ML Deployment

Transfiriendo nuestros proyectos

- Hasta ahora nos hemos centrado en la producción de modelos de manera casi experimental.
- El ML tiene un componente muy importante de ingeniería del software/sistemas, que tenemos que tener en cuenta:

  - ¿Como es el proceso de desarrollo?
  - ¿Cuál es el tipo de proceso en el que estamos?
  - Como es nuestro stack, y los entornos en los que vamos a desplegar nuestrs modelos.

Cultura General:
Agile y DevOps

Una corta introducción a agile

The Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools

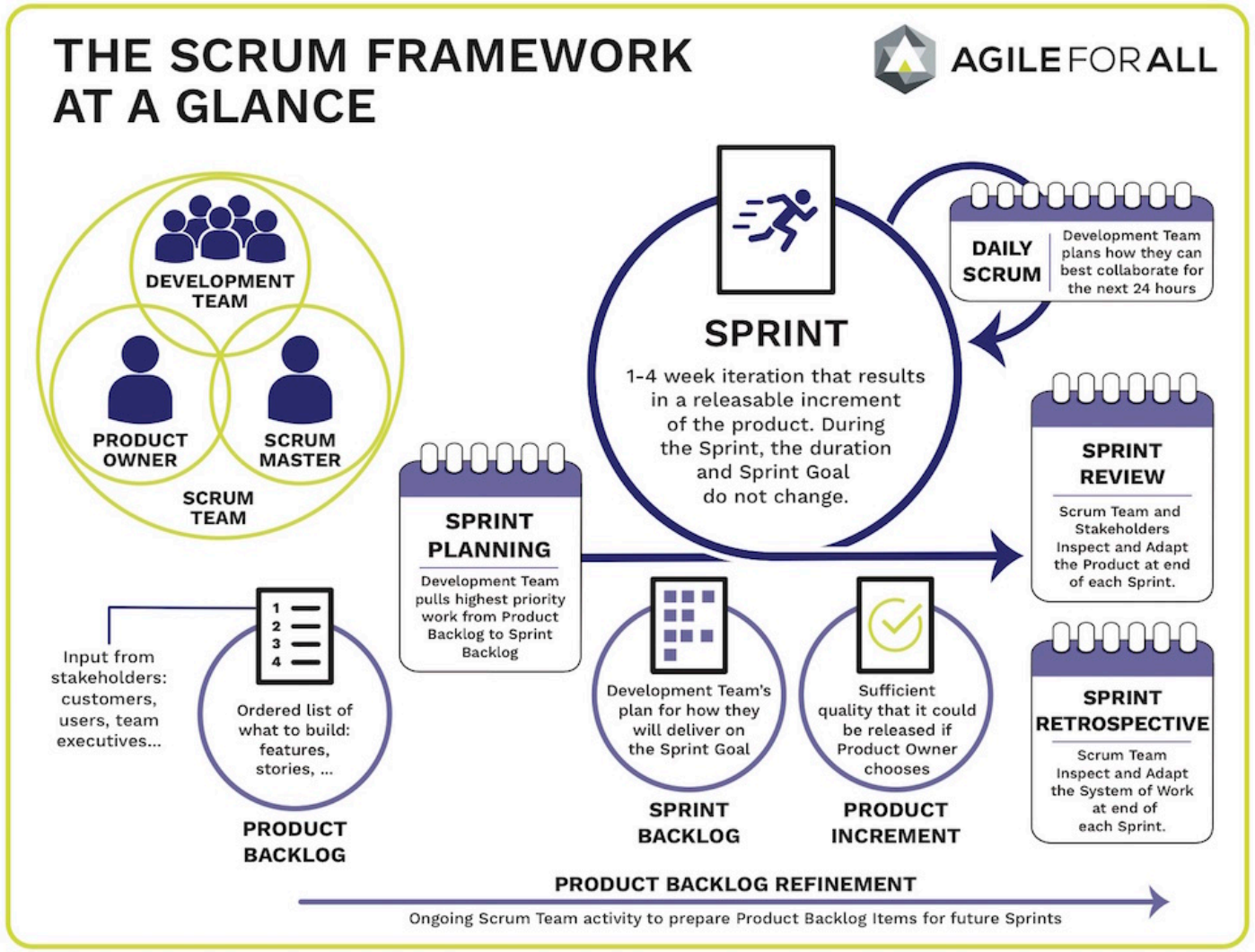
Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

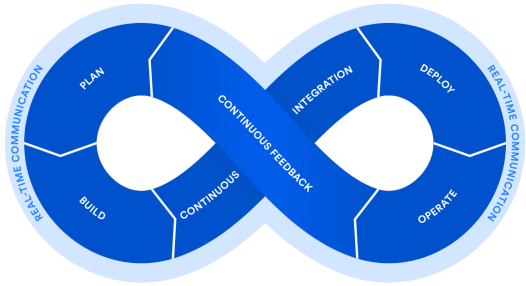
That is, while there is value in the items on the right, we value the items on the left more.

Una corta introducción a agile - Scrum



DevOps & Agile

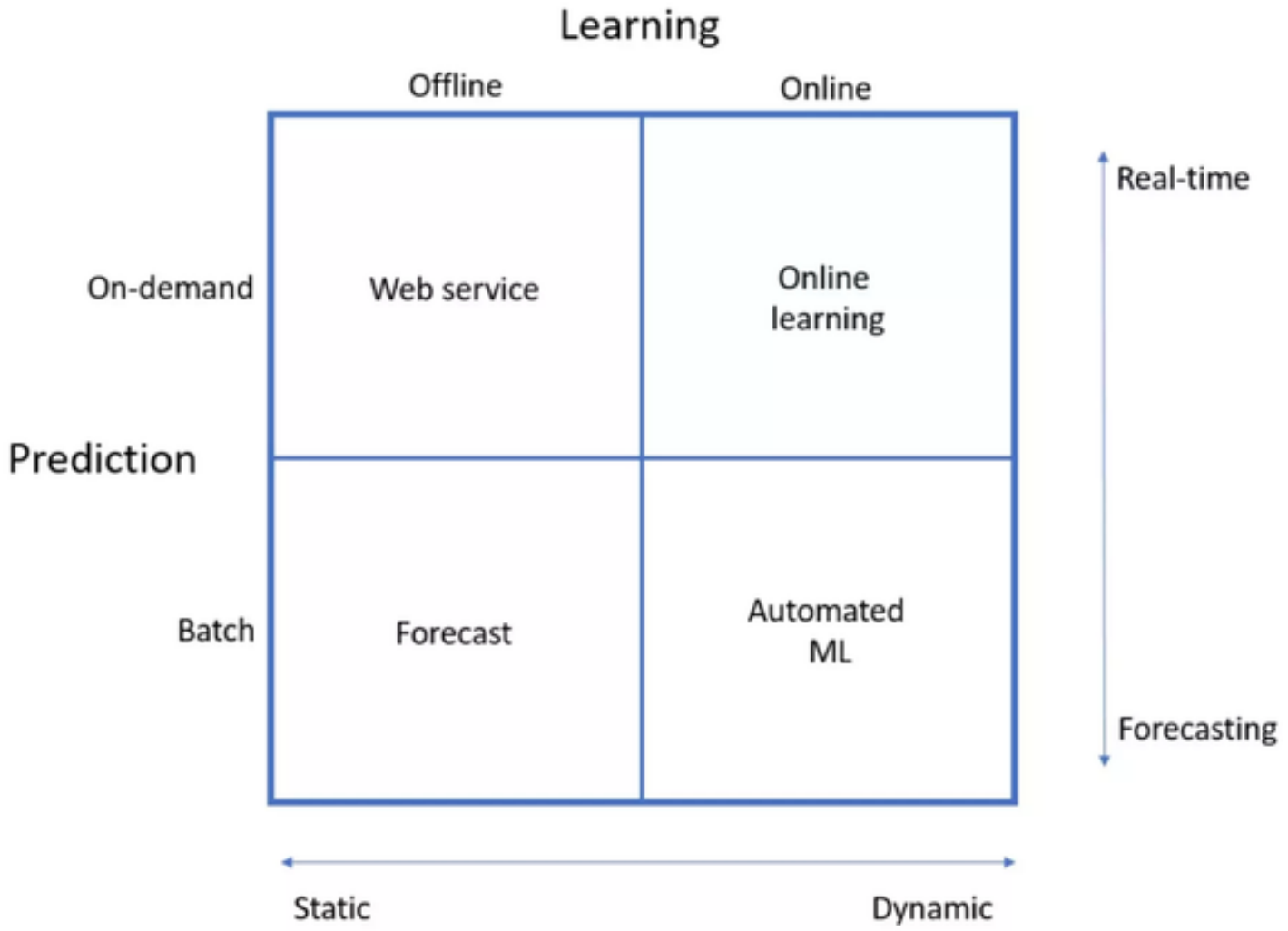
DevOps


PLAN
CONTINUOUS
INTEGRATION
DEPLOY
BUILD
CONTINUOUS FEEDBACK
OPERATE
REAL-TIME COMMUNICATION
REAL-TIME COMMUNICATION


04 TEST
08 MONITOR
03 BUILD
07 OPERATE
DEV
OPS
02 CODE
06 DEPLOY
01 PLAN
05 RELEASE

Modelos de deployment

Que tipo de problema queremos abordar



Workflows más comunes

Batch prediction

Web Service

Real Time Analytics

Modelos de deployment

On premises:

- Que tipo de infraestructura tenemos. Trabajamos con paquetes, creamos contenedores, máquinas virtuales?
- Spark¿?

Cloud:

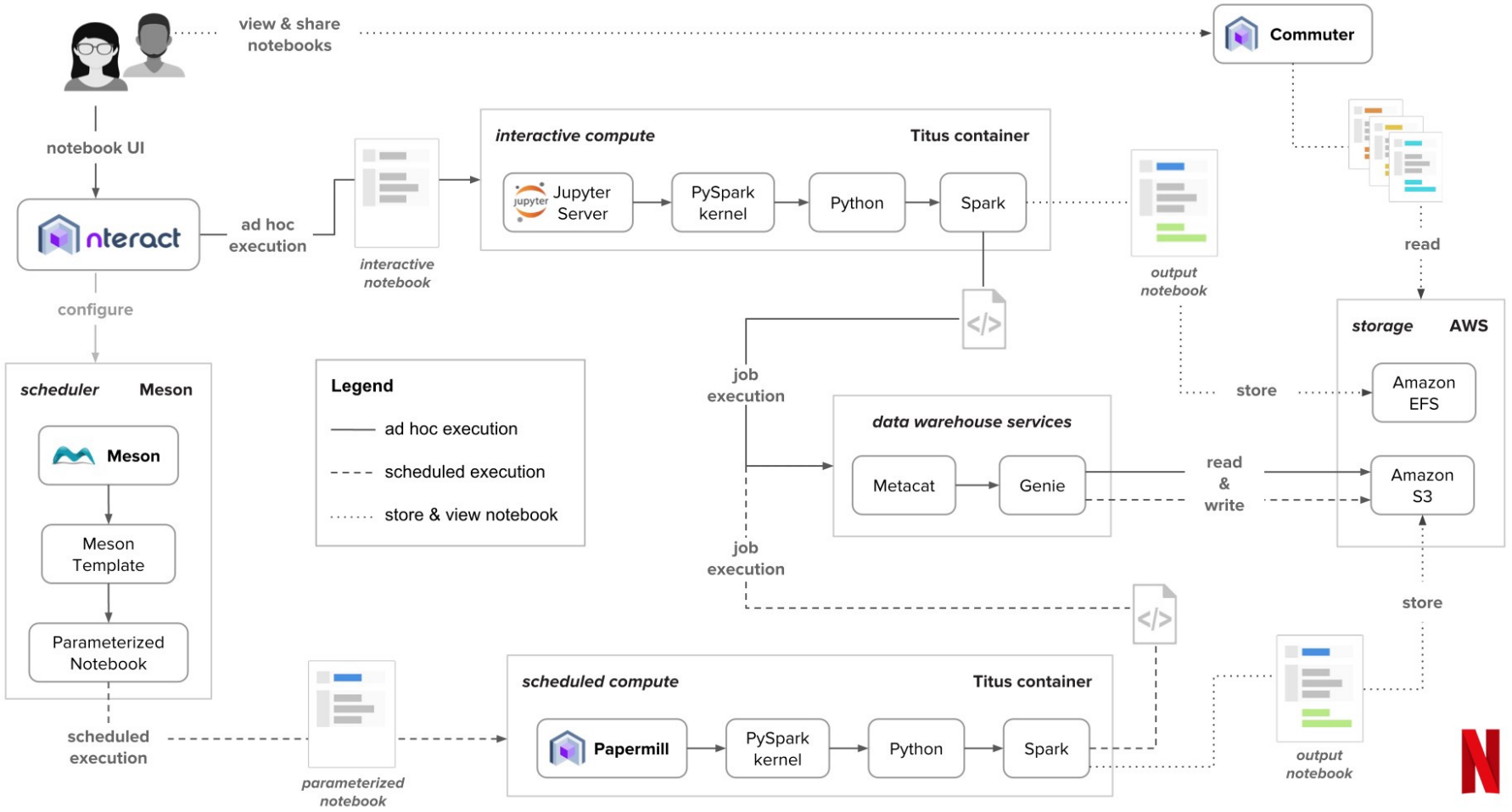
- Que proveedor?
- Qué modelos de deployment.

¿Que principios debemos tener en cuenta?

- Nuestro proyecto debe ser reproducible
- Debe de ser automatizado al máximo
- Extensible y modular
- Escalable
- Debemos tener en cuenta los test
- Debemos tener en cuenta su supervisión y mantenimiento:
  - Loogging
  - Buena documentación
- Debemos tener versionado de modelos y datos

¿Cómo lo hacemos?

Salvo que seas Netflix...

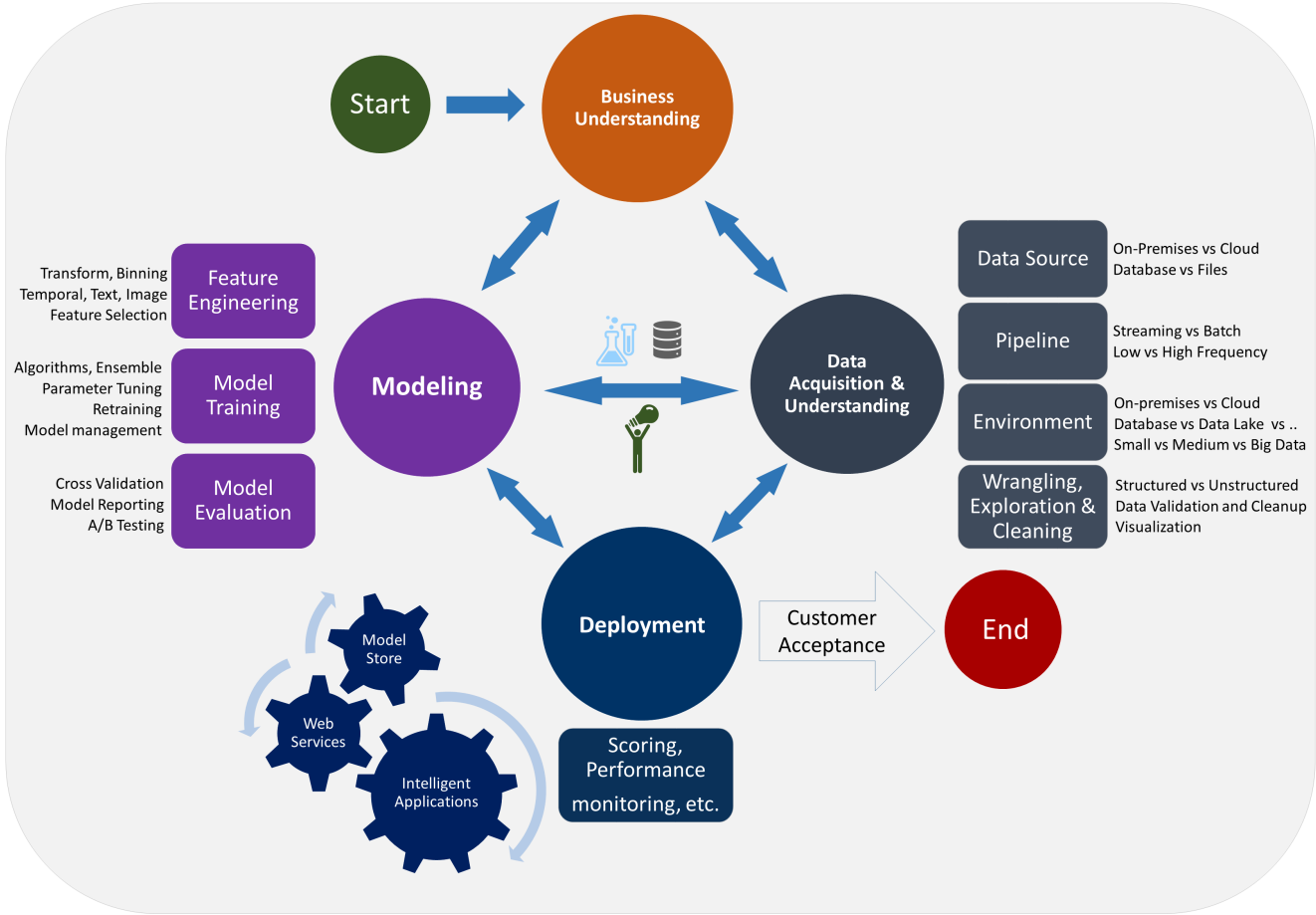


…migraremos nuestros notebooks y crearemos paquetes instalables

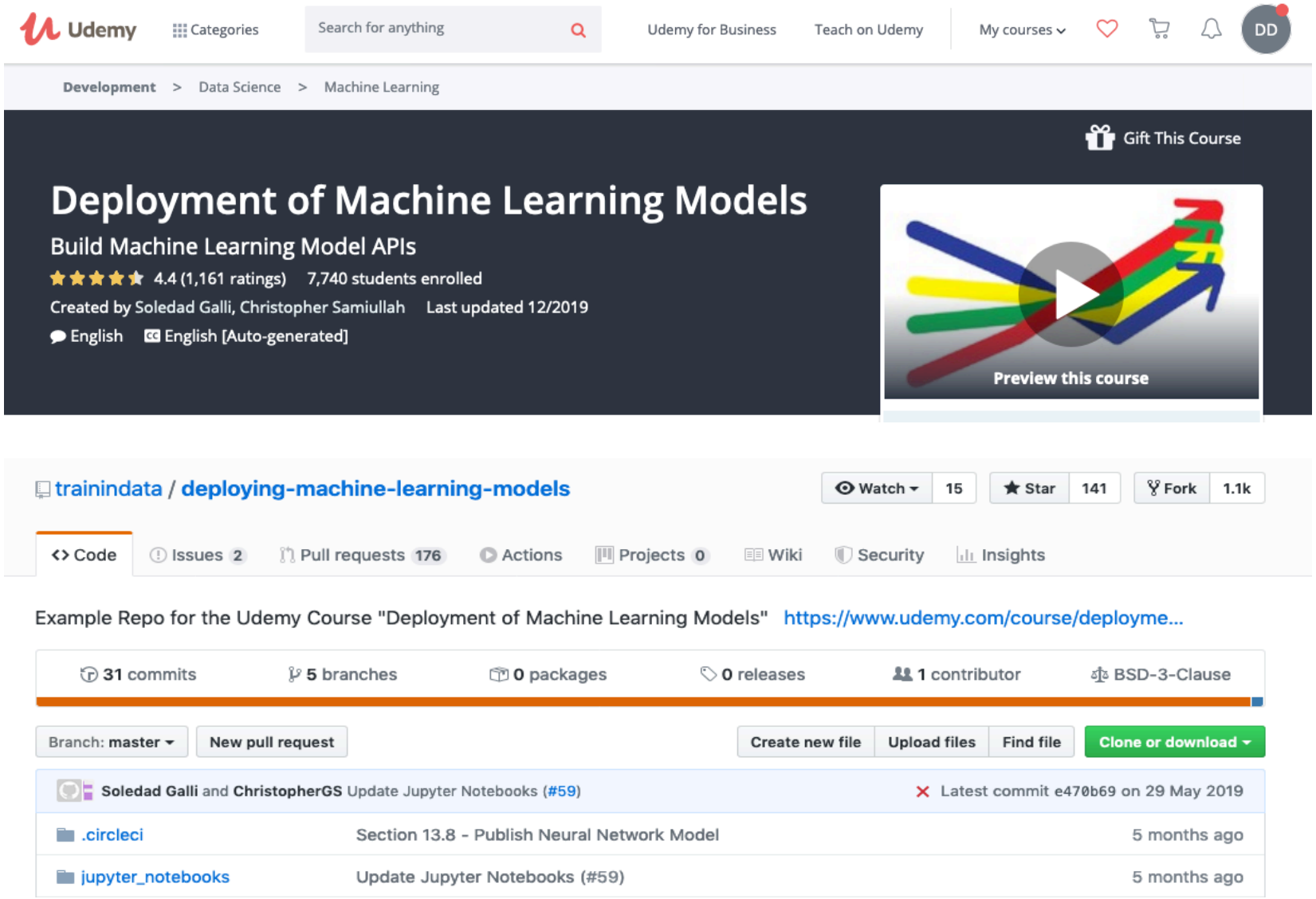
Directory structure

```
├── 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           <- Make this project pip installable with `pip install -e`
├── 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.testrun.org
```

Data Science Lifecycle

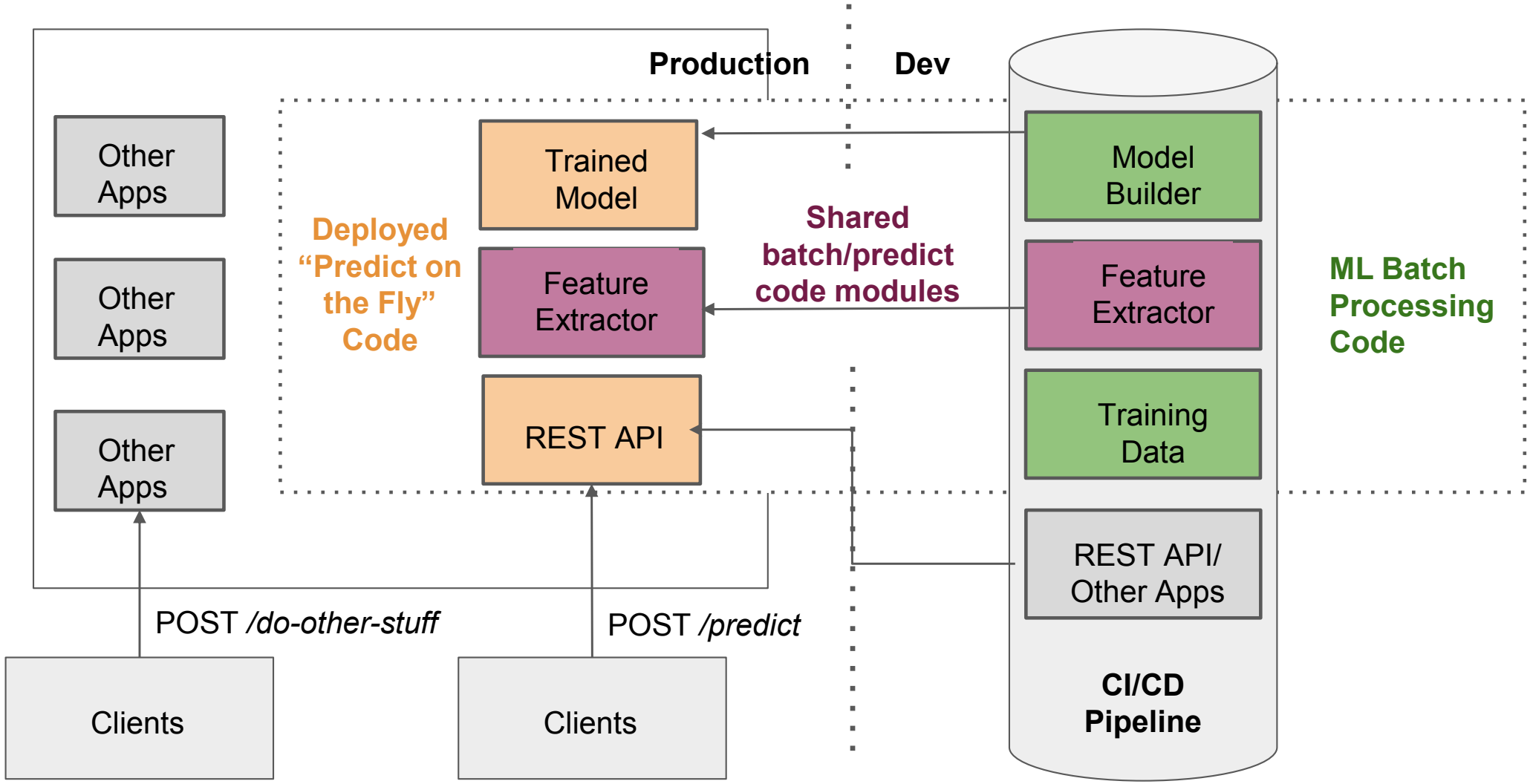


https://github.com/Azure/Azure-TDSP-ProjectTemplate

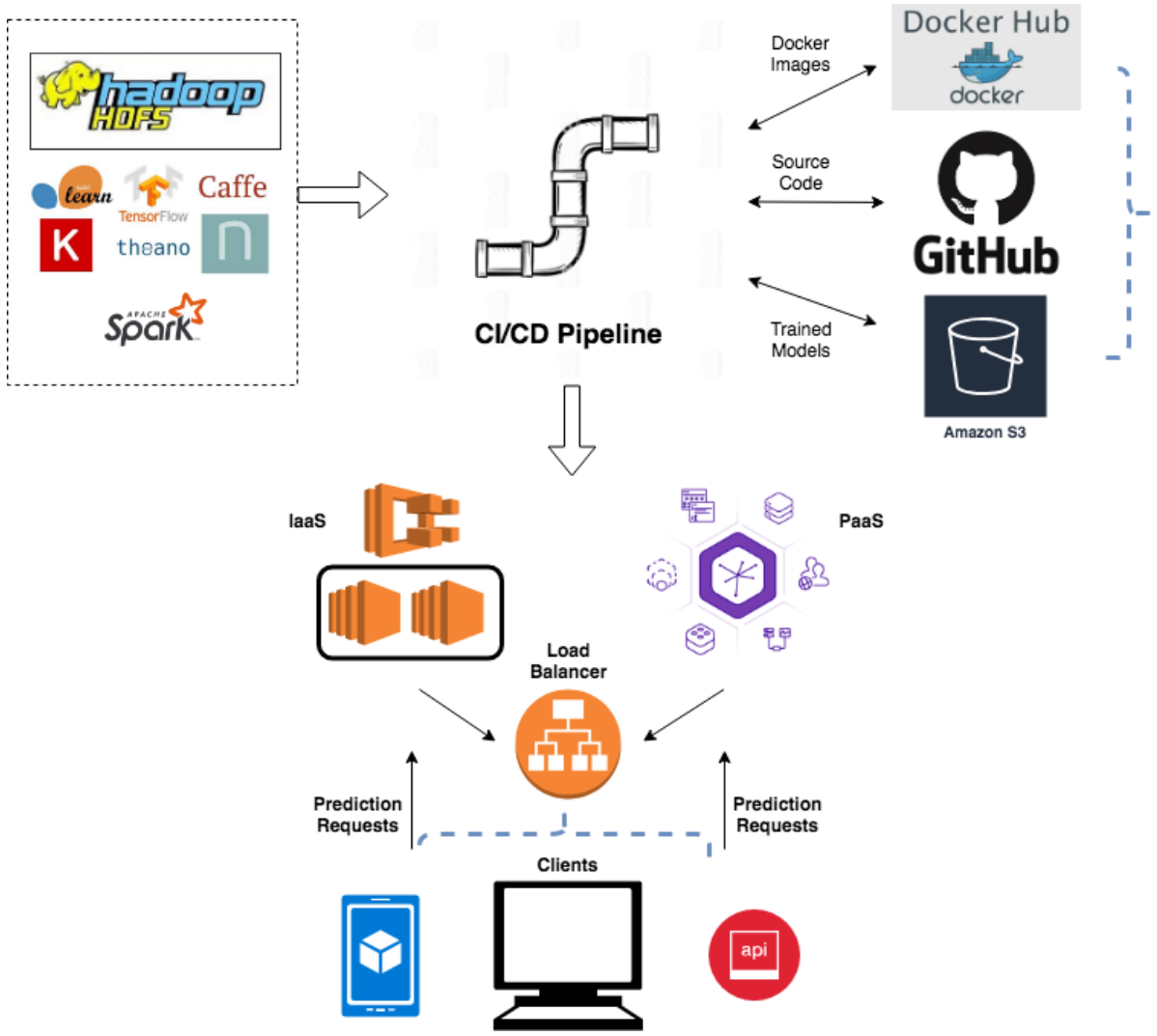


https://github.com/trainindata/deploying-machine-learning-models

Diagram: Train by batch, predict on the fly, serve via REST API



System Diagram



Esta semana

- Pipelines sk-learn, estructura de proyecto, testing, logging
- Serving ML models: API-REST, TF
- Deploying: docker, AWS
- Spark/VirtualBox
- Interpretabilidad

Recursos

Flask

https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world
https://flask.palletsprojects.com/en/1.1.x/tutorial/factory/

Flask + Docker :
http://www.easy-analysis.com/dockerizing-python-flask-app-and-conda-environment

PySpark
https://thegurus.tech/posts/2019/06/how-to-spark-cluster/

Continuos Delivery
https://martinfowler.com/articles/cd4ml.html