# Before we start

ADALTAS

# About me

ADALTAS

- Petra Kaferle Devisschere
- Data Scientist @ Adaltas
  - Working with clients on PoCs and production-destined projects
  - Teaching: Spark, Python, Git, and for the first time MLOps
  - Technical articles: https://www.adaltas.com/en/articles/
- DSTI Data science S19
  - Background in biochemistry with a lot of data analysis and visualization

# About the course

ADALTAS

- NEW => please, give feedback
- 4 days, 2 modules per day
- Material for the labs: https://github.com/adaltas/dsti-mlops-2021-spring
- Focus on understanding the concept, not on coding

**Do interact and ask questions**

# Motivation

ADALTAS

- A lot of MLOps skills implement good practice (versioning, testing…)
  => increase the **quality** and **reproducibility** of work
- **High in-demand skills!**
- To avoid the following situations:

# 1. Introduction to MLOps

## Bringing ML projects to production

# Why is it important?

ADALTAS

- Only models in production bring added value to the business
- Data Scientists are not final users

# But...

- 80% of the models are never deployed
- Developing a model takes weeks/months, deploying takes months/year(s)

# Data science landscape

ADALTAS

Data science is much more than just machine learning!

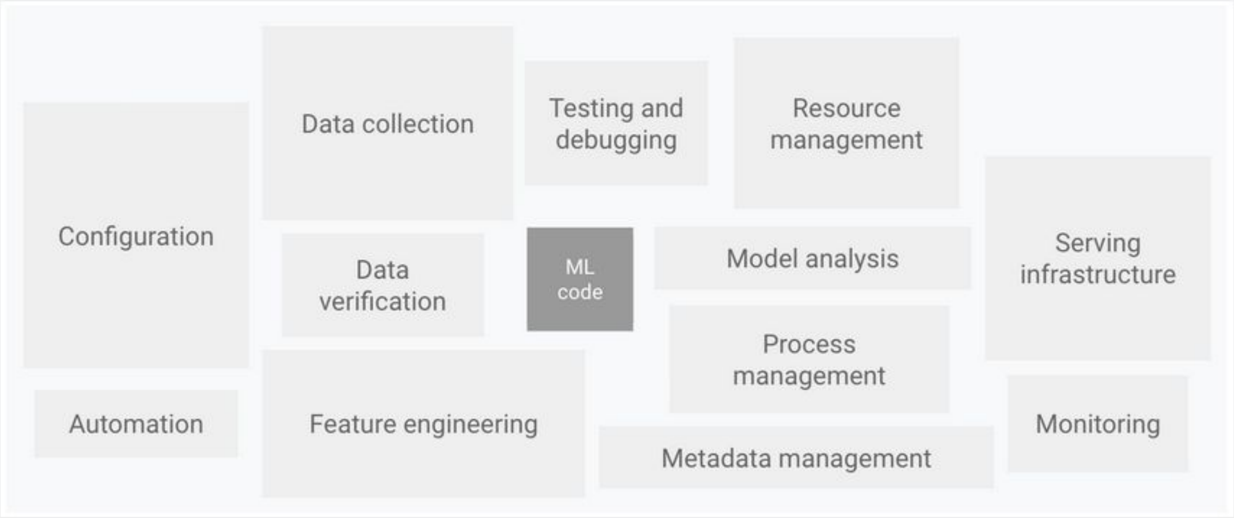



**Figure 1.** Elements for ML systems. Adapted from Hidden Technical Debt in Machine Learning Systems.

https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

# ML in research vs. production

| | Research | Production |
|---|---|---|
| Objectives | Model performance | Different stakeholders have different objectives |
| Computational priority | Fast training, high throughput | Fast inference, low latency |
| Data | Static | Constantly shifting |
| Fairness | Good to have (sadly) | Important |
| Interpretability* | Good to have | Important |

https://www.youtube.com/watch?v=g08qBcdk3Ss

# Challenges of deploying ML

Table 1: All considerations, issues and concerns explored in this study. Each is assigned to the stage and step of the deployment workflow where it is commonly encountered.

| Deployment Stage | Deployment Step | Considerations, Issues and Concerns |
|---|---|---|
| Data management | Data collection | Data discovery |
| | Data preprocessing | Data dispersion<br>Data cleaning |
| | Data augmentation | Labeling of large volumes of data<br>Access to experts<br>Lack of high-variance data |
| | Data analysis | Data profiling |
| Model learning | Model selection | Model complexity<br>Resource-constrained environments<br>Interpretability of the model |
| | Training | Computational cost<br>Environmental impact |
| | Hyper-parameter selection | Resource-heavy techniques<br>Hardware-aware optimization |
| Model verification | Requirement encoding | Performance metrics<br>Business driven metrics |
| | Formal verification | Regulatory frameworks |
| | Test-based verification | Simulation-based testing |
| Model deployment | Integration | Operational support<br>Reuse of code and models<br>Software engineering anti-patterns<br>Mixed team dynamics |
| | Monitoring | Feedback loops<br>Outlier detection<br>Custom design tooling |
| | Updating | Concept drift<br>Continuous delivery |
| Cross-cutting aspects | Ethics | Country-level regulations<br>Focus on technical solution only<br>Aggravation of biases<br>Authorship<br>Decision making |
| | End users' trust | Involvement of end users<br>User experience<br>Explainability score |
| | Security | Data poisoning<br>Model stealing<br>Model inversion |

ADALTAS

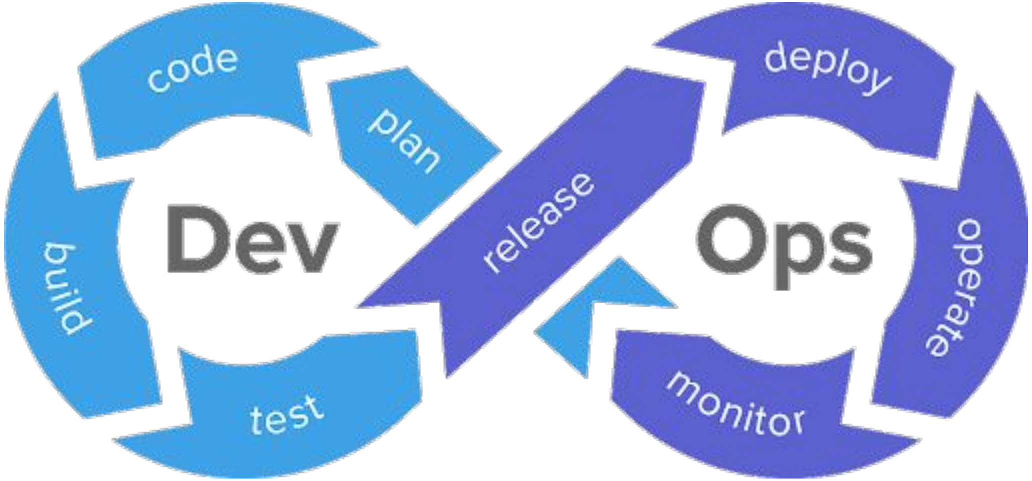
https://arxiv.org/pdf/2011.09926.pdf

# Software is being deployed all the time, so what's the big deal?

Let’s start with analogy in software development => **DevOps**

# But what is DevOps?

ADALTAS




- **<u>It's a culture</u>**
- Automation
- Quick delivery of results

# DevOps Principles (Software development)

ADALTAS

**Version Control.** Developers submit code changes to a central repository several times a day. Prior to submitting code to the master repository (master branch), all code must be verified. To facilitate collaboration, other developers can track changes.

**Continuous Integration.** Members of the development team integrate their code in a shared repository, several times a day. Each developer segments the work into small, manageable chunks of code and detects potential merge conflicts and bugs quicker.

**Continuous Delivery.** As the code is continuously integrated, it is also consistently delivered to the end-user. Smaller contributions allow faster update releases, which is a crucial factor for customer satisfaction.

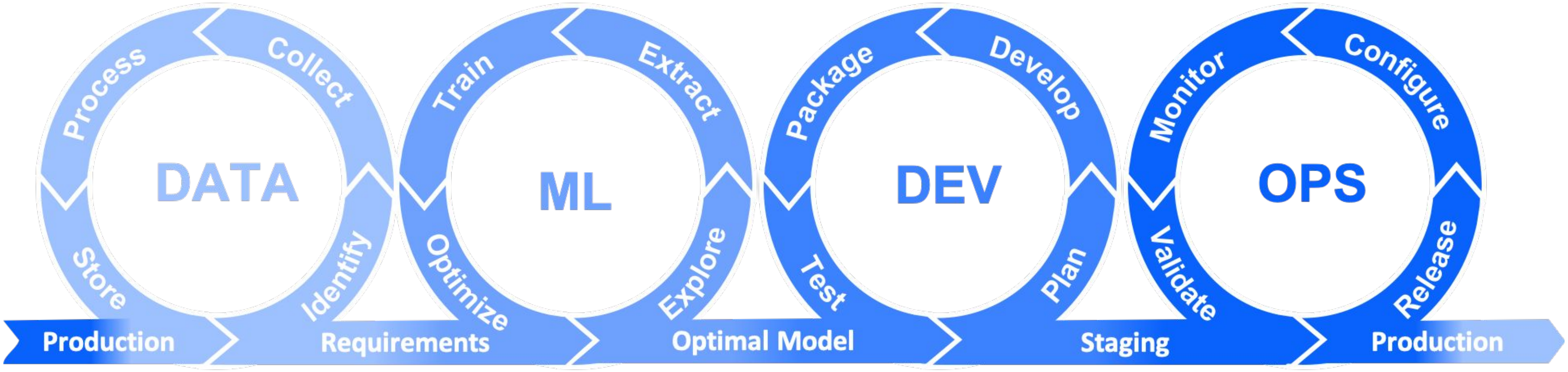
**Continuous Deployment.** A big part of DevOps is automating processes to speed up production. Continuous deployment involves automating releases of minor updates that do not pose a substantial threat to the existing architecture.

**Continuous Testing.** Such a strategy involves testing as much as possible in every step of development. Automated tests give valuable feedback and a risk assessment of the process at hand.

**Continuous Operations.** The DevOps team is always working on upgrading software with small but frequent releases. That is why DevOps requires constant monitoring of performance. Its main goal is to prevent downtime and availability issues during code release.

**Collaboration.** One of the main goals of DevOps is to foster collaboration and feedback sharing. Development and Operations need to proactively communicate and share feedback to maintain an efficient DevOps pipeline.

# MLOps = DevOps for ML



# MLOps Principles :

## replace `code` with `code + model + data` => MLOps

**Version Control.** Developers submit code changes to a central repository several times a day. Prior to submitting code to the master repository (master branch), all code must be verified. To facilitate collaboration, other developers can track changes.

**Continuous Integration.** Members of the development team integrate their code in a shared repository, several times a day. Each developer segments the work into small, manageable chunks of code and detects potential merge conflicts and bugs quicker.

**Continuous Delivery.** As the code is continuously integrated, it is also consistently delivered to the end-user. Smaller contributions allow faster update releases, which is a crucial factor for customer satisfaction.

**Continuous Deployment.** A big part of DevOps is automating processes to speed up production. Continuous deployment involves automating releases of minor updates that do not pose a substantial threat to the existing architecture.

**Continuous Testing.** Such a strategy involves testing as much as possible in every step of development. Automated tests give valuable feedback and a risk assessment of the process at hand.

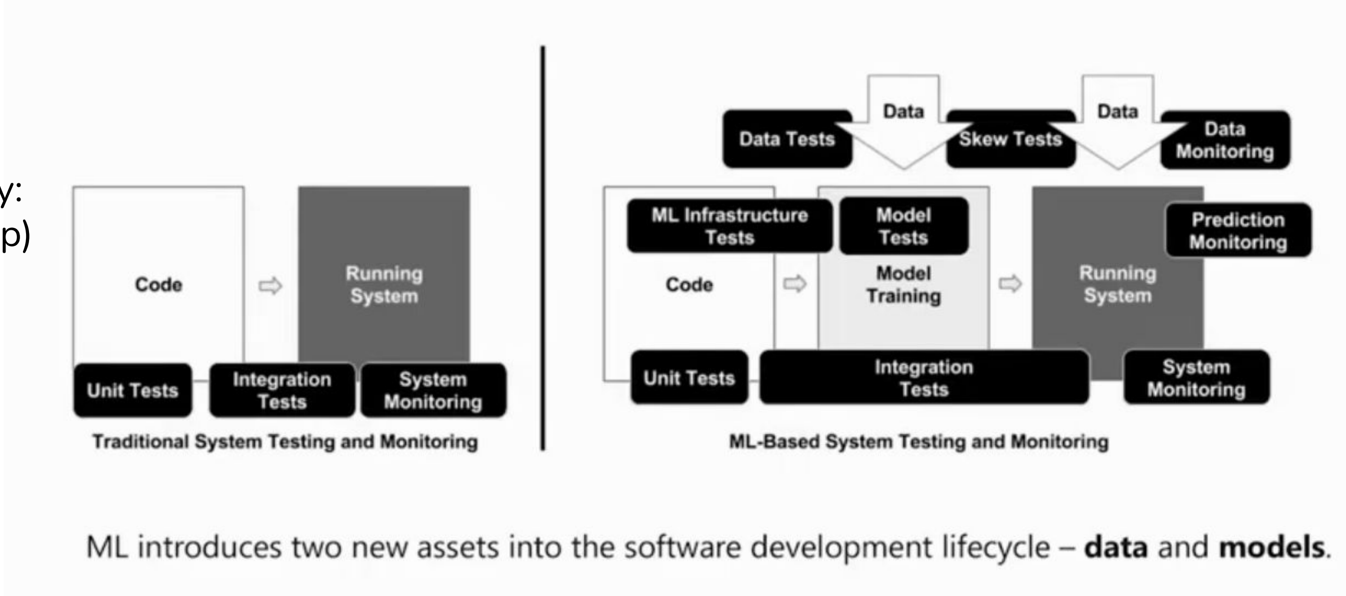
**Continuous Operations.** The DevOps team is always working on upgrading software with small but frequent releases. That is why DevOps requires constant monitoring of performance. Its main goal is to prevent downtime and availability issues during code release.

**Collaboration.** One of the main goals of DevOps is to foster collaboration and feedback sharing. Development and Operations need to proactively communicate and share feedback to maintain an efficient DevOps pipeline.

**Model feedback.** Saving incoming data and predictions once a model is deployed

# DevOps vs MLOps

We deploy:
- code (app)

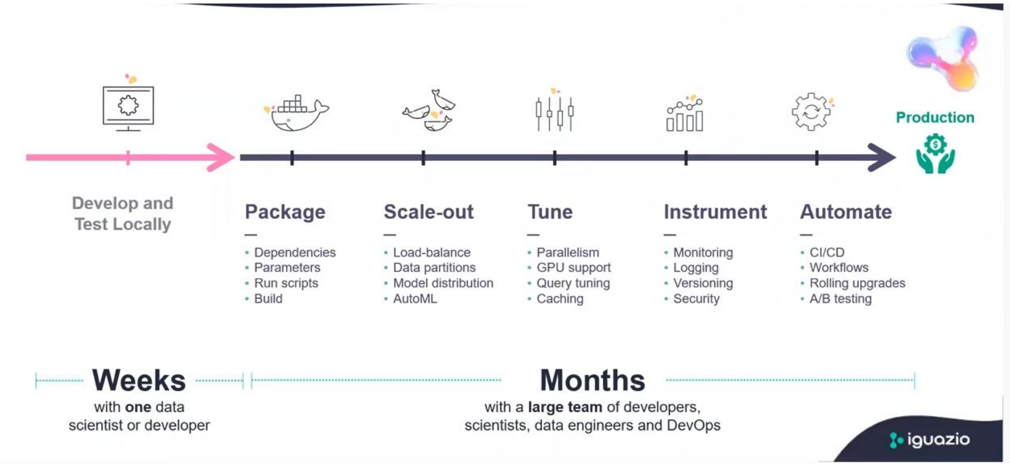


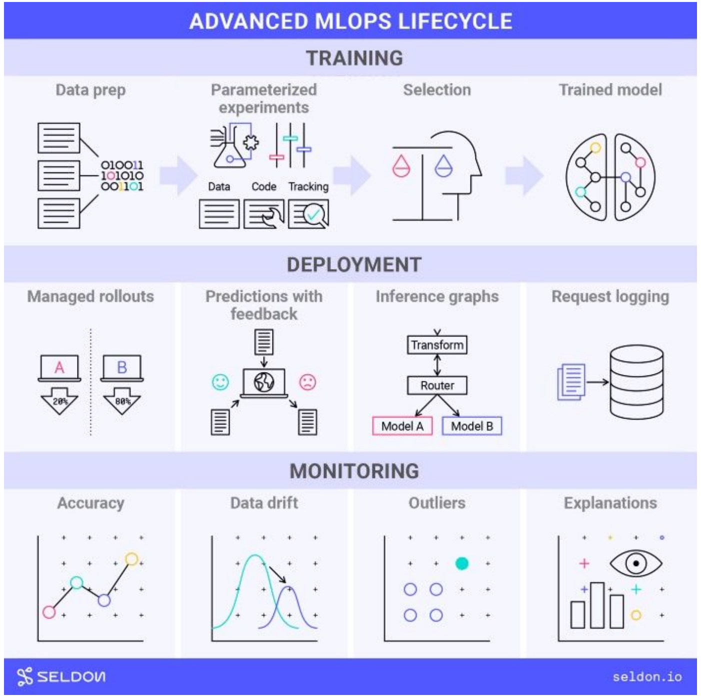

We deploy:
- code:
  + data pre-processing
  + feature engineering
- model
- sometimes data
  + aggregates

**=> pipeline**

https://www.youtube.com/watch?v=pqppGvTJm-A

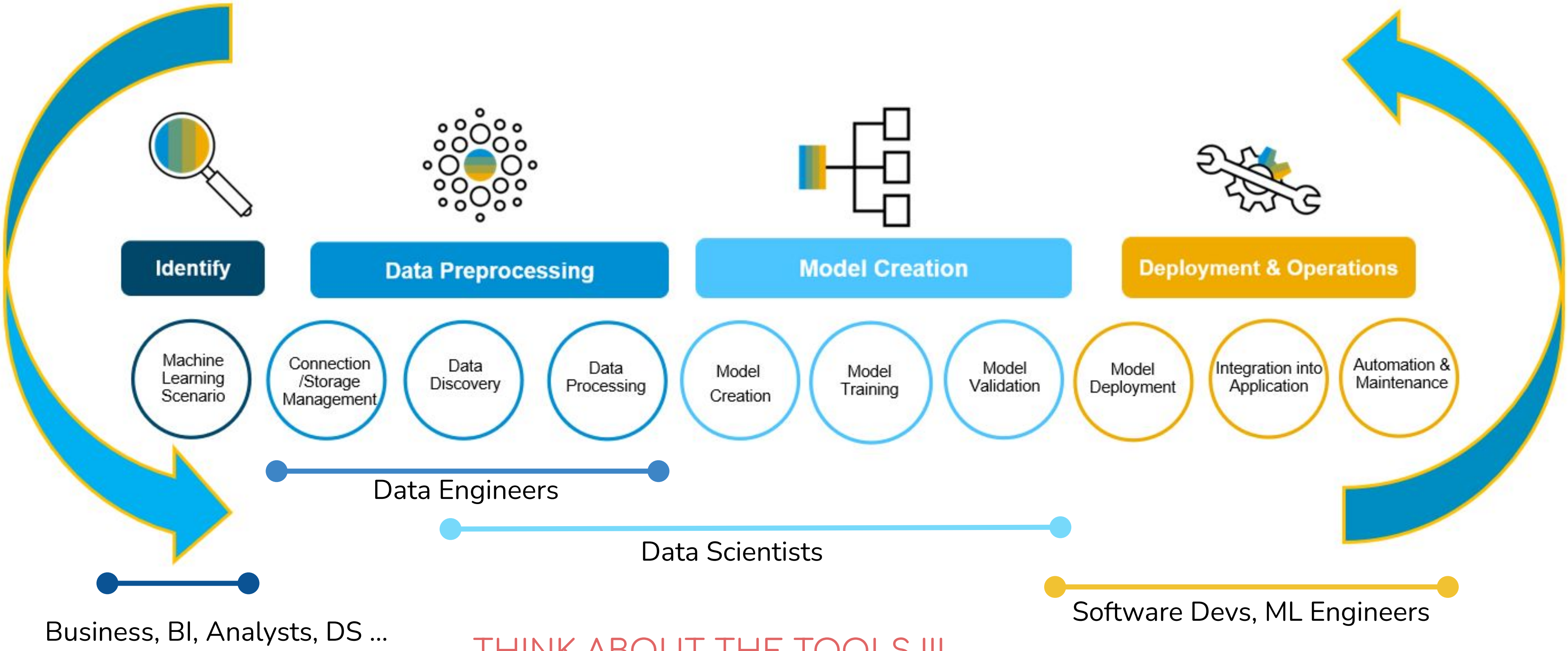
# Different tasks (and definitions) of MLOps



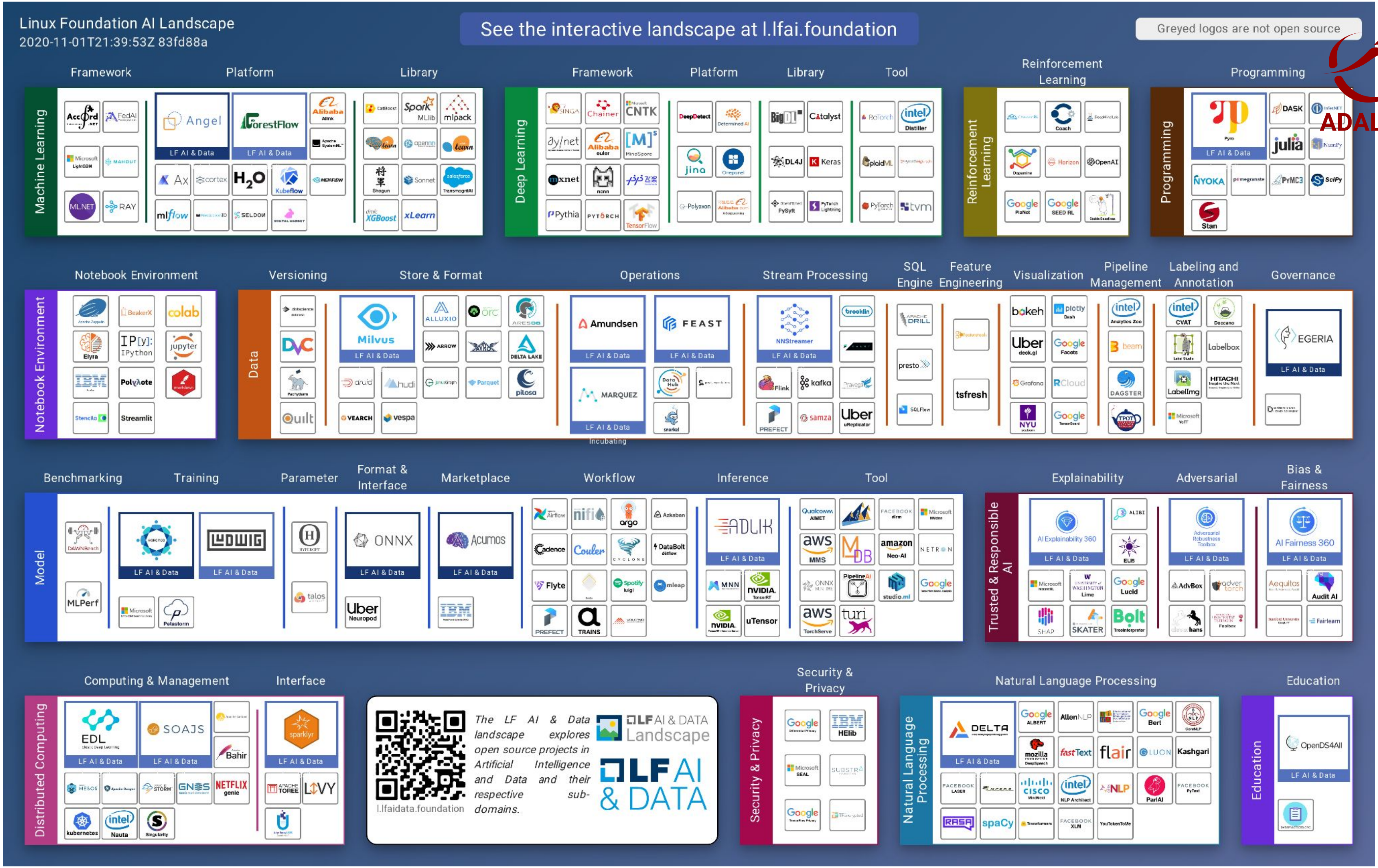




https://www.youtube.com/watch?v=VCUDo9umKEQ

# ML Lifecycle and Roles




THINK ABOUT THE TOOLS !!!



Linux Foundation AI Landscape
2020-11-01T21:39:53Z 83fd88a
See the interactive landscape at l.lfai.foundation
Greyed logos are not open source
Machine Learning
Framework
Platform
Library
Deep Learning
Framework
Platform
Library
Tool
Reinforcement Learning
Programming
Notebook Environment
Data
Versioning
Store & Format
Operations
Stream Processing
SQL Engine
Feature Engineering
Visualization
Pipeline Management
Labeling and Annotation
Governance
Incubating
Model
Benchmarking
Training
Parameter
Format & Interface
Marketplace
Workflow
Inference
Tool
Trusted & Responsible AI
Explainability
Adversarial
Bias & Fairness
Distributed Computing
Computing & Management
Interface
The LF AI & Data landscape explores open source projects in Artificial Intelligence and Data and their respective sub-domains.
l.lfaidata.foundation
LF AI & DATA Landscape
LF AI & DATA
Security & Privacy
Natural Language Processing
Education
LF AI & Data

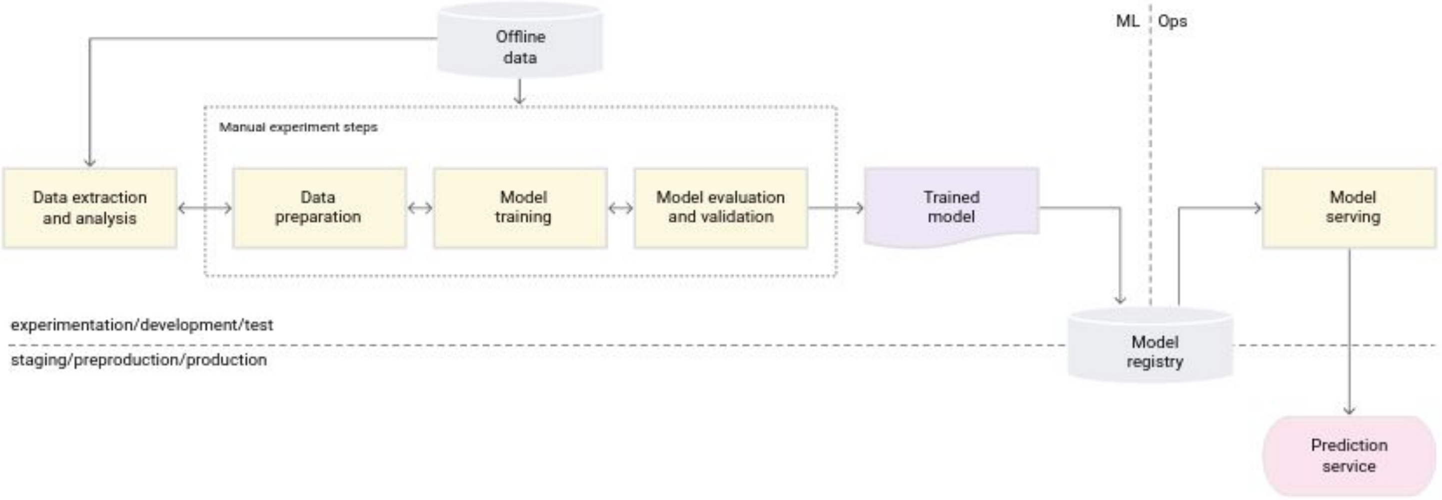
# Maturity levels of MLOps systems (according to Google)

ADALTAS

- Level 0 - manual
- Level 1- ML pipeline automation
- Level 2 - CI/CD pipeline automation

https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

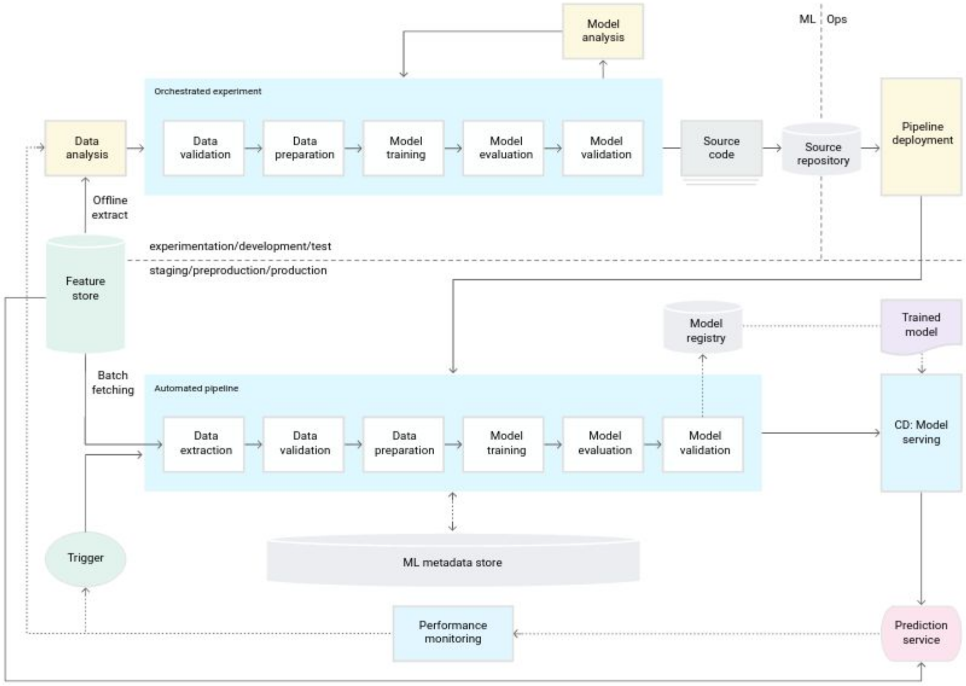
# MLOps level 0: manual process




- Data scientists and Ops teams are separated
- Only the model is served (not the whole pipeline)
- Model is not frequently updated
- No monitoring

# MLOps level 1: ML pipeline automation

ADALTAS

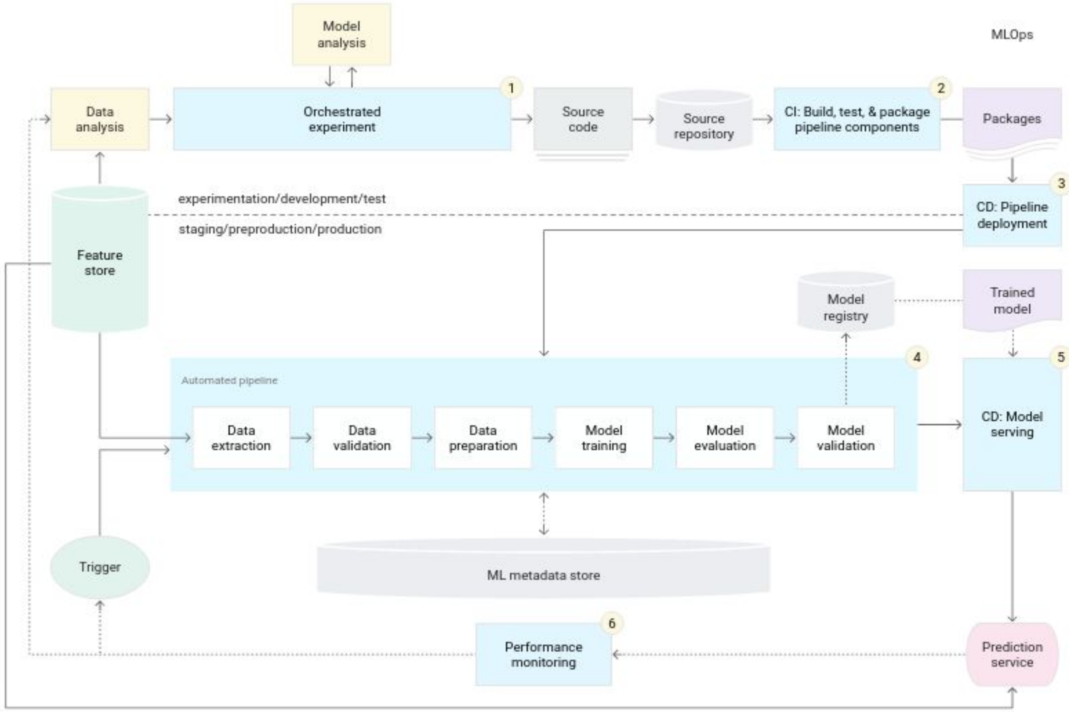



- The goal is CT in production when **new data** is available
- We need to deploy the whole pipeline (data+model)
- The steps need to be automated and orchestrated
- Code is reproducible and modular

- Need of additional components:
  - Triggers - to trigger the re-training
  - Feature store - stores pre-computed features that model uses during serving
  - Metadata store - execution logs

# MLOps level 2: CI/CD pipeline automation

ADALTAS




- Automatically build, test, and deploy the new pipeline components to the target environment
- Deployment:
  - Automated - to the test env.
  - Semi-automated - to pre-production (triggered by merge)
  - Manual - to production

# How does it work in real life?

- Depends a lot on the skills and experience of the group
- Depends a lot on the number of models to deploy/maintain
- Depends on how often we need to retrain the model

- Generally:
    - Starting with the simplest system
    - Gradual improvements: automation of critical steps, adding tests, adding model monitoring...