Machine Learning Operations

02476 Machine Learning Operations

Nicki Skafte Detlefsen,

Postdoc

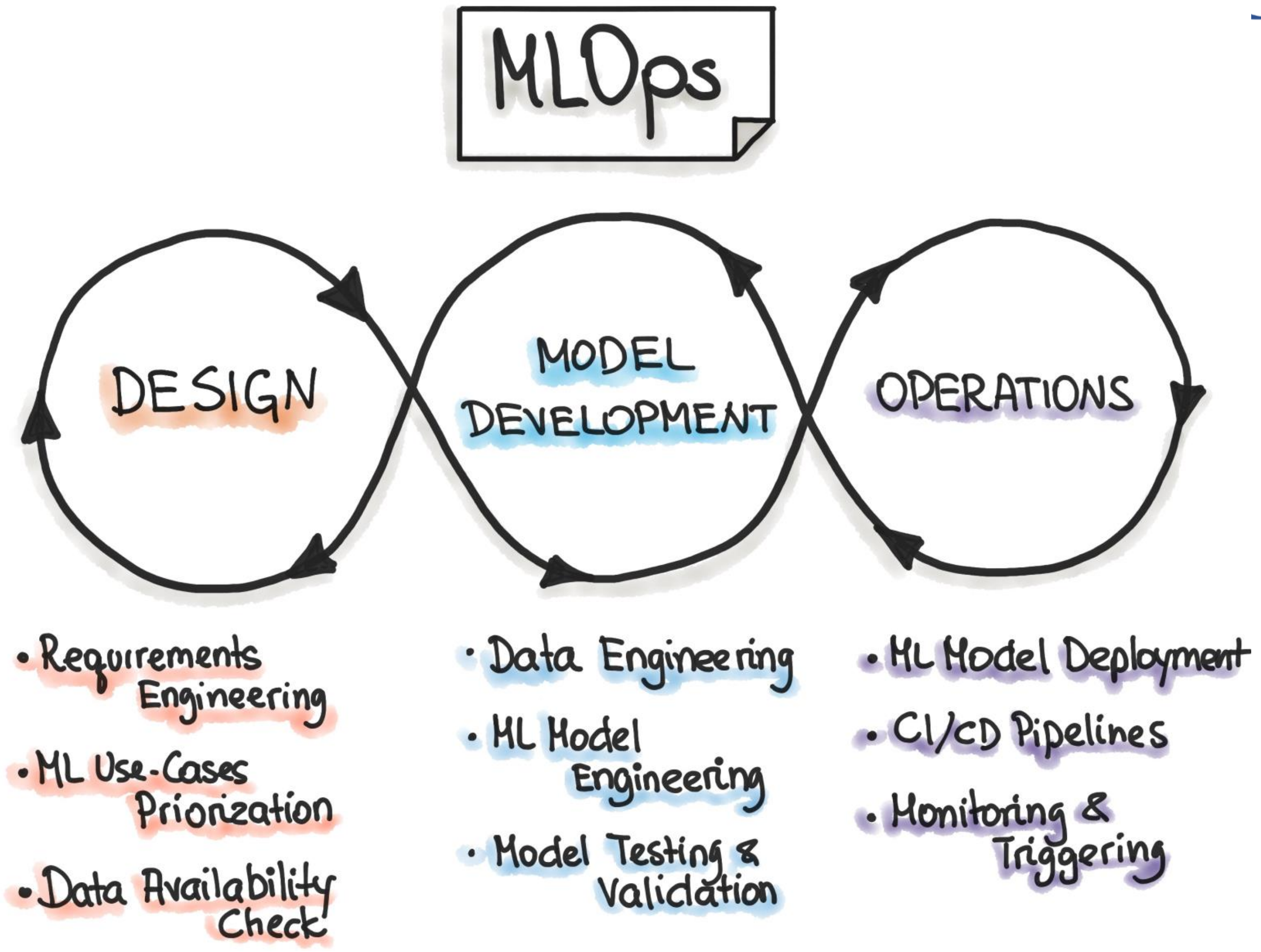
DTU Compute

Loosely based on https://www.youtube.com/watch?v=VU5Em1qkWDU

What is machine learning operations

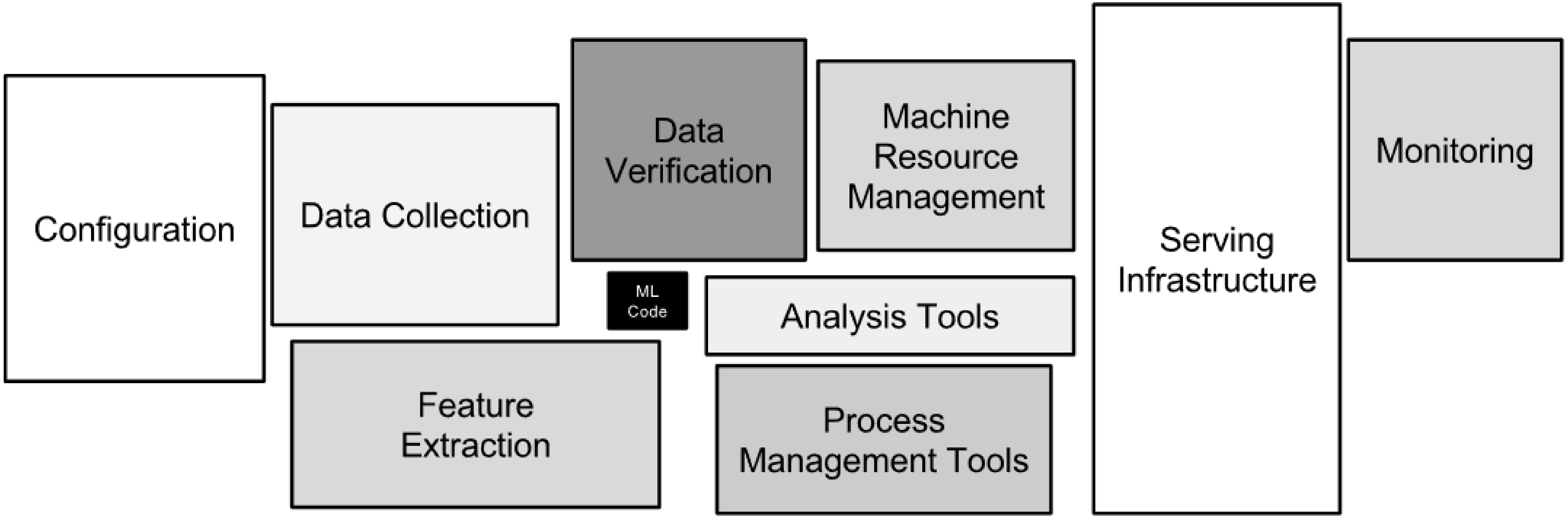
*Is a set of <u>tools</u>, <u>processes</u>, and <u>mindset</u> that aim to make ML Lifecycle **reproducible**, **trackable**, **testable** and **maintainable***

Notice: ITS A CYCLE!



Why should you care?

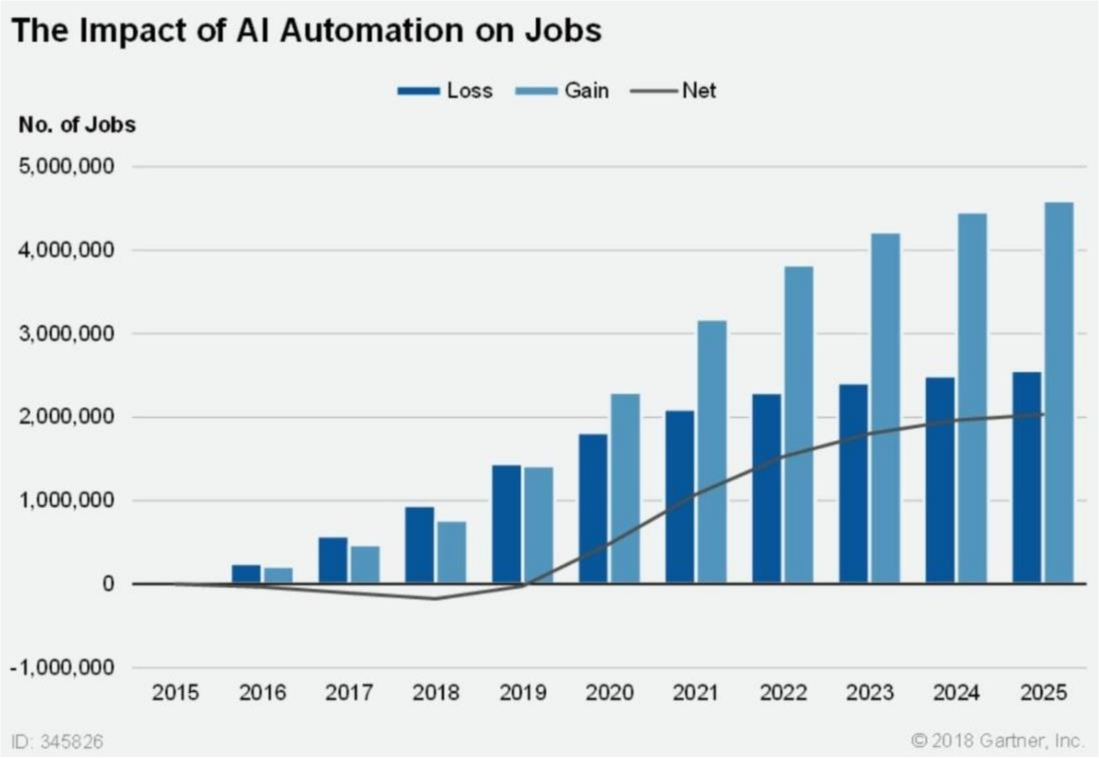
Teeny tiny part is actual ML code, the rest is operations



D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. **Hidden technical debt in Machine learning systems**. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2* (*NIPS'15*). MIT Press, Cambridge, MA, USA, 2503–2511.

Why does companies care

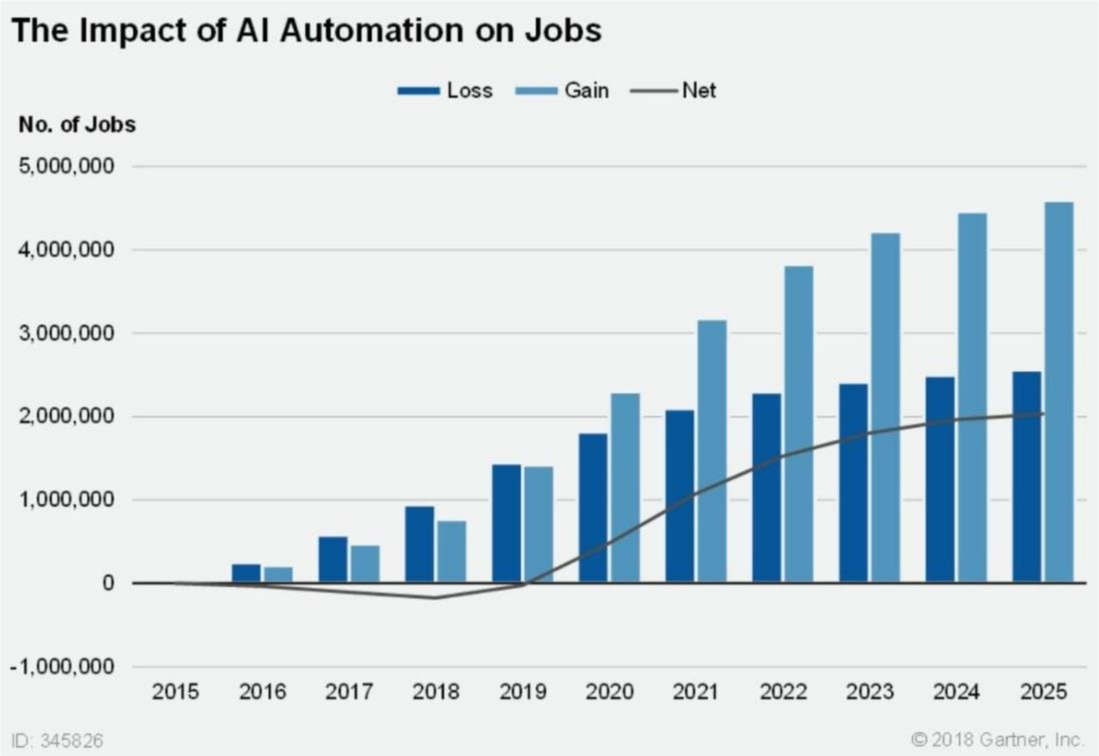
- ML automatization is going to increase over the years
- Examples:
  - Which stocks to buy or sell?
  - Where is the tumor in the picture
  - What should be the price of a banana today?



Why does companies care

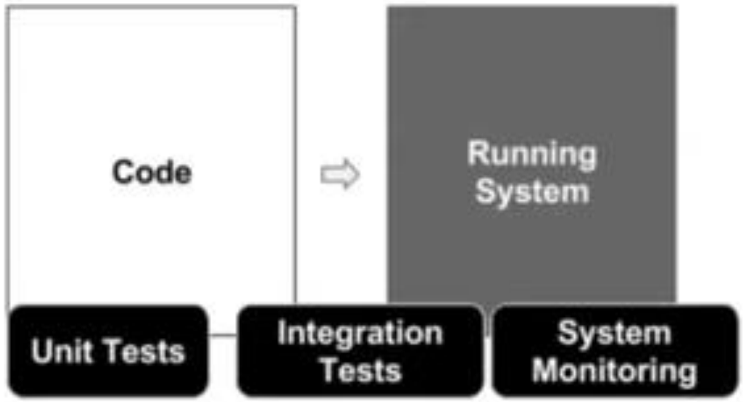
- Having automated model deployed with errors can cost ALOT of money:

  ”A famous example of the dangers here was Knight Capital´s system losing $465 millons in 45 minutes, apparently because of unexpected behavior from obsolete experimental codepaths” – Hidden Technical depth in Machine Learning Systems

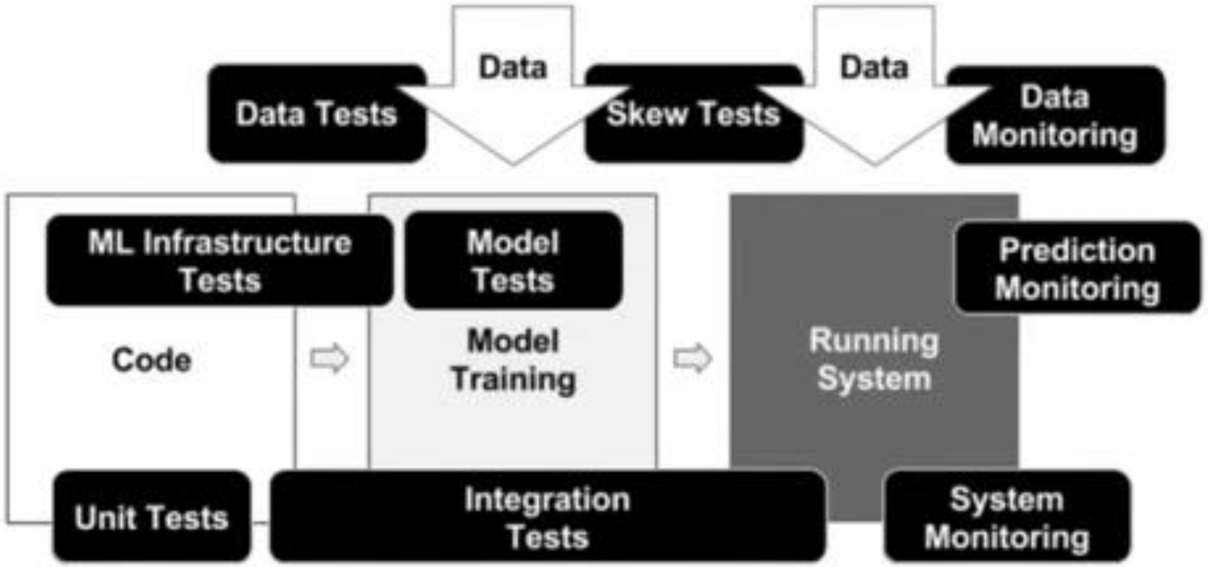


Why is MLOps harder than DevOps

- It involves a freaking lot of testing



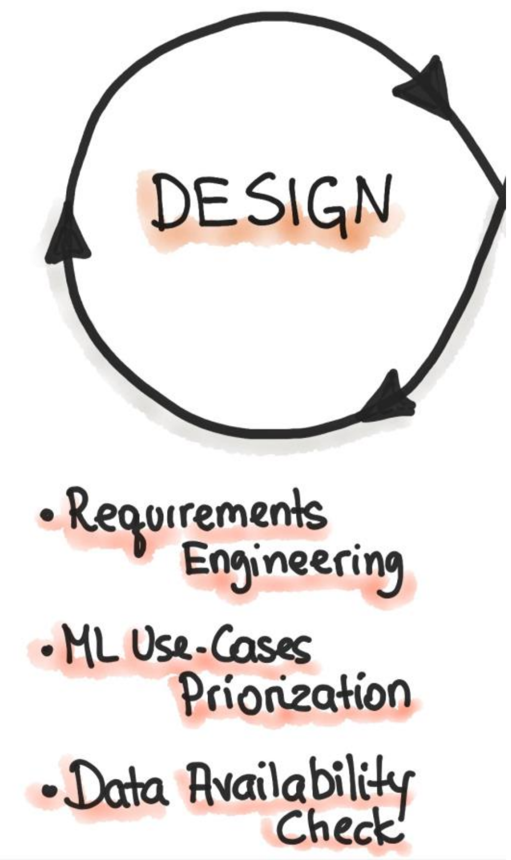
Traditional System Testing and Monitoring



ML-Based System Testing and Monitoring

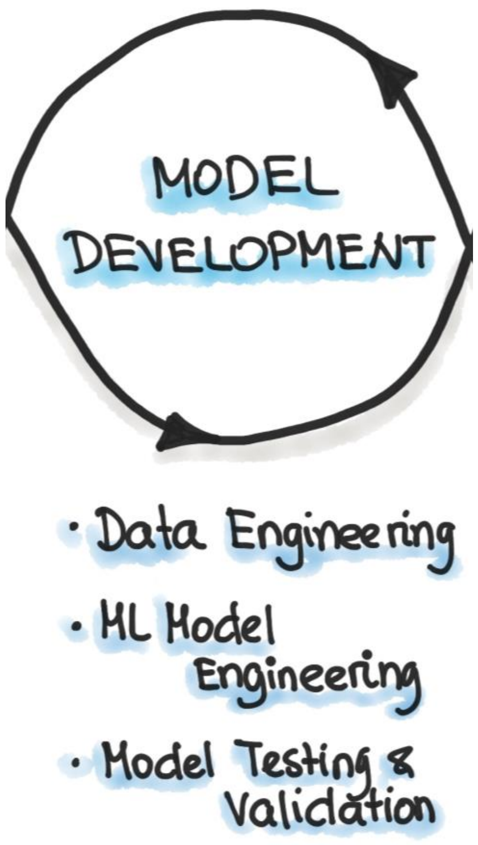
Design

- The is the main part we train you at DTU
  - Analyze a problem
  - Look in litterature for references
  - Check if you have access to data for investigating this



Development

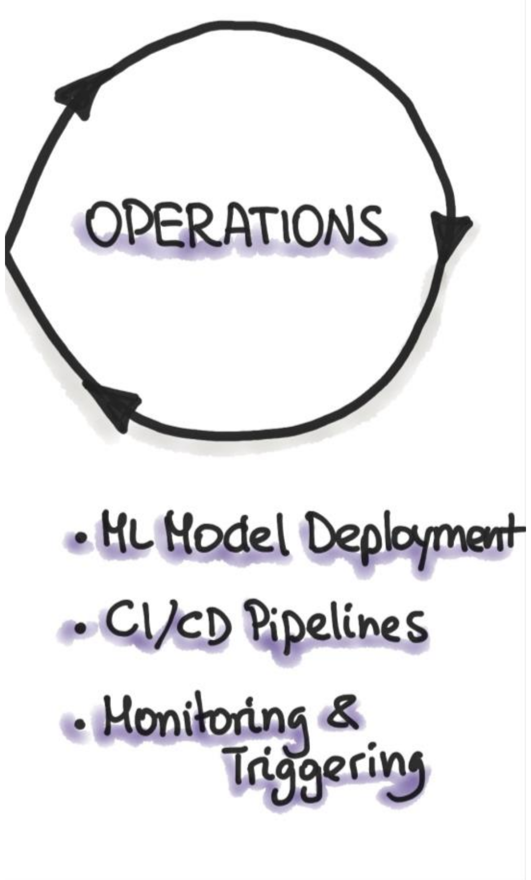
- This is somewhat covered in other courses
  - Going from ideas to practical implementation
  - How should data be formatted to guide the development
  - How should model be validated and tested
- This course will introduce tools to be more organised in this phase



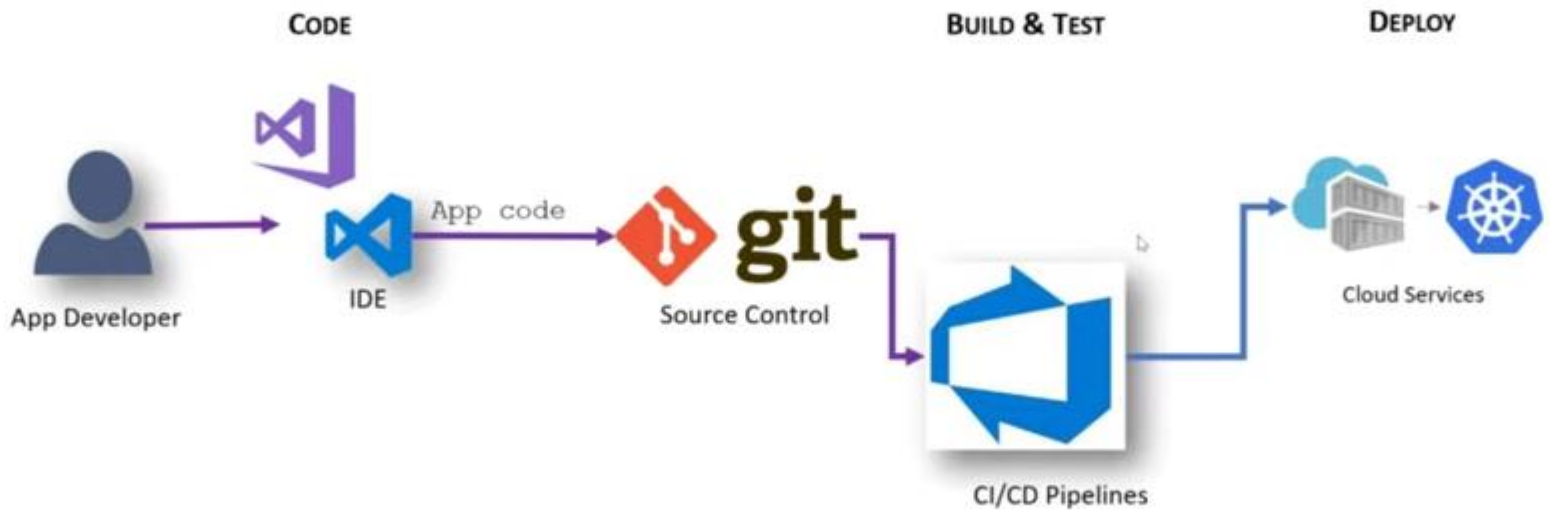
Operations (The new kid)

- To my knowledge, is not teached at DTU

- Operations = How to make sure models do not break
  - My hope is that you will get at feeling of this topic
  - Specifically we will touch apon deployment and CI

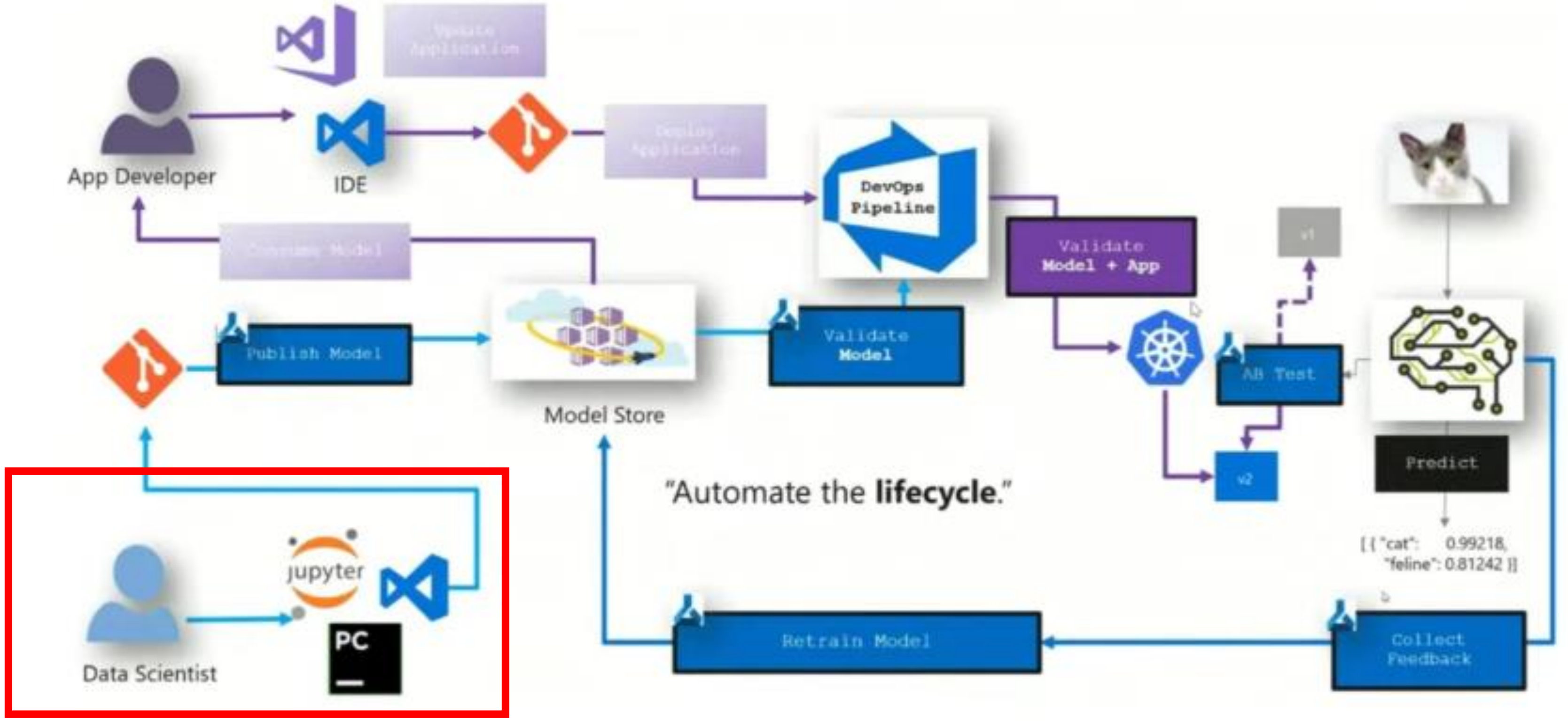


The workflow of standard DevOps



The workflow of MLOps

DevOps on steriods



This is what I expect you can do

The big difference is MLOps requires domain knowledge

MLOps at a high level

1. Optimizing workflows
   - Getting organized cost time initially but will save you time down the line
2. Versioning
   - Keep track of code changes, trained models etc. so everything can be backtracked
3. Automatization and Continuous X
   - Make sure that new changes automatically gets tested, deployed etc.
4. Reusability
   - Why rewrite the same code for a new project if you can reuse
5. Reproducibility
   - Make sure that your results can be redon by others

The first step of MLOps: Organization, code style and version control

- Today exercises is all about organizing your workflow.

- While organization is maybe not that big of a deal on personal projects, it is an essential factor when working on large scale projects

```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── setup.py           <- makes project pip installable (pip install -e .) so src can be imported
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models to make
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented visualizations
│       └── visualize.py
│
└── tox.ini            <- tox file with settings for running tox; see tox.readthedocs.io
```

The first step of MLOps: Organization, code style and version control

- There is no right and wrong way
- It comes down to being consistent
- However, complying to standard

style_script.py

```python
def list_sum(my_list):
    #First method
    sum=0
    for i in my_list:
        sum+=i
    return sum
my_list=[1,2,3,4]
output = list_sum(my_list)
if (output>=10):
    print("You have reached the threshold.")
```

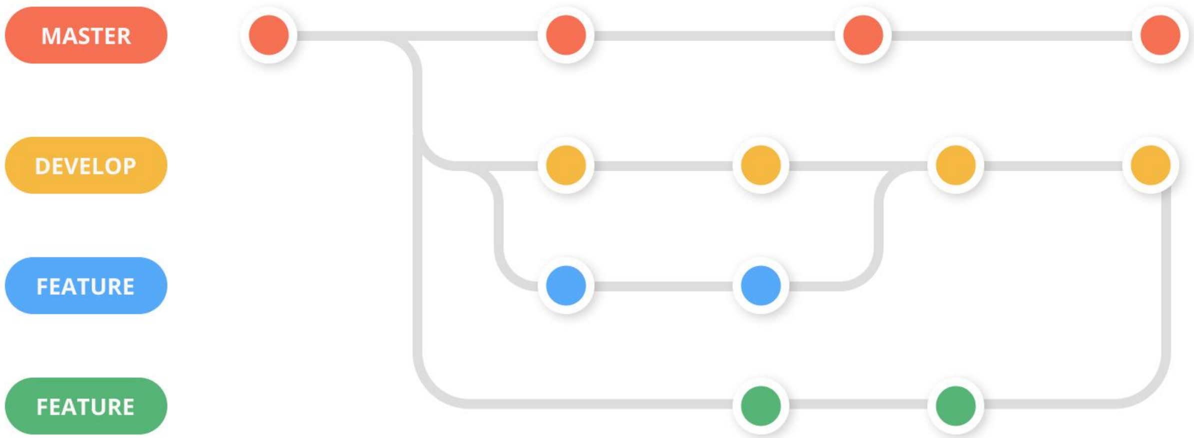
style_script.py

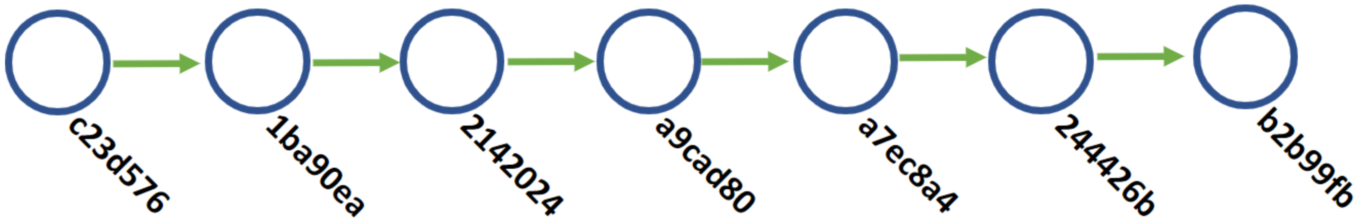
```python
def list_sum(my_list):
    # First method
    sum = 0
    for i in my_list:
        sum += i
    return sum


my_list = [1, 2, 3, 4]
output = list_sum(my_list)
if output >= 10:
    print("You have reached the threshold.")
```

The first step of MLOps: Organization, code style and version control

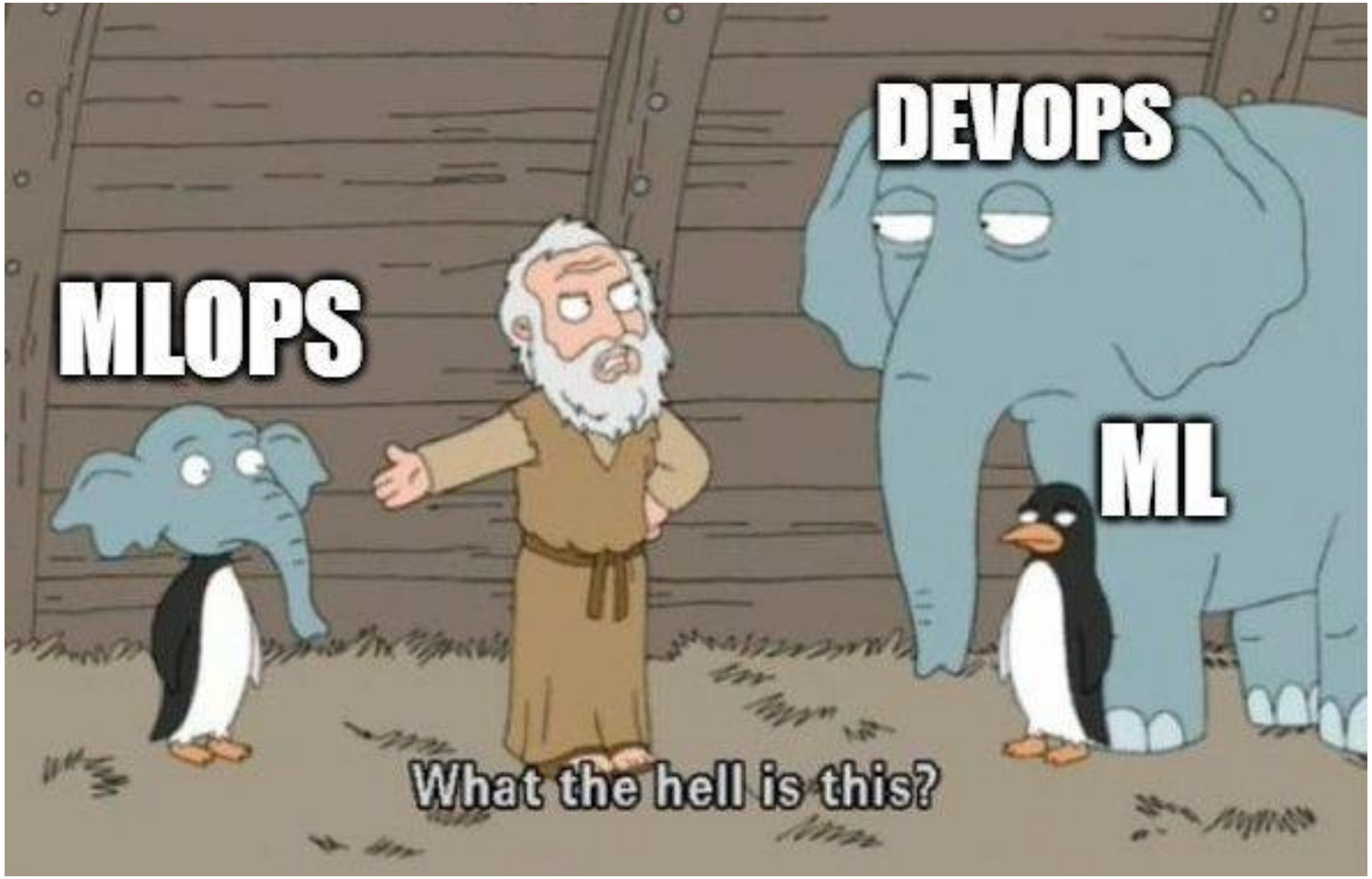
- Version control helps keep track of code changes, enabling to always roll back if something goes wrong

- I able to scale to 1000+ developers working on the same codebase





Meme of the day



https://skaftenicki.github.io/dtu_mlops/s2_organisation_and_version_control/S2.html