Introduction to MLOps

Vladislav Goncharenko

girafe ai

HSE, fall 2023

Владислав Гончаренко

- Автор курсов по машинному обучению и магистерской программы в МФТИ
- Исследователь в области машинного обучения
- Руководитель команды ранжирования видео в Дзене
- ex-руководитель команды восприятия в беспилотных грузовиках
- фанат open source

girafe ai

Дзен

Preface

girafe ai

00

Prerequisites

- UNIX command line basics
- Python programming experience
  - web services
- Basic machine learning knowledge
  - pytorch

Your part

1. Knowledge level of each student is different
2. I adjust materials based on your knowledge
3. Your knowledge is estimated by feedback

More you ask - more you get from this course!

I can give you a way but you need to make an effort to follow it

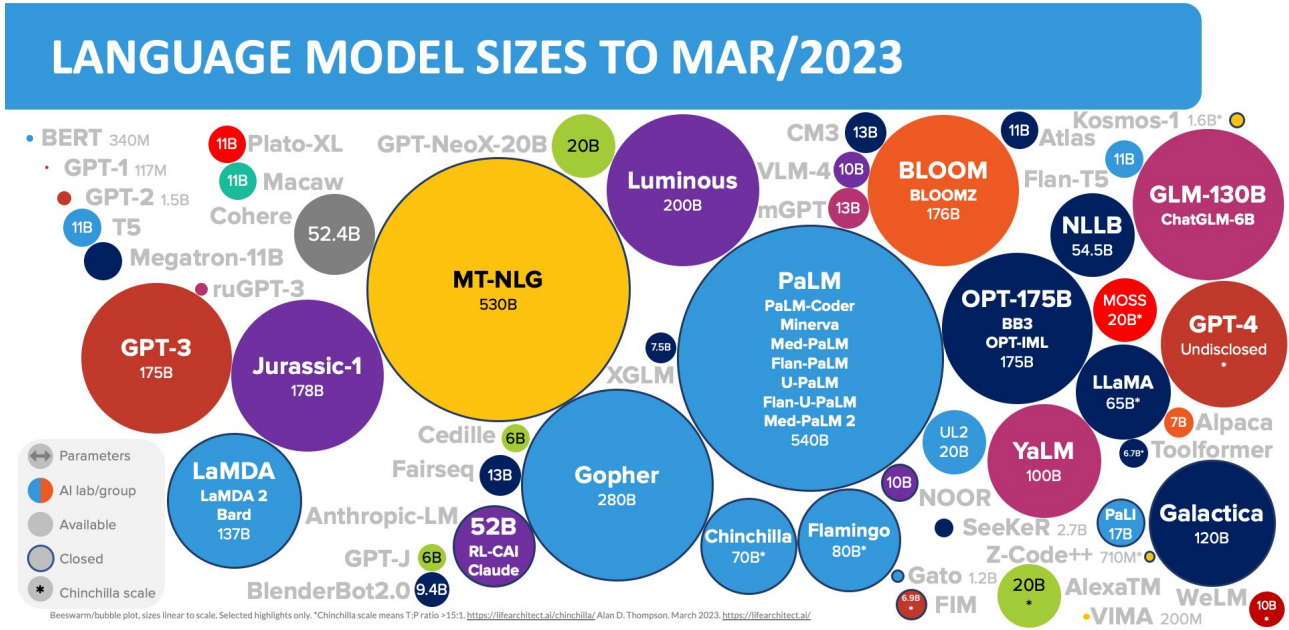
Motivation for MLOps

girafe
ai

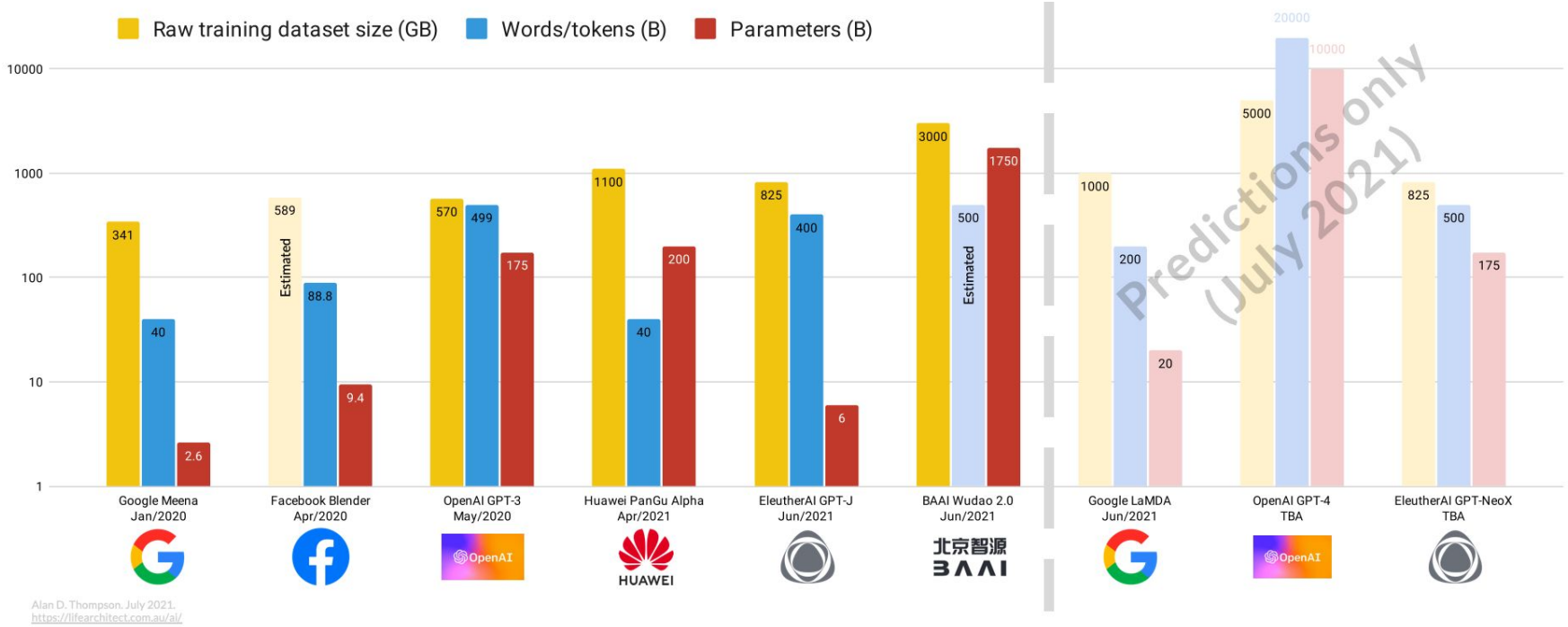
01

Зачем нужен MLOps?

- Ресурсы, затрачиваемые на разработку моделей, всё растут
  - майнинг данных (cpu)
  - разметка данных (люди)
  - обучение моделей (gpu)



LANGUAGE MODEL SIZES & PREDICTIONS (JUL/2021)



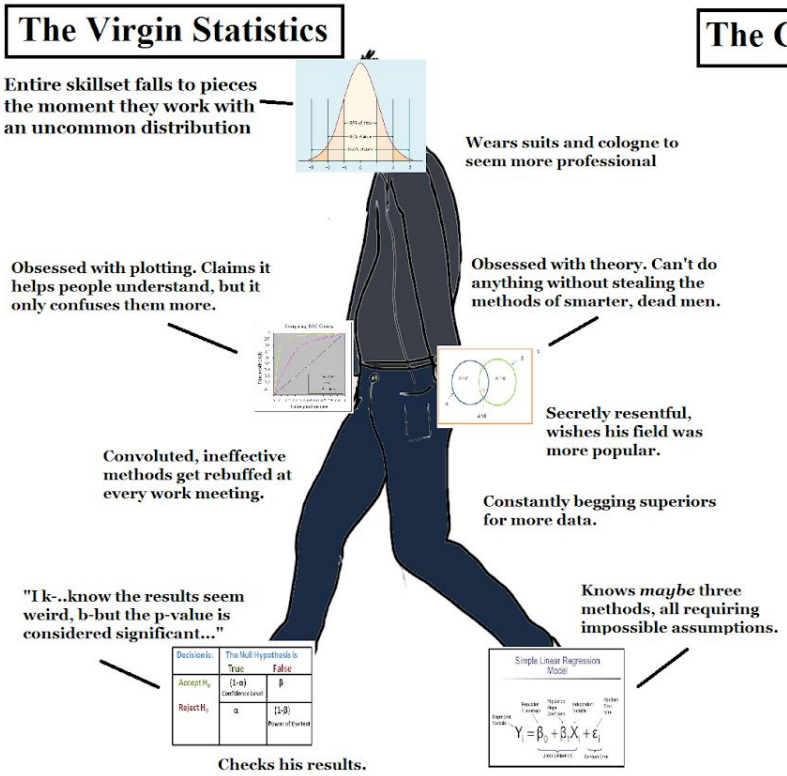
Зачем нужен MLOps?

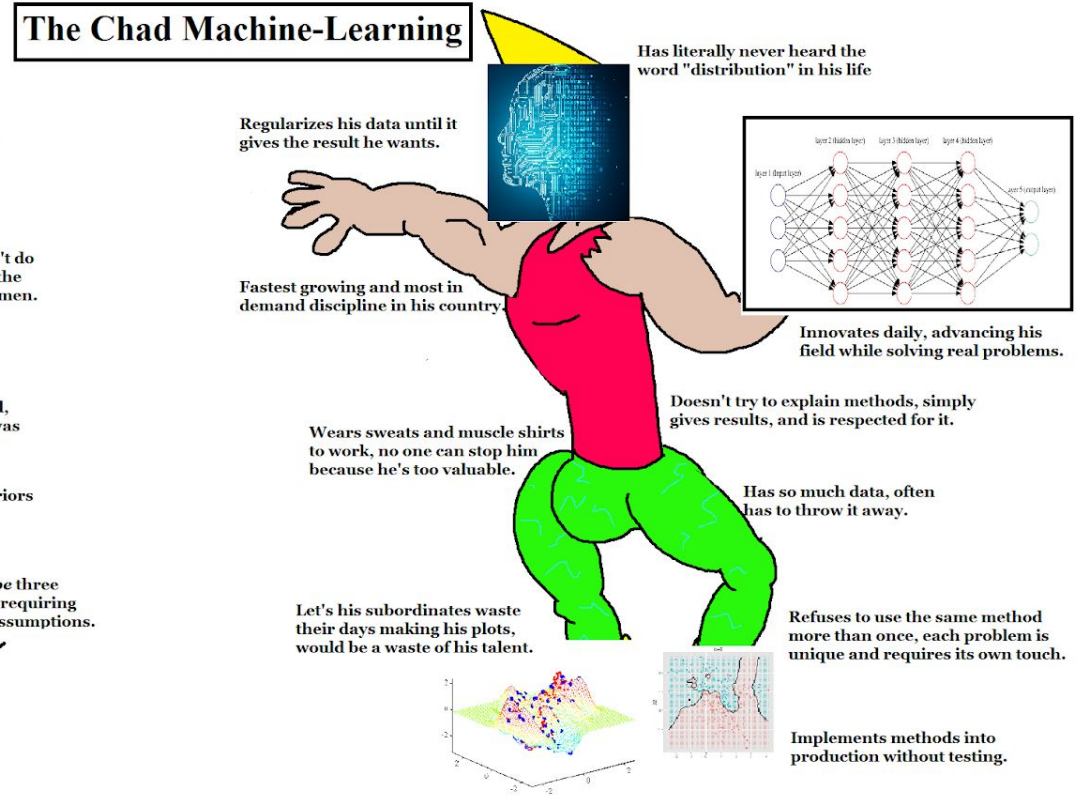
- Ресурсы, затрачиваемые на разработку моделей, всё растут
  - майнинг данных (cpu)
  - разметка данных (люди)
  - обучение моделей (gpu)
- Воспроизводимость тренировок
  - не только в индустрии, но и в исследованиях

Зачем нужен MLOps?

- Ресурсы, затрачиваемые на разработку моделей, всё растут
  - майнинг данных (cpu)
  - разметка данных (люди)
  - обучение моделей (gpu)
- Воспроизводимость тренировок
  - не только в индустрии, но и в исследованиях
- Организация доставки
  - сократить time-to-market
  - исключить рутину
- Декомпозиция компетенций
  - более глубокое разделение труда

Зачем MLOps МЛщикам?





Зачем MLOps МЛщикам?

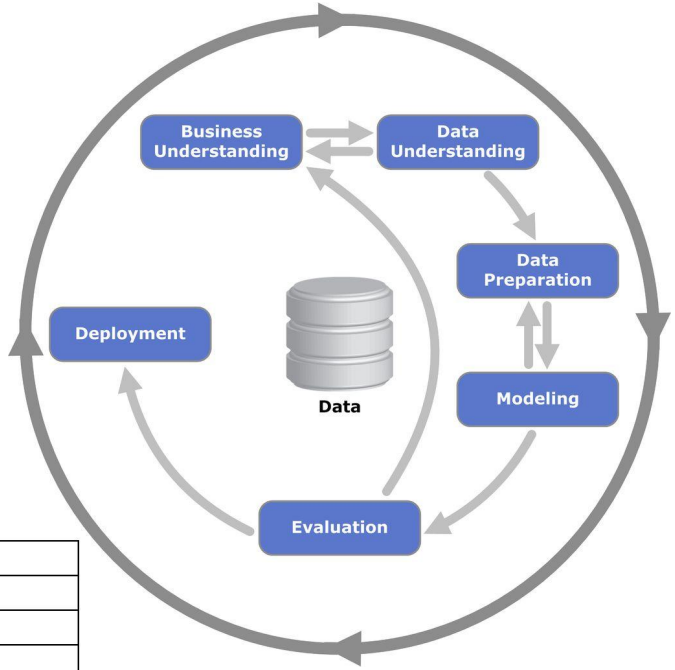
потому что никто другой для вас её не построит!!!

Для приобщения к теме проектирования систем рекомендую system design primer

CRISP DM

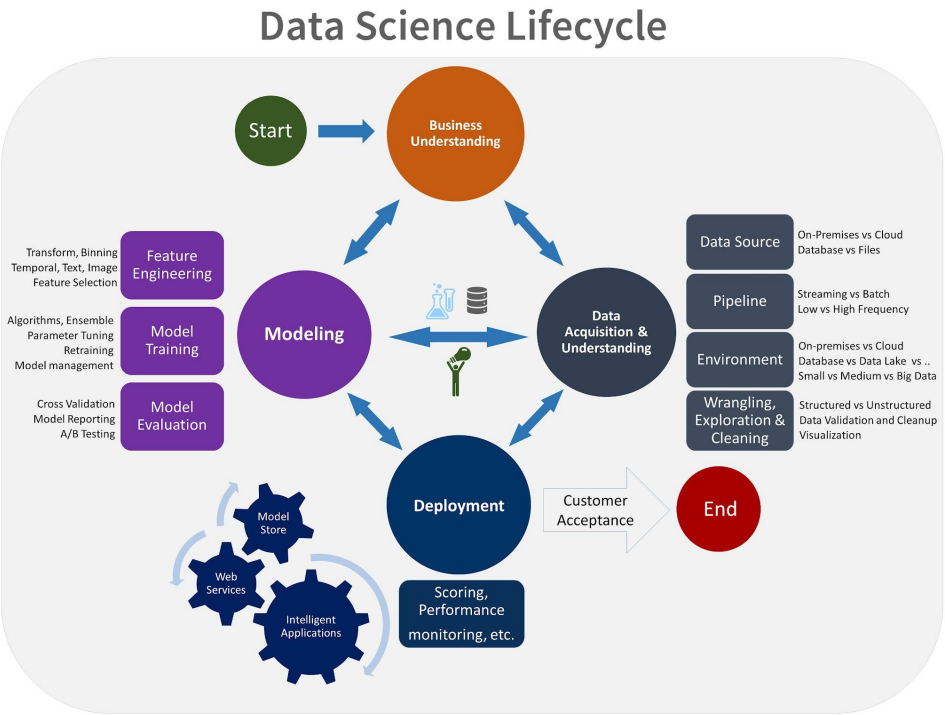
In the field of Data Science is used. Cross-industry standard process for data mining (CRISP DM) is a process standard for projects using predictive analytics, proposed in 1999. It was used by half of the companies engaged in Data Mining at the beginning of the century.

The development of this direction was the approaches of Knowledge Discovery in Databases (KDD) and SEMMA.



Poll Years	2002	2004	2007	2014
CRISP-DM	51%	42%	42%	43%
SEMMA	12%	10%	13%	8.5%
KDD process			7%	7.5%
My organization's	7%	6%	5%	3.5%
My own	23%	28%	19%	27.5%
Other (incl. domain specific)	4%	6%	9% (5%)	10% (2%)
None	4%	7%	5%	0%

TDSP



The most modern Team Data Science Process (TDSP) from Microsoft

It is distinguished by the allocation of roles and a more detailed study of the stages of the cycle

Other standards:

- SEMMA, link
- KDD Process

link_01, link_02

Что такое MLOps?

girafe ai

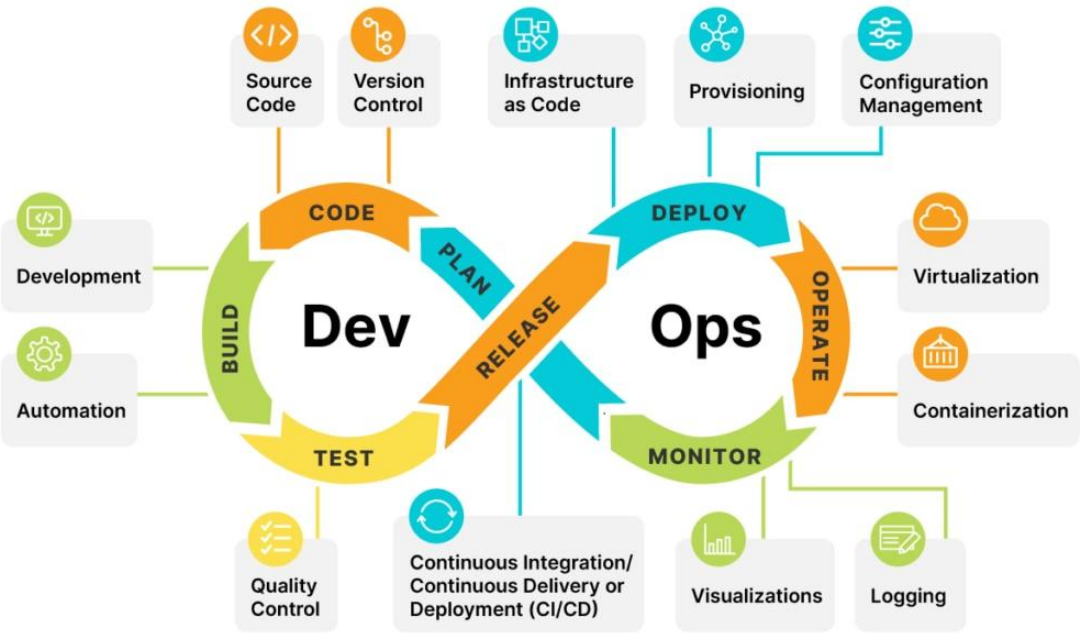
02

Определение DevOps

DevOps is a methodology in the software development and IT industry. Used as a set of practices and tools.

DevOps integrates and automates the work of software development (Dev) and IT operations (Ops) as a means for improving and shortening the systems development life cycle

[Well-known common knowledge site]

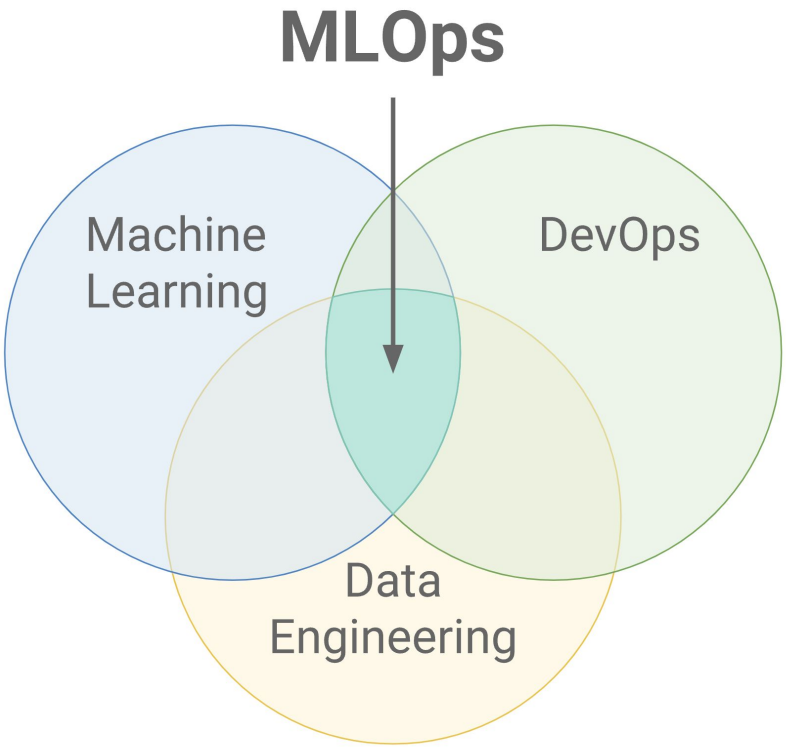


Определение MLOps

MLOps is a paradigm that aims to deploy and maintain machine learning models in production reliably and efficiently.

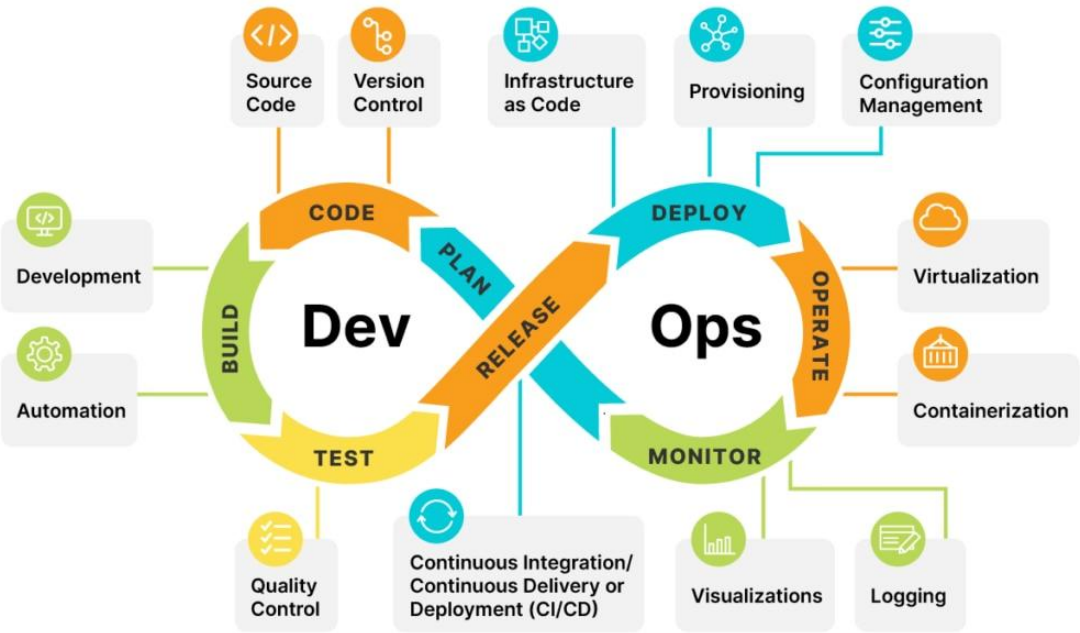
MLOps seeks to increase automation and improve the quality of production models, while also focusing on business and regulatory requirements

Well-known common knowledge site



Связь с другими дисциплинами

- SQL
- Аналитика
- Базы данных
- Machine Learning

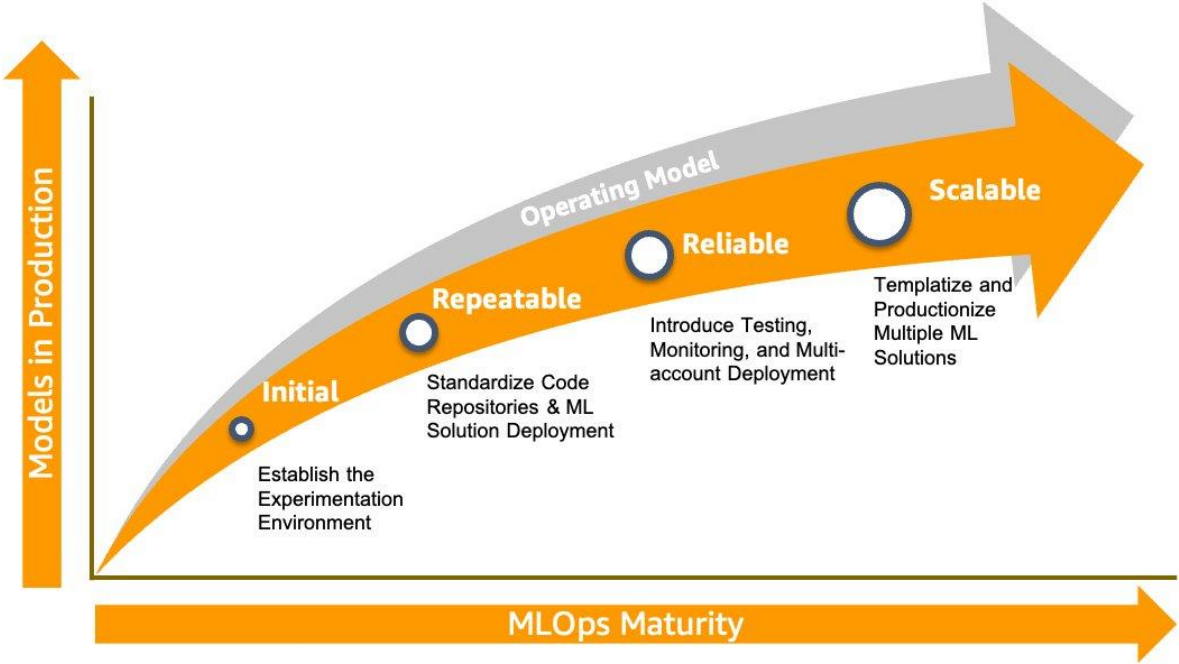


MLOps at big tech

Big companies has their stack to organize models training and publishing pipelines

- Amazon
- Google
- Nvidia
- Yandex
- Databricks

Will dig into them later today



Even Gucci!

Youtube video

MLOps subtasks

Course structure

girafe ai

03

MLOps subtasks

- Хранение и управление кодовой базой
  - Git
  - CI & CD
  - code quality
  - Тестирование (автотесты, тестовые стенды)
  - генерация доки
- Хранение и управление данными
  - Data Engineering
  - АБ тесты

MLOps subtasks

- Хранение и управление кодовой базой
- Хранение и управление данными
- Логирование экспериментов и продакшена
- Вычислительные мощности
  - GPU
- Хранение и оптимизация работы моделей
- Эффективное применение моделей
- Оптимизация весов эмбедингов и поиска ближайших соседей
- Разметка данных и краудсорс
- Обзор готовых решений
  - Amazon stack
  - Yandex cloud
  - ...

MLOps stacks overview

girafe
ai

04

MLOps at big tech

- Amazon https://aws.amazon.com/sagemaker/mlops/
- Google https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning
- Nvidia https://blogs.nvidia.com/blog/2020/09/03/what-is-mlops/
- Microsoft Azure
- Yandex: YTSaurus https://ytsaurus.tech/ + Nirvana + Toloka https://toloka.ai/
- Databricks https://www.databricks.com/glossary/mlops

Generally variance of instruments is huge over different companies (from only bare metal ssh machines to very solid custom solutions)

MLOps at academia

It just doesn’t exist

You publish or perish:
no time for bullshit

What about open source stack?

Will discuss it during all the course.

Main problem of FOSS instruments - lack of orchestration.

Each instrument primarily aims to solve one task but then start to add some default solutions for adjacent problems which are not that great.

All in all no instrument solve every problem.
Finally you need to write glue code yourself.

Revise

- Зачем нужен MLOps
- Стандарты разработки Data Science проектов
- Что такое MLOps
- Подзадачи MLOps
- Обзор решений от крупных компаний
- Проблемы open source стека

Thanks for attention!

Questions?

girafe ai

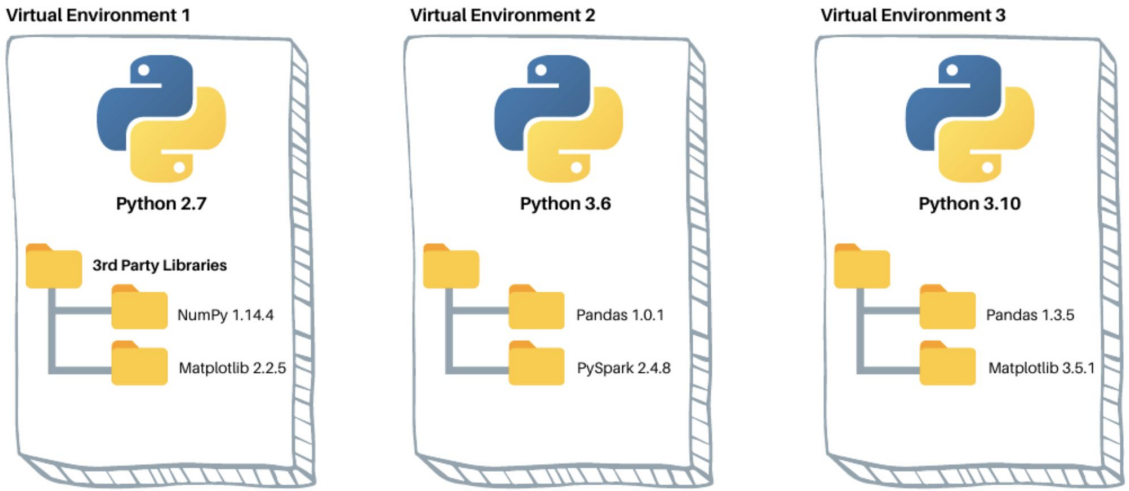
Python environments

girafe ai

05

What do we want from environments?

- Create environments
  - separate for each project on one host machine
- Remember and install packages
  - repeatable set of packages for each project



Vanilla solution

- Create environments
  - `python -m venv /path/to/new/virtual/environment`
- Remember and install packages
  - `pip freeze > requirements.txt`
  - `pip install -r requirements.txt`

Any problems with this?

- only python dependencies managed
- python version not saved
- long and structureless requirements file
- no groups of dependencies (prod vs dev)
- not portable to all OSes

Managing environments

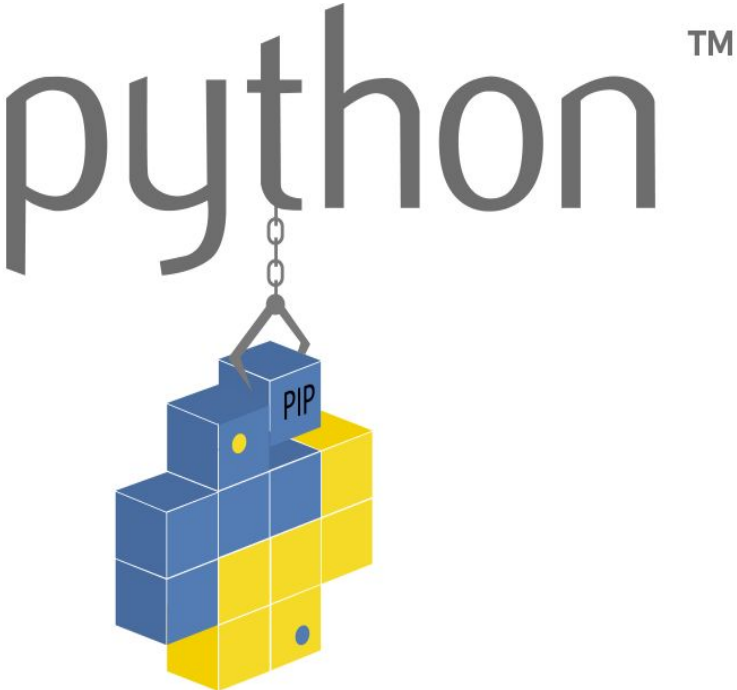
- venv - Python standard library
- virtualenv - extended toolkit for virtual environments
- poetry
- conda - able to install non python dependencies
  - miniconda
  - mamba
- pyenv - automatic envs switching (good article)
- pipx
- pipenv

More on virtual envs

Managing dependencies

- pip (via requirements.txt)
- poetry - convenient tool covering most needs
- pip-tools
- Pipfile (via pipenv)

Alternative is docker container



Modules and packaging

- python package structure
  - common extensible format
  - able to be packaged
- build systems
  - setuptools
  - poetry
  - flit
  - ...

More on python packaging

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/                  Subpackage for sound effects
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/                  Subpackage for filters
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

pyproject.toml

Standard file format for modern package metadata

Packaging user guide, Pip documentation on pyproject

Set of tools supporting pyproject.toml

```toml
[project]
name = "mypackage"
version = "2023.05.1"
description = "A package for demonstrating Python packaging"

[build-system]
requires = ["setuptools >= 61.0.0"]
build-backend = "setuptools.build_meta"

[tool.setuptools.packages.find]
where = ["src"]
```

poetry capabilities

- managing environments
- dependencies for all platforms simultaneously
- better conflicts resolving

One tool for all - poetry

- dares to install ONE tool per OS
- better conflicts resolving

Creates two files:

- `pyproject.toml` - initial dependencies, config for all tools
- `poetry.lock` - all concrete dependencies (platform agnostic)

If using conda environments:

```
poetry config virtualenvs.create false
```

Also it is able to build and publish to PyPI or your own package registry

```
poetry build && poetry publish
```

poetry usage

- managing environments
- dependencies for all platforms simultaniously
- better conflicts resolving
- additional package indexes
-

One tool for all - poetry

- dares to install ONE tool per OS
- better conflicts resolving

Creates two files:

- `pyproject.toml` - initial dependencies, config for all tools
- `poetry.lock` - all concrete dependencies (platform agnostic)

poetry

One tool for all - poetry

- dares to install ONE tool per OS
- better conflicts resolving

Creates two files:

- `pyproject.toml` - initial dependencies, config for all tools
- `poetry.lock` - all concrete dependencies (platform agnostic)

If using conda environments:

```
poetry config virtualenvs.create false
```

For installing dependencies:

```
poetry install
```

Also it is able to build and publish to PyPI or your own package registry

```
poetry build && poetry publish
```

Command Line Interfaces (CLI)

Default tool for CLI in Python is argparse.
Which is clone of C library and is painful to use.

Also there are several tools made for humans.

One of them is Fire.
It makes CLI out of pure Python functions and classes.

```
import fire

def hello(name="World"):
  return "Hello %s!" % name

if __name__ == '__main__':
  fire.Fire(hello)
```

Then, from the command line, you can run:

```
python hello.py  # Hello World!
python hello.py --name=David  # Hello David!
python hello.py --help  # Shows usage information.
```