранилища ?

Стандартное

Холодное

Создать бакет



Настройка S3 хранилища

< Облако

mlops-course

Каталог

Дашборд каталога

Сервисные аккаунты

Федерации

Уведомления об инцидентах

Права доступа

Операции

Дашборд каталога

Создать ресурс

Object Storage



1

568.84 МБ

Бакет

Занятое место

Virtual Private Cloud



1

3

Сеть

Подсети

Сервисы



Compute Cloud



Managed Service for PostgreSQL



Managed Service for MongoDB



Managed Service for Redis



Managed Service for MySQL



Managed Service for Kafka



Настройка S3 хранилища

☰

Yandex Cloud

Поиск по облачным ресурсам

ML

mlops-course

mlops

?

🔔

⚙️

👤

< Каталог

Object Storage

Сервис

📁 Бакеты

Бакеты

Имя бакета

Доступ —

Имя	Размер	Кол-во объектов	⚙️
dvc	568.84 МБ	15 объектов	⋮

Создать бакет



Настройка S3 хранилища

< Облако

mlops-course

Каталог

Дашборд каталога

Сервисные аккаунты

Федерации

Уведомления об инцидентах

Права доступа

Операции

Дашборд каталога

Создать ресурс

Object Storage

1568.84 МБ

БакетЗанятое место

Virtual Private Cloud

13

СетьПодсети

Сервисы

Compute Cloud

Managed Service for PostgreSQL

Managed Service for MongoDB

Managed Service for Redis

Managed Service for MySQL

Managed Service for Kafka



Настройка S3 хранилища

☰

Yandex Cloud

Поиск по облачным ресурсам

ML

mlops-course

mlops

?

🔔

⚙️

< Облако

mlops-course

Каталог

📊 Дашборд каталога

🔧 Сервисные аккаунты

👤 Федерации

✉️ Уведомления об инцидентах

👥 Права доступа

⚙️ Операции

Документация

Тарифы Compute Cloud

Синтез речи в SpeechKit

Создать виртуальную машину

Сервисные аккаунты

Создать сервисный аккаунт

Фильтр по имени или идентификатору

Имя
mlops-developer

Создание сервисного аккаунта

✕

Имя ?

mlops-developer

Описание ?

Роли в каталоге

+ Добавить роль

stor|

storage.admin

storage.configViewer

storage.configurer

storage.editor

storage.uploader

storage.viewer

Отменить

Создать



Настройка S3 хранилища

☰

Yandex Cloud

Поиск по облачным ресурсам

ML

< Каталог

Сервисный аккаунт

Обзор

Мониторинг

Права доступа

Обзор

Идентификатор ajenhns1bos5qki954s7

Имя mlops-developer

Дата создания 23 апреля 2022, в 15:57

Роли в каталоге storage.editor

API-ключи

+ Создать новый ключ

Создать статический ключ доступа
Для Object Storage, Message Queue
и Yandex Database

Создать API-ключ
Для упрощенной аутентификации,
вместо IAM-токена

Создать авторизованный ключ
Для запроса IAM-токена



Настройка S3 хранилища

Новый ключ



Идентификатор ключа:

YCAJEMqXHQDXkde5SOmJJJdna

Ваш секретный ключ:

YCPOTn4Om8Obl9XtCB0FVnMK7DorXUTm03zP81sB



Сохраните идентификатор и ключ. После закрытия диалога значение ключа будет недоступно.

Закрыть



Настройка S3 хранилища

Yandex Cloud

Поиск по облачным ресурсам

Консоль управления

Compute Cloud

Managed Service for PostgreSQL

Managed Service for MongoDB

Managed Service for ClickHouse

Managed Service for Redis

Managed Service for MySQL

Managed Service for Kafka

Managed Service for Elasticsearch

Managed Service for SQL Server

Managed Service for Greenplum

Managed Service for Kubernetes

Managed Service for GitLab

Data Proc

Yandex Database

Object Storage

Ваши ресурсы ?

Перейти в текущий каталог

MLOps

Статус

Пробный период

Баланс

1000 ₽ / 3000 ₽

Осталось

56 дней

Перейти на платную версию

mlops

Павел Кикин и компания

mlops-course

b1g2qrs65pqon6j2g89i

b1gtvrvp95ao19q2o15i

А знаете ли вы

Как загрузить файлы в объектное хранилище за 4 шага

1. Выберите **Object Storage** в списке сервисов.

2. Нажмите **Создать бакет**.

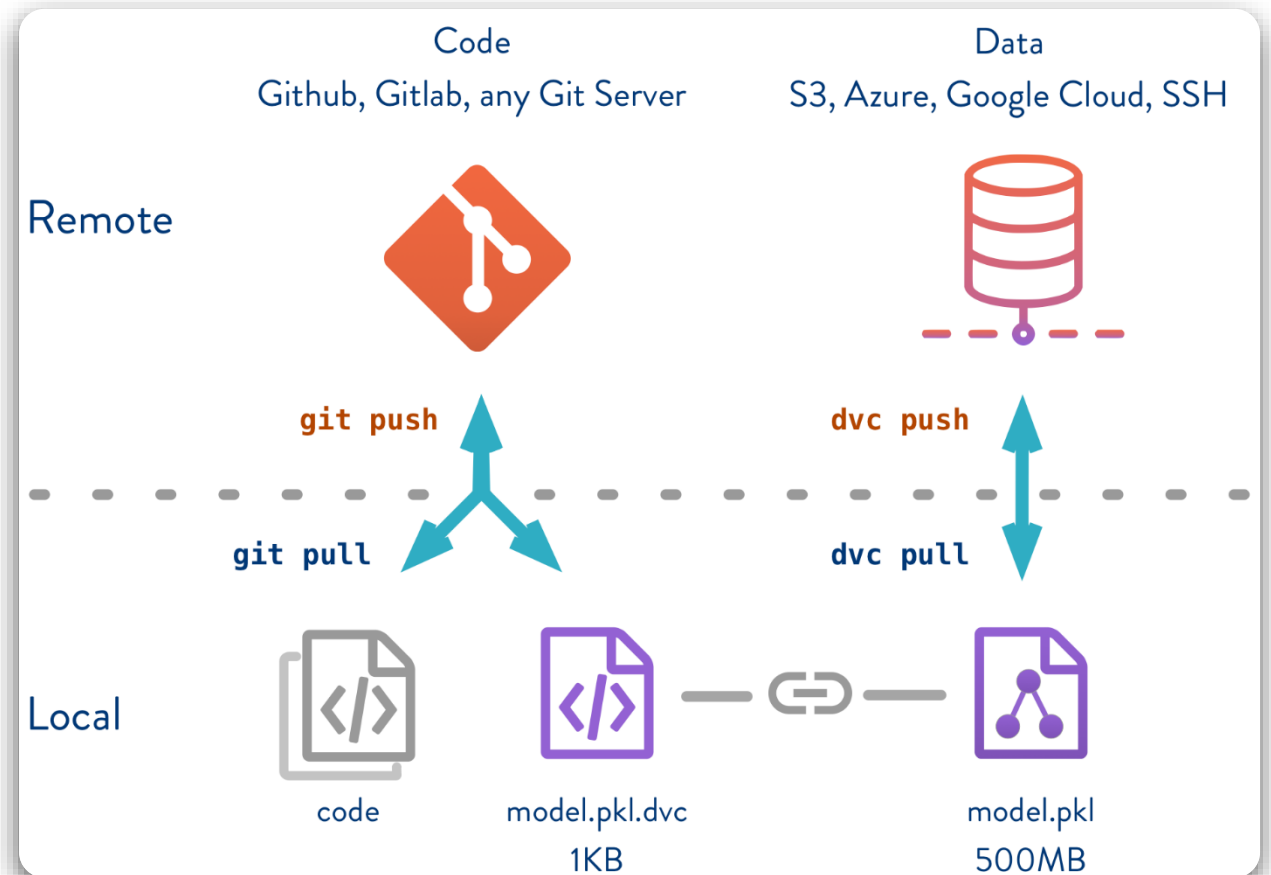
3. Укажите параметры бакета и нажмите **Создать бакет**.

4. Нажмите **Загрузить** и выберите файлы для размещения в Yandex.Cloud.



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

```
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 |
+-----+
```



- 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 }  
  ]  
}
```



- Вывод метрик

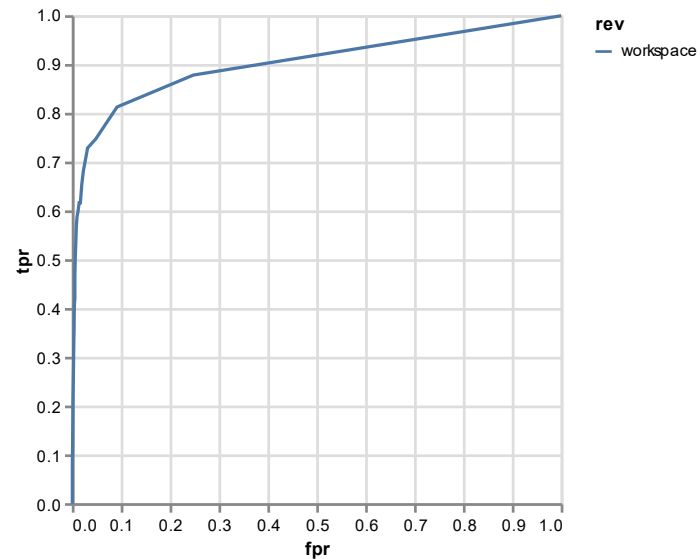
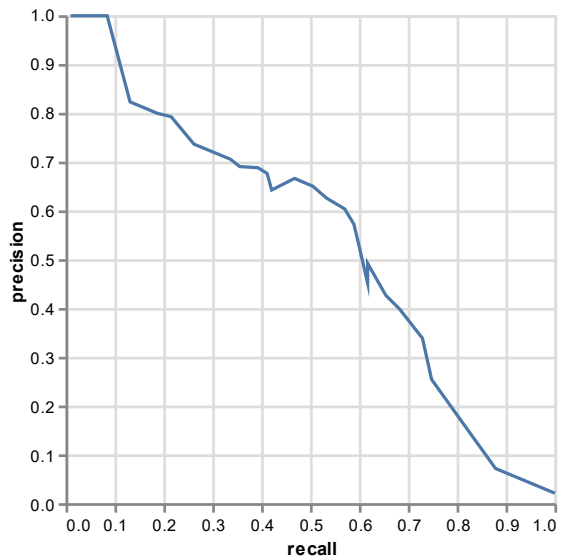
```
$ dvc metrics show
```

Path	avg_prec	roc_auc
scores.json	0.52048	0.9032

- Визуализация метрик

```
$ dvc plots show
```

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



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

