Project

MLOps experimental

Giorgia Bertacchini

2022

Intro

Project

The project is a practical implementation of **MLOps practices**.

For the entire life cycle of ML there are two projects that represent the phases: develop and production.

Develop phase

- Data Ingestion
- Data Preparation
- Model Building & Training
- Experimentation
- Model Deploy
- Model Serving

Production phase

- Prediction Service
- Monitoring
- Alerting
- Application & Trigger

Tools & libraries

Develop phase

Workflow orchestration

Kedro

Data analysis and manipulation

pandas

Model training

scikit learn

Experimentation management

mlflow

Model packaging and serving

BENTOML

Data versioning

DVC

Code versioning

GitHub

Deploying pipeline kedro

kedro-docker

Tools & libraries

Production phase

Drift and model monitoring

EVIDENTLY AI

Systems monitoring and alerting

Prometheus

Alert management

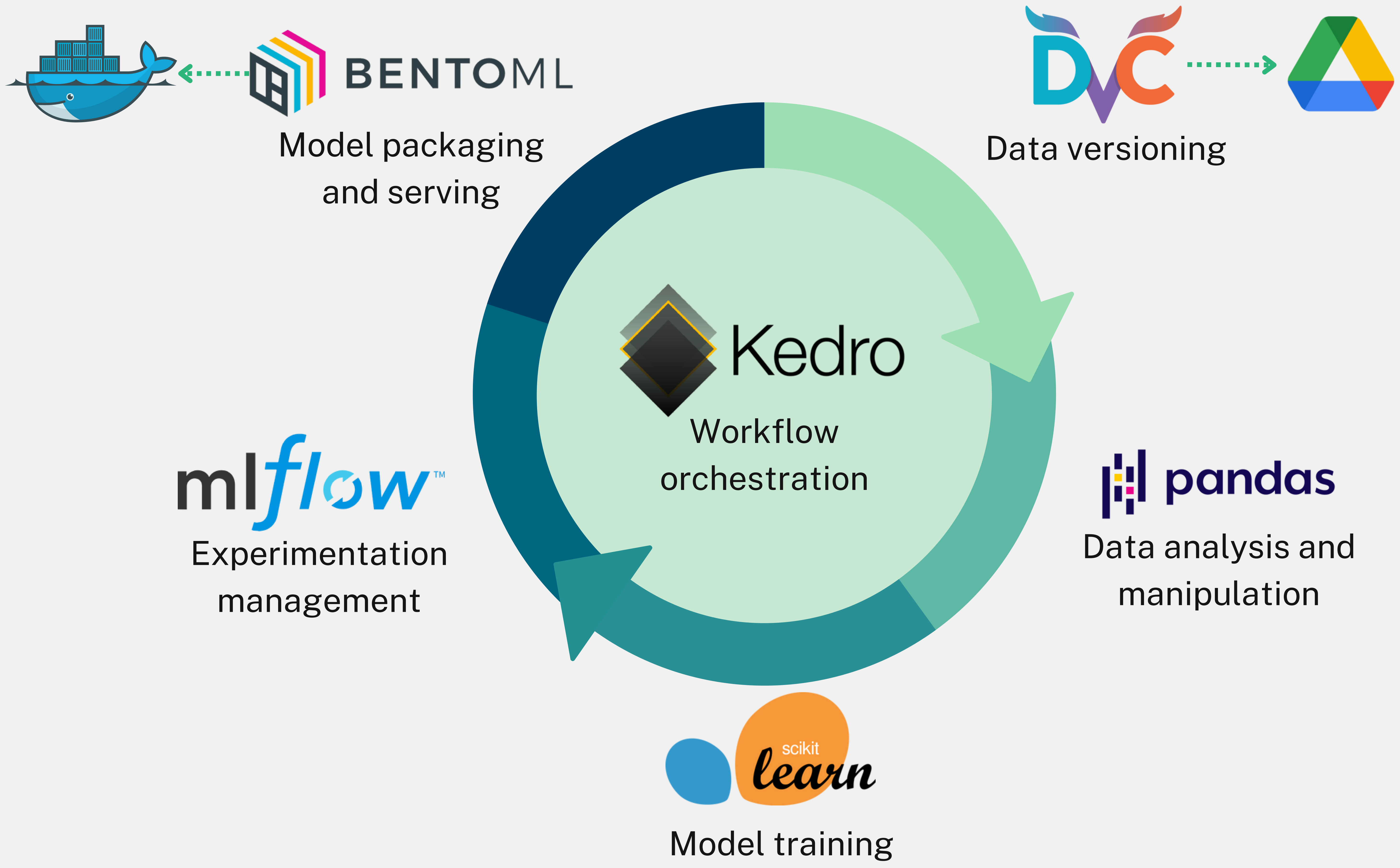
Alertmanager

Managed observability platform

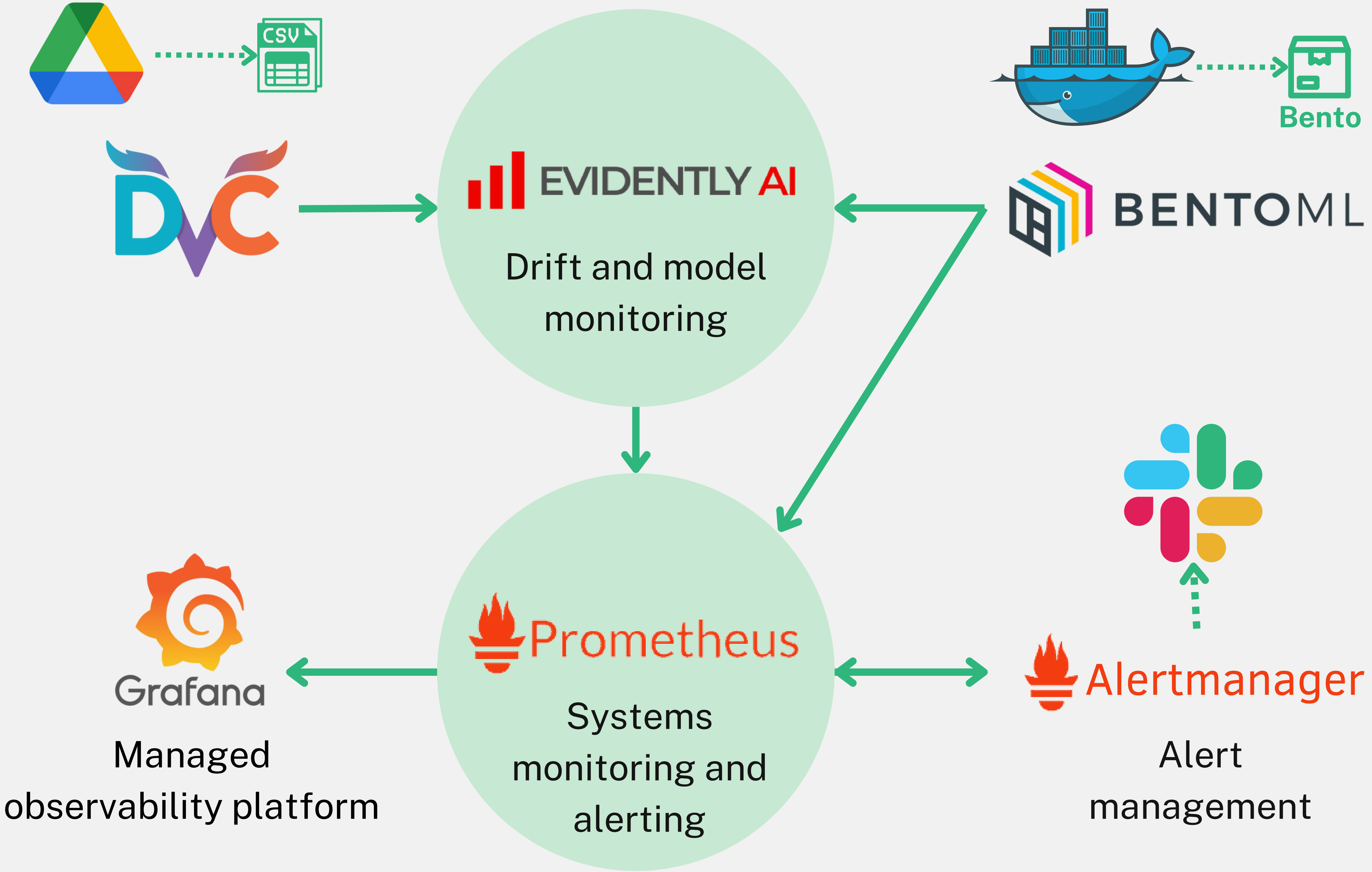
Grafana

Building application

Streamlit

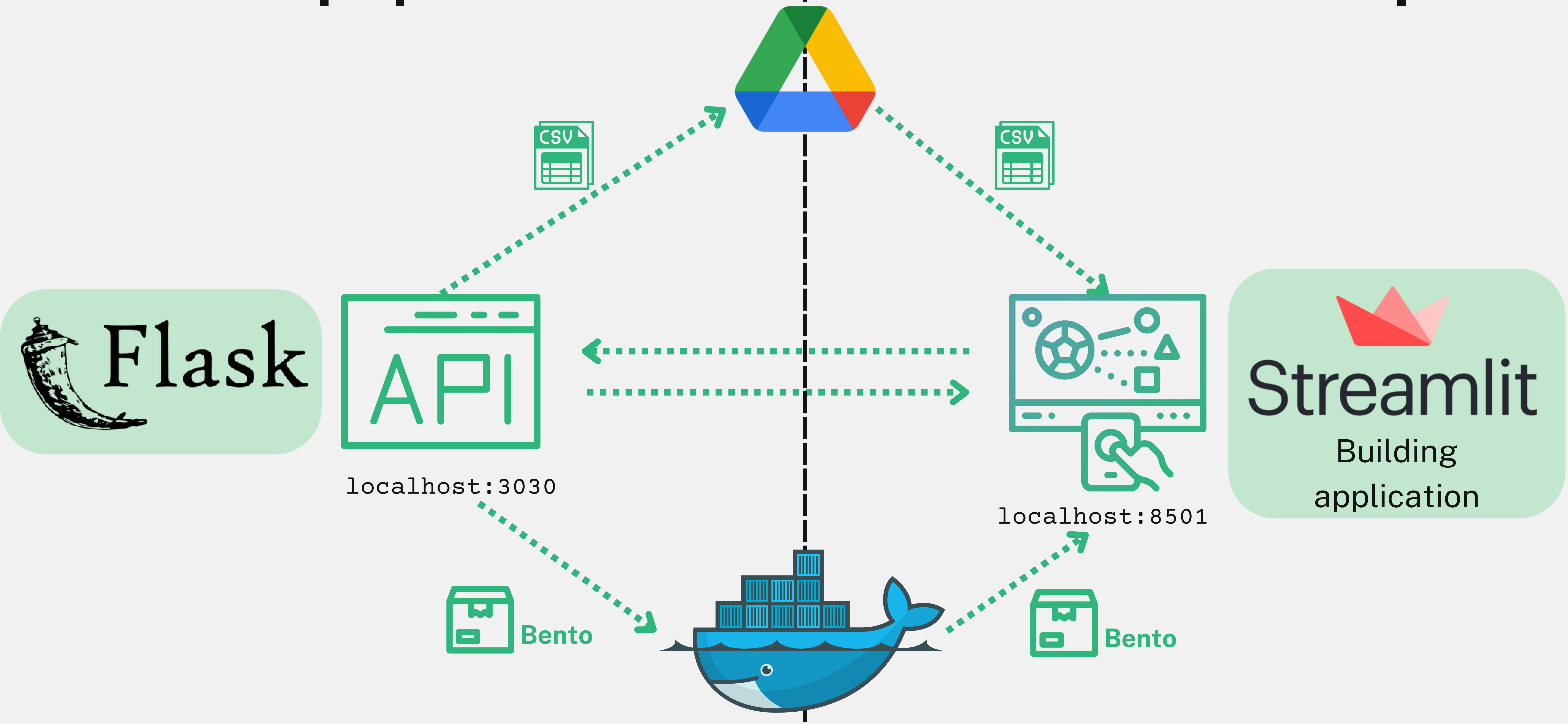


Develop phase



Production phase

Develop phase | Production phase



Commands line

Develop phase file: run.py

Commands line	Develop phase file: run.py
python run.py pipeline	open kedro ui to localhost:4141
python run.py exp	open mlflow ui to localhost:5000
python run.py pipeline-docker	create docker image about pipeline to create ml model
python run.py run	run training model and building bento service
python run.py new-dataset <global_url>	upload dataset from file csv with <global_url>

Flask app

Develop phase file: app.py

`@app.get("/dvc_file")`	return the code for take reference dataset from Google Drive.
`@app.route("/load_new_data", methods=["POST"])`	receive a dataframe to load as new reference dataset.
`@app.get("/retrain")`	load in Google drive new dataset, run mlflow to training model, build and dockerize bento.
`@app.get("/bento")`	build and dockerize bento.
`@app.get("/header")`	return name columns of dataset header.

MLOps Stack Canvas

Project name: MLOps tirocinio

Team members: Giorgia Bertacchini

Data analysis & experiment management

For data analysis uses Pandas, a Python library. For experiment management uses MLFlow, an open source platform.

Feature store & workflows

Feature are manipulate by Pandas, to work with Dataframe, and are collect by Kedro versioning they.

Data Sources & Data Versioning

Data sources are static file in table format, as csv. Use specialized data versioning systems as DVC to save in a Google Drive folder the raw dataset. Another system use is Kedro that save each versions of intermediate and output data and of ml model.

Value Proposition

This project would predict the quality of each case receive from csv file.

MLOps Dilemmas

Tooling: are all open-source.
Platforms: using a hybrid solution, with more platforms.

Foundations

For source version control system uses GitHub.

ML pipeline orchestration

The DAG (directed acyclic graph) is create by Kedro tool, an open-source Python framework.

Model & Data & Application Monitoring

To observing the model there are Evidently and Prometheus tools, which can alerts user and Grafana provide a dashboard for users.

Model registry & Model versioning

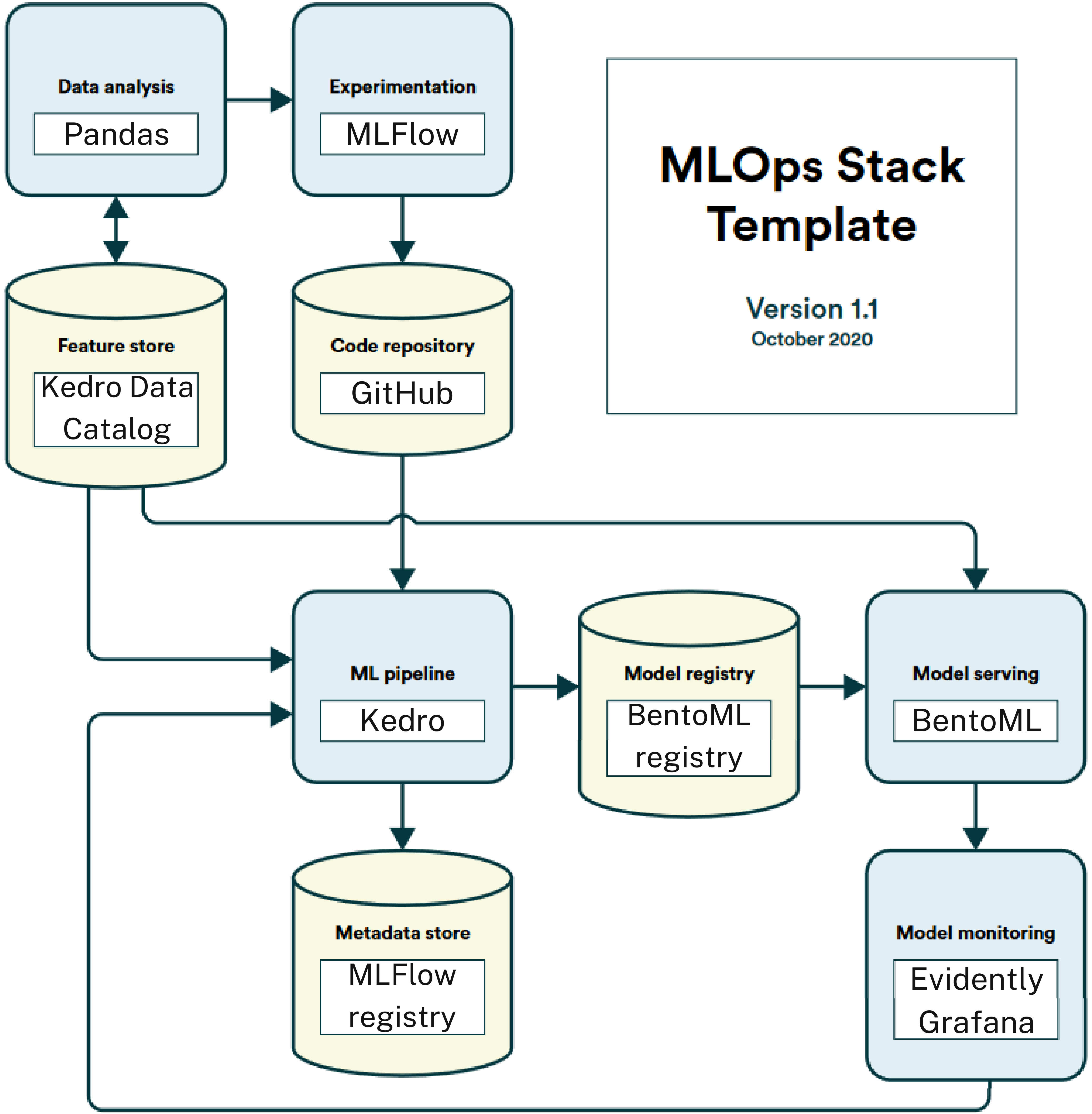
All model are collect in MLFlow ui and in BentoML platform.

Model deployment

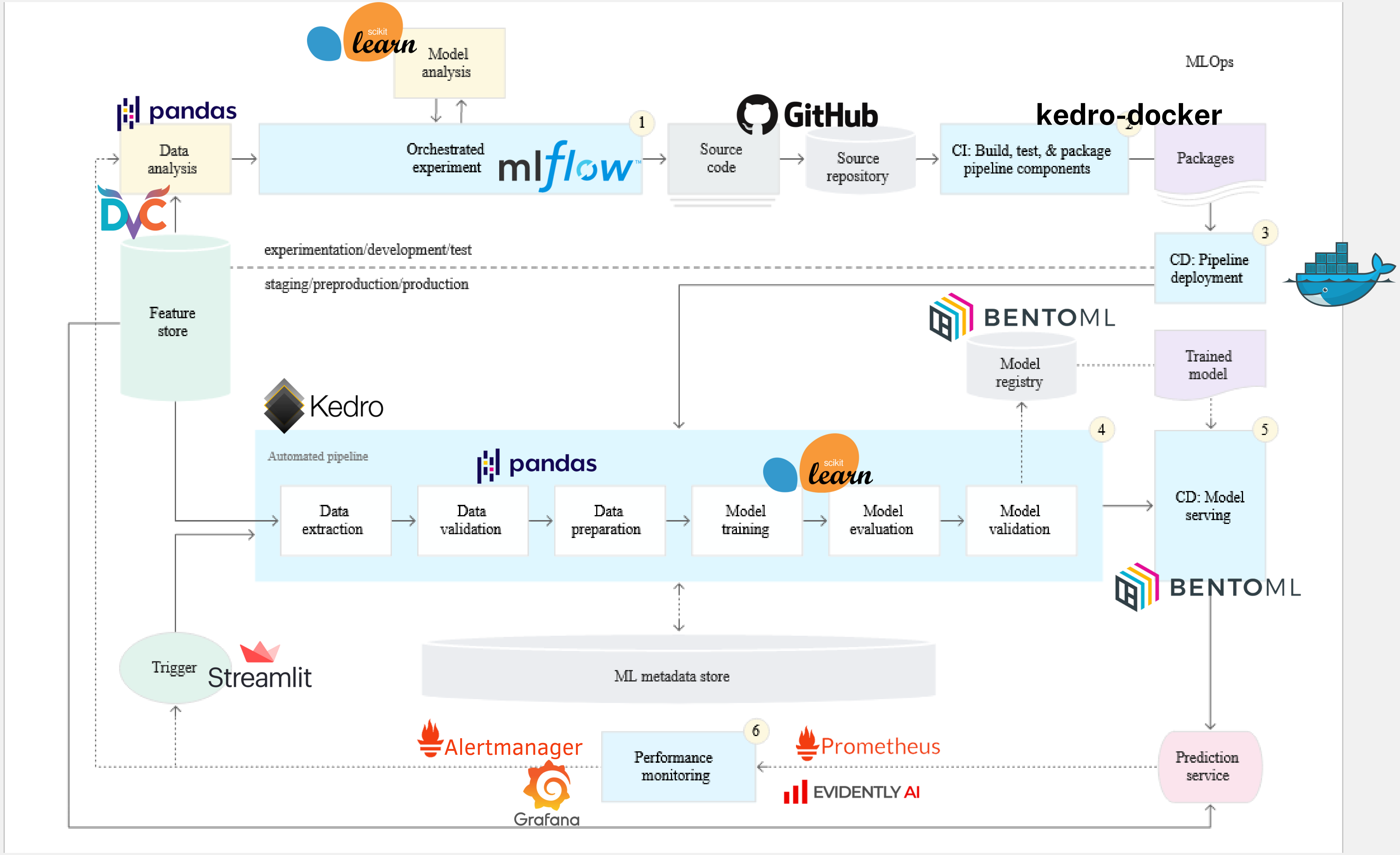
Ml model and its environment are deployment through BentoML, that create deployable artifacts (Bentos).

Prediction serving

BentoML provide a service, to require predictions to ml model in Bentos.


Data analysis
Pandas
Experimentation
MLFlow
MLOps Stack Template
Version 1.1
October 2020
Feature store
Kedro Data Catalog
Code repository
GitHub
ML pipeline
Kedro
Model registry
BentoML registry
Model serving
BentoML
Metadata store
MLFlow registry
Model monitoring
Evidently
Grafana

MLOps stack

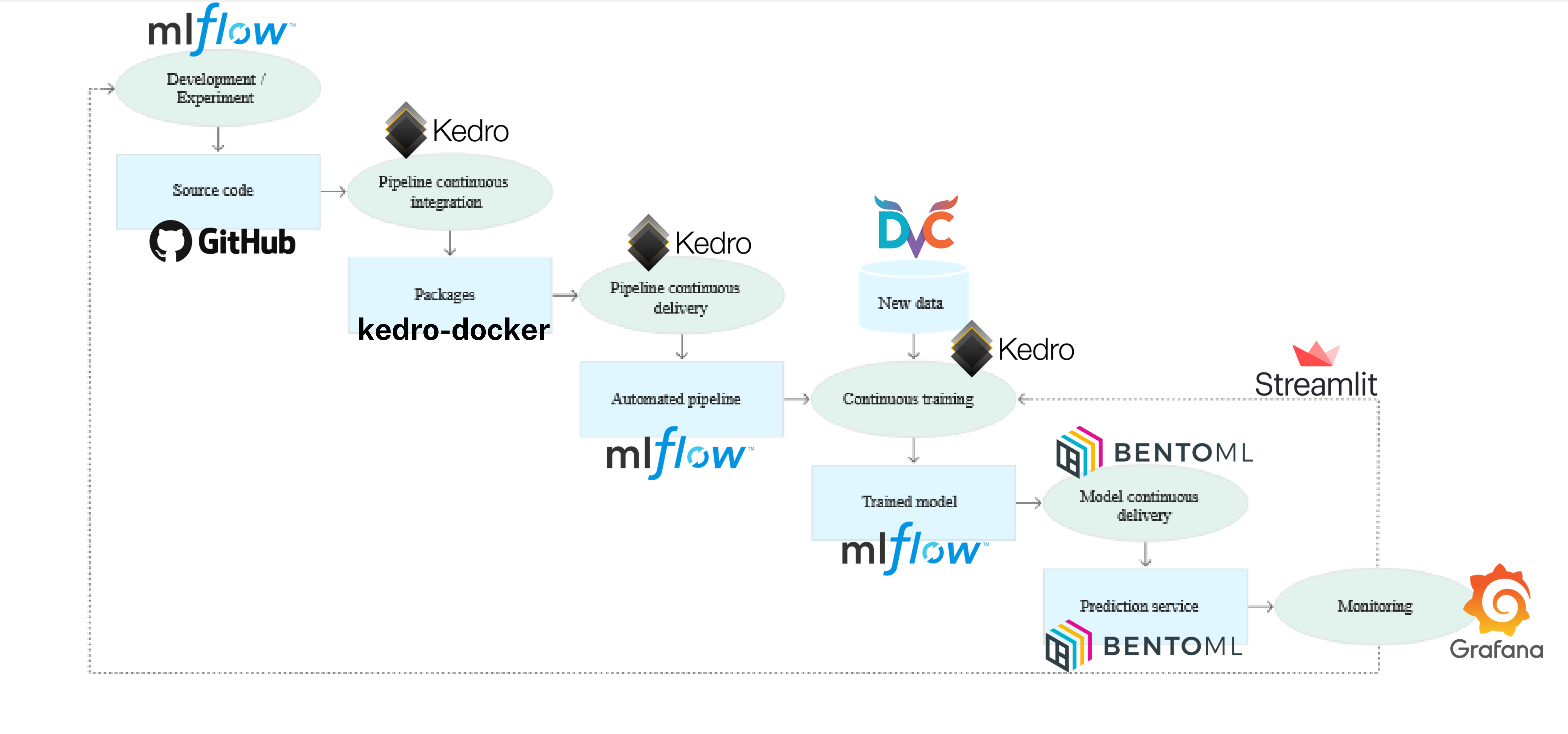


MLOps
Model analysis
Data analysis
Orchestrated experiment
1
Source code
Source repository
CI: Build, test, & package pipeline components
2
Packages
3
CD: Pipeline deployment
experimentation/development/test
staging/preproduction/production
Feature store
Model registry
Trained model
Automated pipeline
Data extraction
Data validation
Data preparation
Model training
Model evaluation
Model validation
4
5
CD: Model serving
ML metadata store
Trigger
6
Performance monitoring
Prediction service
pandas
scikit learn
mlflow
GitHub
kedro-docker
DVC
BENTOML
Kedro
Streamlit
Alertmanager
Grafana
Prometheus
EVIDENTLY AI

MLOps stack: Specifications

mlflow GitHub	**Development and experimentation**: you iteratively try out new ML algorithms and new modeling where the experiment steps are orchestrated. The output of this stage is the source code of the ML pipeline steps, which are then pushed to a source repository.
scikit learn Kedro	**Pipeline continuous integration**: The outputs of this stage are pipeline components (packages, executables, and artifacts) to be deployed in a later stage. Model validation via cross-validation for choose hyperparameters.
kedro-docker	**Pipeline continuous delivery**: you deploy the artifacts produced by the CI stage to the target environment. The output of this stage is a deployed pipeline with the new implementation of the model.
Streamlit	**Automated triggering**: the pipeline is automatically executed in production based on a schedule or in response to a trigger. The output of this stage is a newly trained model that is pushed to the model registry.
BENTOML	**Model continuous delivery**: you serve the trained model as a prediction service for the predictions. The output of this stage is a deployed model prediction service.
Prometheus Grafana	**Monitoring**: you collect statistics on model performance based on live data. The output of this stage is a trigger to execute the pipeline or to execute a new experiment cycle.

MLOps stack



mlflow
Development / Experiment
Source code
GitHub
Kedro
Pipeline continuous integration
Packages
kedro-docker
Kedro
Pipeline continuous delivery
Automated pipeline
mlflow
DVC
New data
Kedro
Continuous training
Streamlit
Trained model
mlflow
BENTOML
Model continuous delivery
Prediction service
BENTOML
Monitoring
Grafana

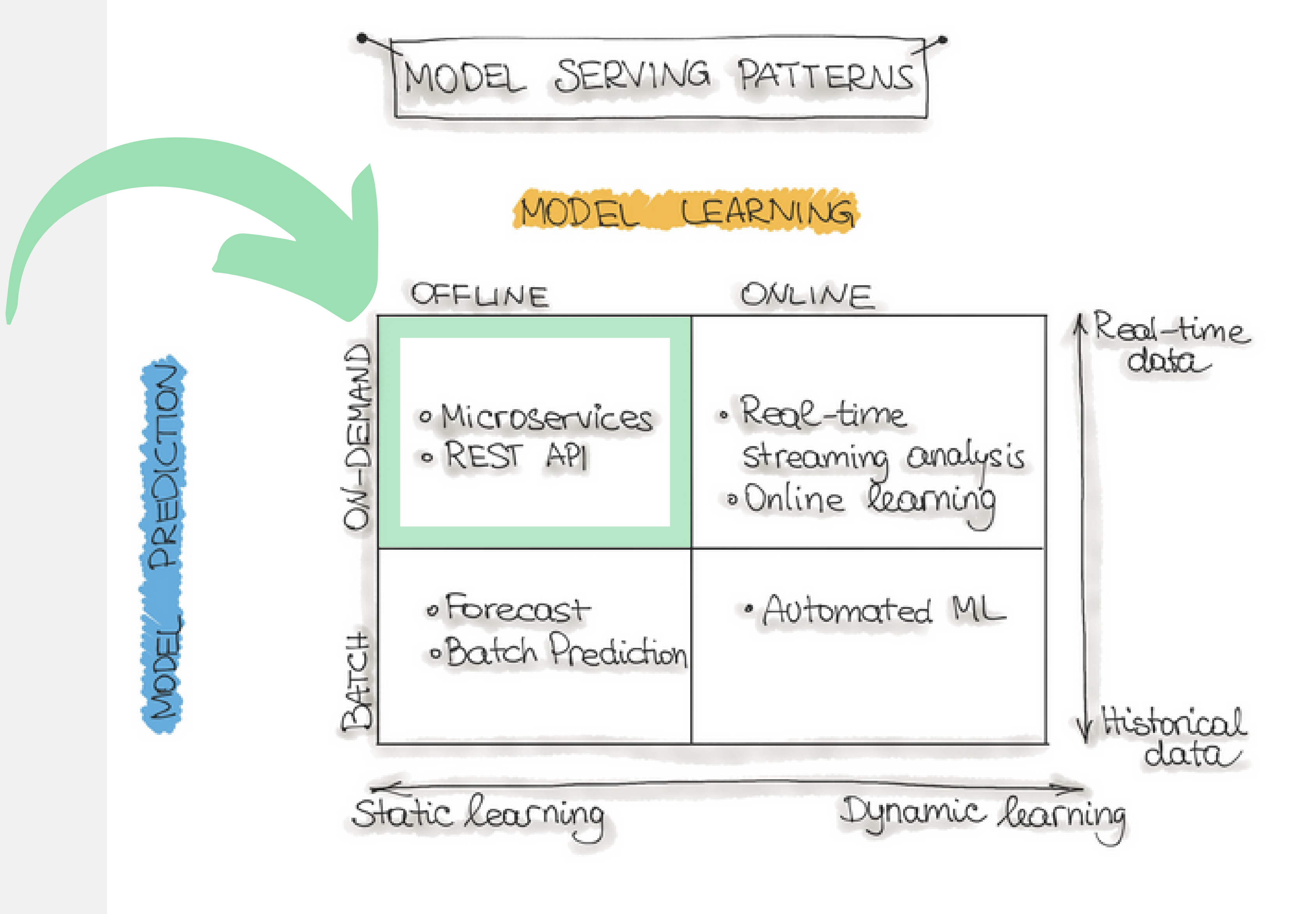
To classified the project

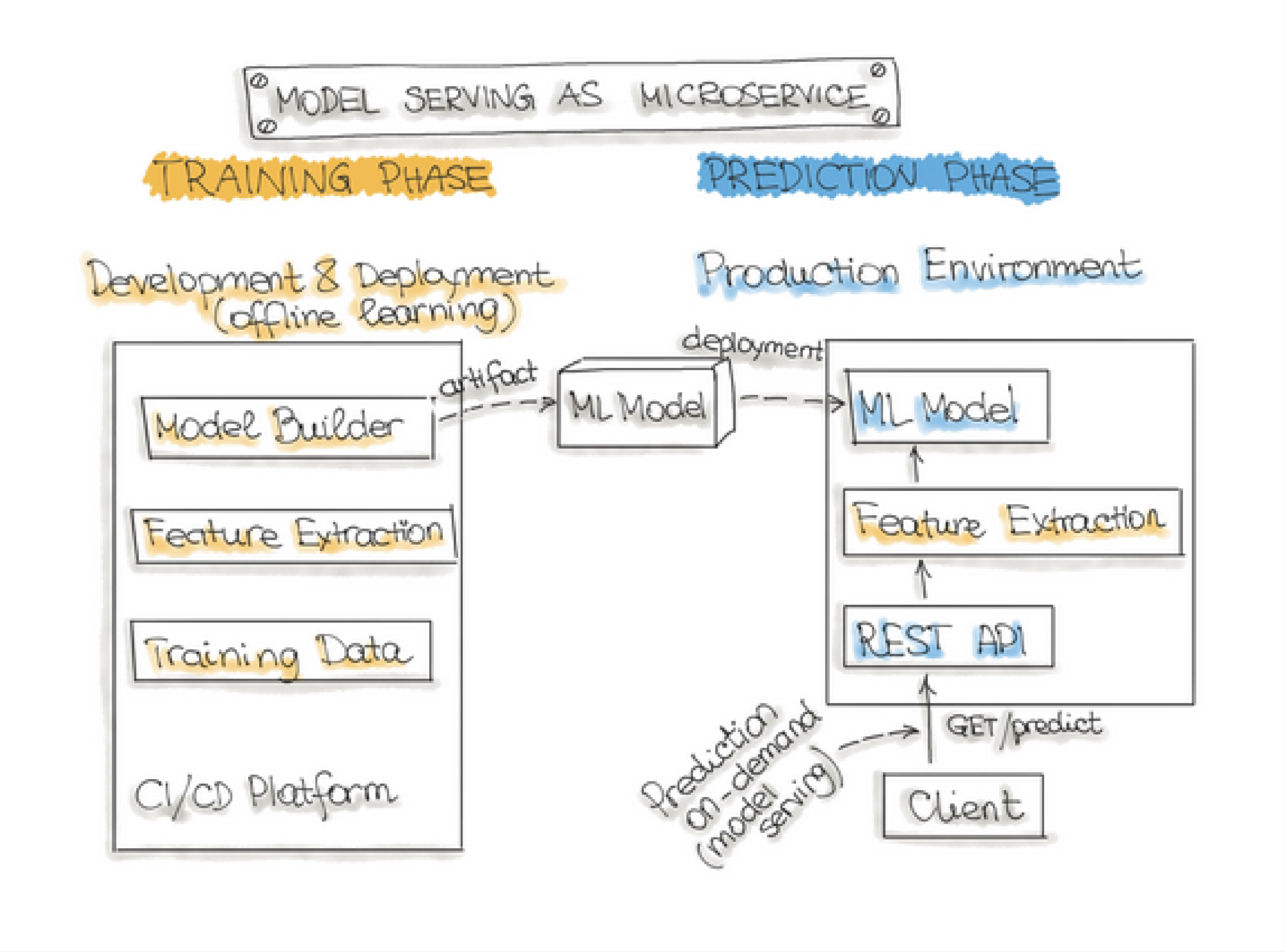
This project have:

- Model learning Offline
- Model predictions On-Demand

So use Microservices and Rest API application.

As we can see the next schema is very similar to previous schema, where the two phase are separate but connected.





GitHub url

All project and documentation:
https://github.com/giorgiaBertacchini/MLOps

Develop phase:
https://github.com/giorgiaBertacchini/MLOps-DevelopPhase
Production phase:
https://github.com/giorgiaBertacchini/MLOps-ProductionPhase