



Machine Learning Operations (MLOps)

Overview, Definition, Architecture and Implementation

Abolfazl Yarian
Spring 2023

Outline

Introduction

MLOps definition

Open-source tools

1

3

5

2

4

6

Data science
Overview

Architecture

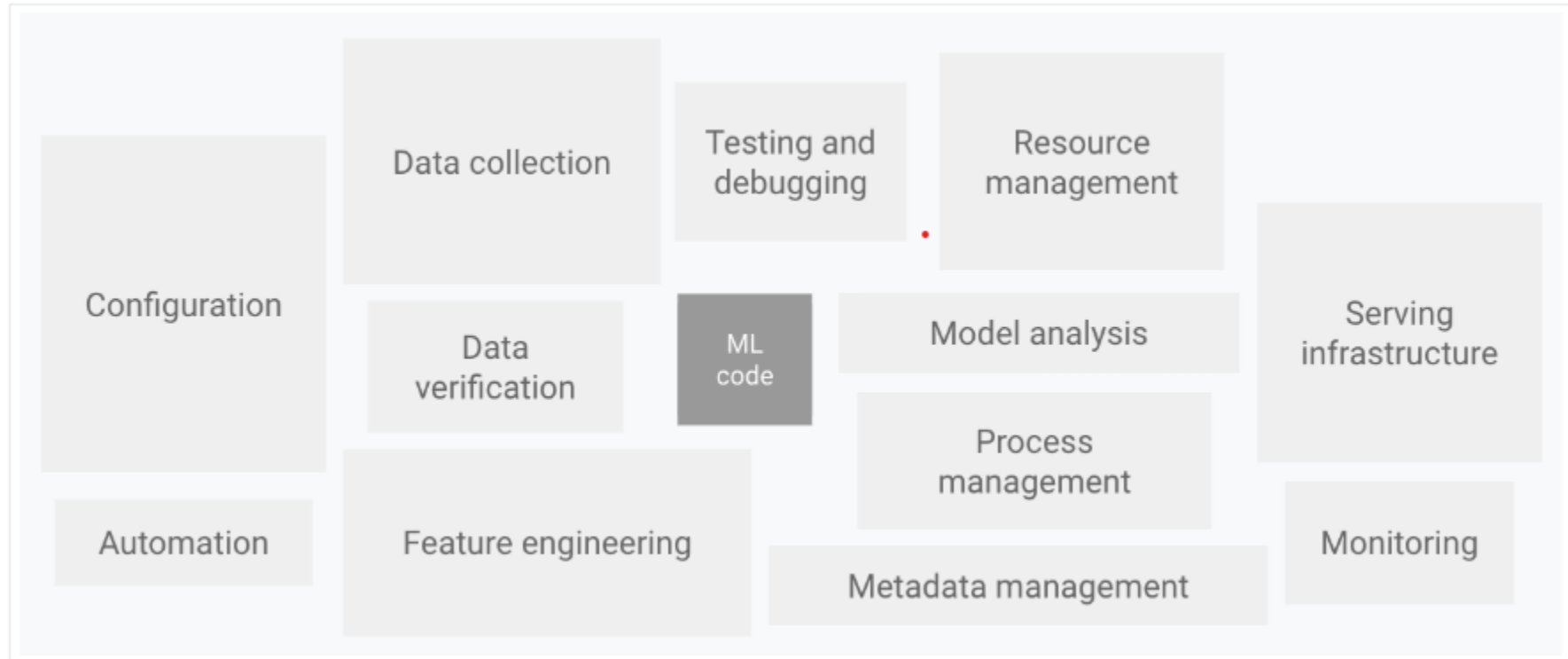
References

Data science steps for ML

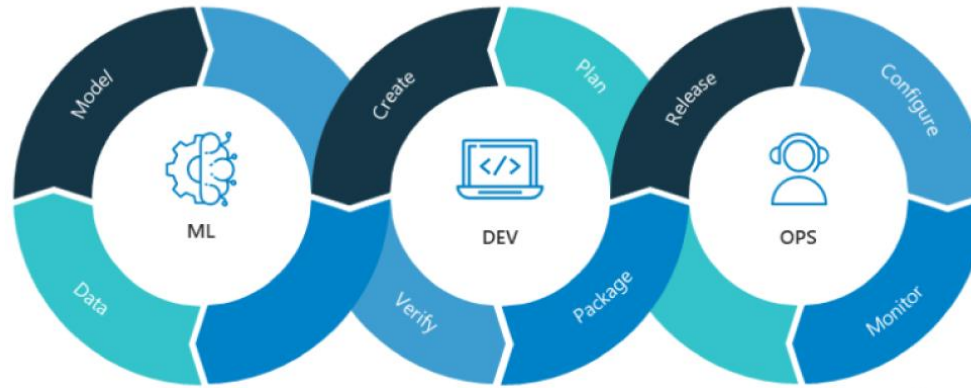
- ⊙ Data extraction
- ⊙ Data analysis
- ⊙ Data preparation
 - Data cleaning
 - Data splitting
 - Transformation and feature engineering

Data science steps for ML

- ◎ Model training
 - Implement different algorithm
 - Hyperparameter tuning
- ◎ Model evaluation/validation
- ◎ Model serving
 - Microservices
 - Edge device
- ◎ Model monitoring



MLOps



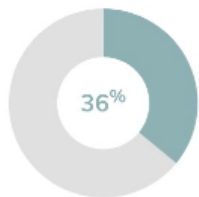
MLOps (Machine Learning Operations) refers to the practice of applying DevOps (Development Operations) principles to the machine learning workflow. It involves a set of processes, tools, and techniques to build, deploy, monitor, and manage machine learning models in production environments

MLOps

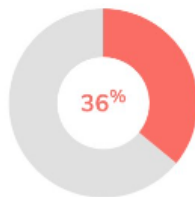
- ◎ MLOps manages and automates the end-to-end lifecycle of machine learning models.
- ◎ Combines DevOps and data science to streamline development, deployment, and monitoring.
- ◎ Improves collaboration, efficiency, and scalability by standardizing tools, processes, and infrastructure.
- ◎ Enables continuous integration and delivery, ensuring reliability, security, and performance in production.
- ◎ Involves monitoring, testing, and updating ML models to remain accurate and relevant.
- ◎ Accelerates innovation, reduces costs, and improves customer satisfaction.

Responses from 582 survey respondents

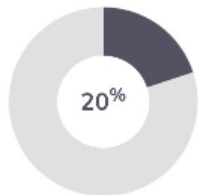
What percentage of your data scientists' time is spent deploying ML models?



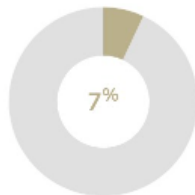
36% of survey participants said their data scientists spend **a quarter** of their time deploying ML models



36% of survey participants said their data scientists spend **a quarter to half** of their time deploying ML models



20% of survey participants said their data scientists spend **half to three-quarters** of their time deploying ML models



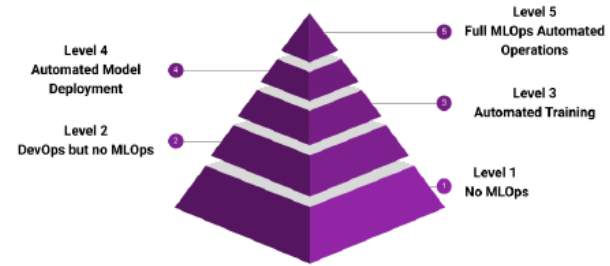
7% of survey participants said their data scientists spend **more than three-quarters** of their time deploying ML models

1% of respondents said they were unsure.

MLOps Maturity Levels

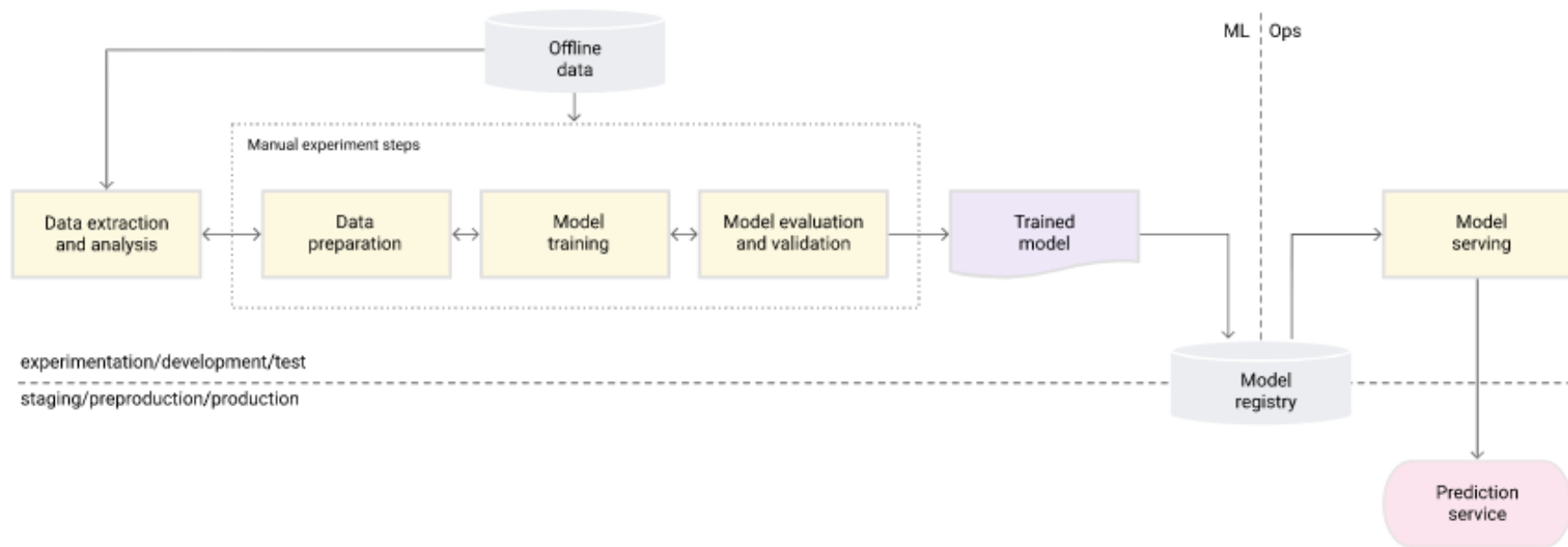


Googles Maturity Levels.

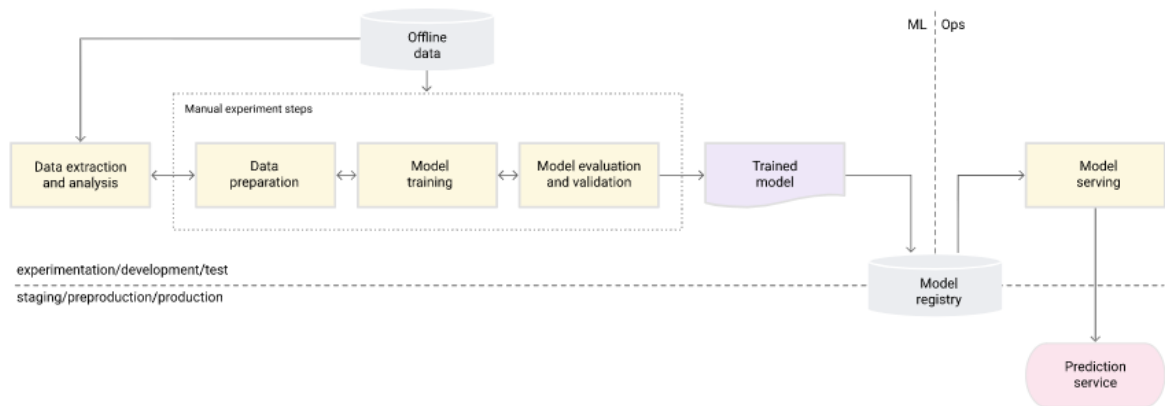


Microsoft Maturity Levels.

MLOps level 0: Manual process



Characteristics

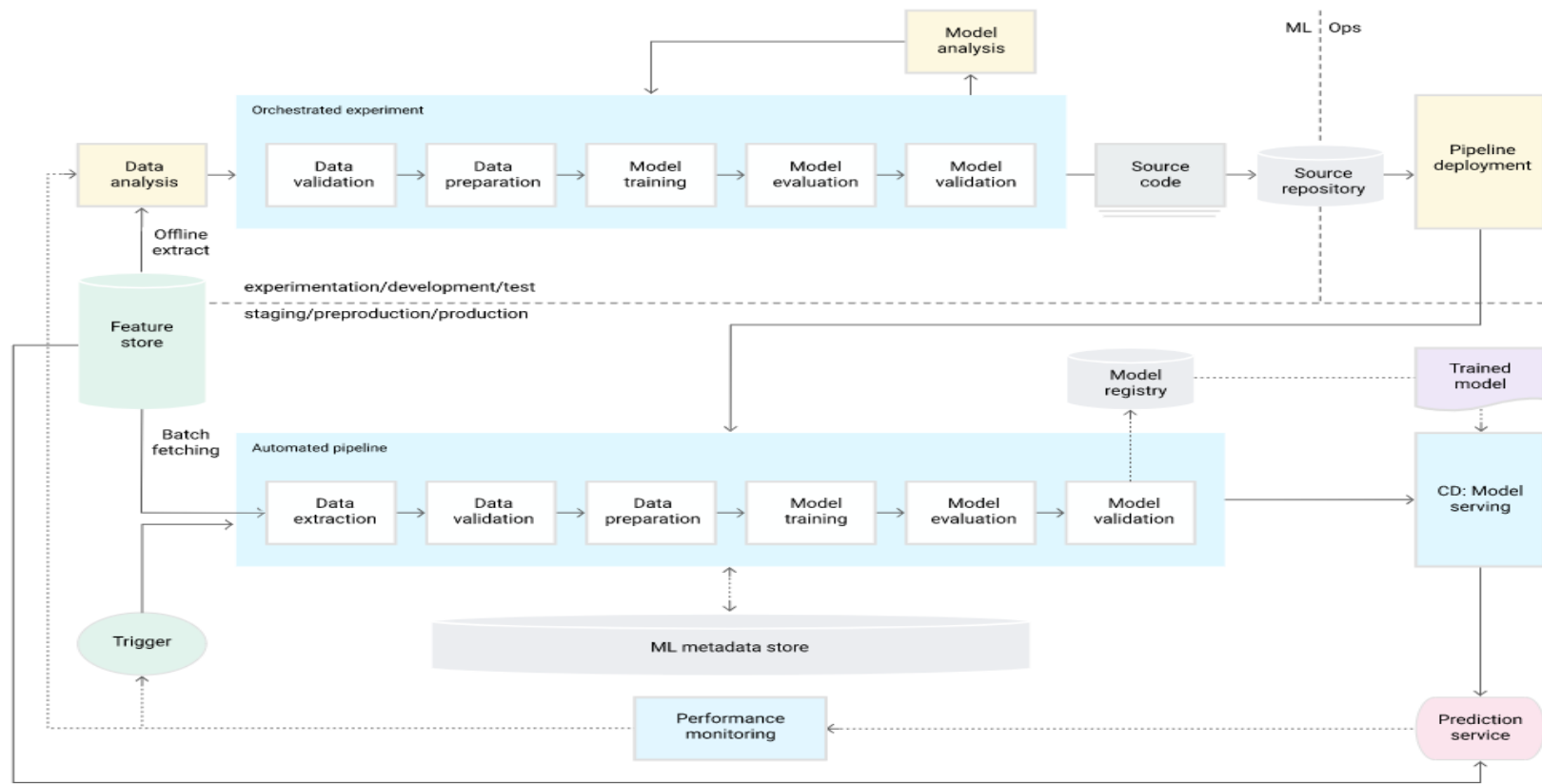


- ⊙ Manual, script-driven, and interactive process
- ⊙ Disconnection between ML and operations
- ⊙ Infrequent release iterations
- ⊙ No CI/CD
- ⊙ Deployment refers to the prediction service
- ⊙ Lack of active performance monitoring

Challenges

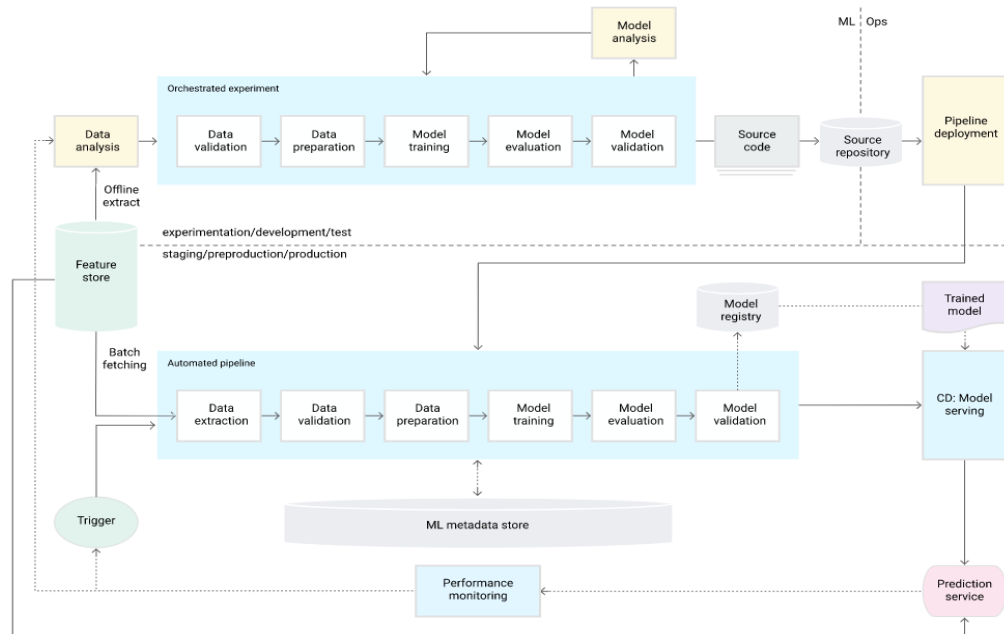
- ◎ Maintain model's accuracy in production
 - Actively monitor the quality of your model in production
 - Frequently retrain your production models
 - Continuously experiment with new implementations to produce the model
- ◎ Set up CT/CI/CD to rapidly test, build and deploy new implementation of the ML pipeline

MLOps level 1: ML pipeline automation(CT)



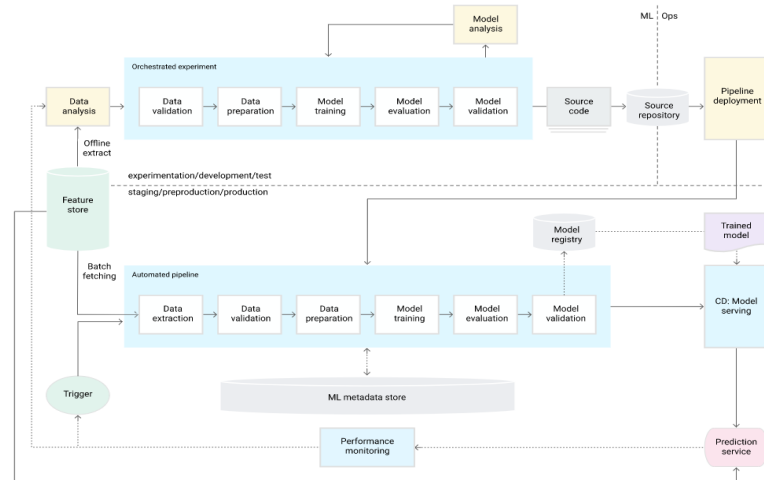
Data validation

- ◎ Data schema skews (stop)
 - Unexpected features
 - Unexpected values
 - Lack of all expected features
- ◎ Data value skew (retrain)



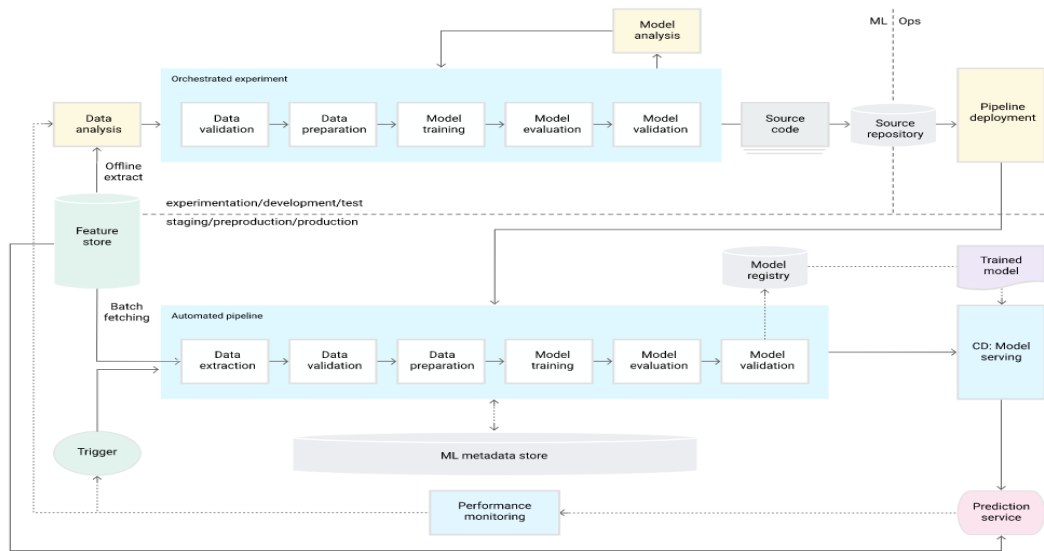
Model validation (offline)

- ⊙ Evaluate predictive quality on test dataset
- ⊙ Compare with current model performance
- ⊙ Check for consistency across data segments
- ⊙ Test for deployment and infrastructure compatibility
- ⊙ Conduct online validation through canary or A/B testing



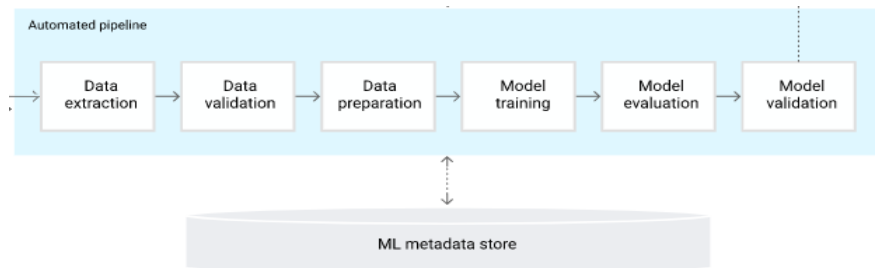
Feature store

- Discover and reuse existing feature sets to avoid duplication
- Serve up-to-date feature values from the feature store.
- Use the feature store for experimentation, CT, and online serving to avoid training-serving skew.
- avoid training-serving skew for:
 - Experimentation (offline)
 - continuous training
 - Online prediction

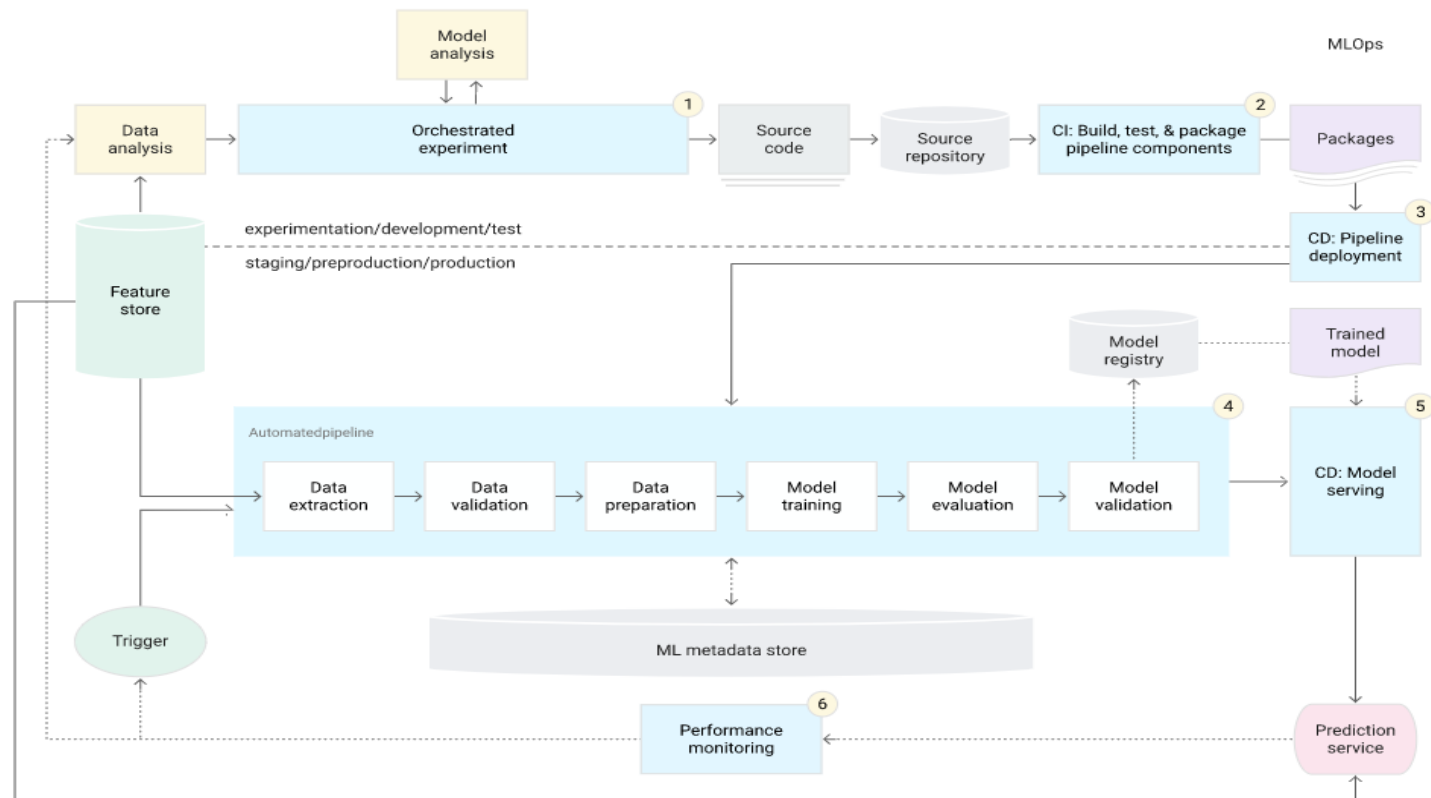


Metadata management

- ◎ Record pipeline versions, timestamps, and executor for lineage, reproducibility, and debugging
- ◎ Store parameter arguments passed to the pipeline
- ◎ Store pointers to the artifacts produced by each step of the pipeline
- ◎ Store pointers to previous models and evaluation metrics for comparison

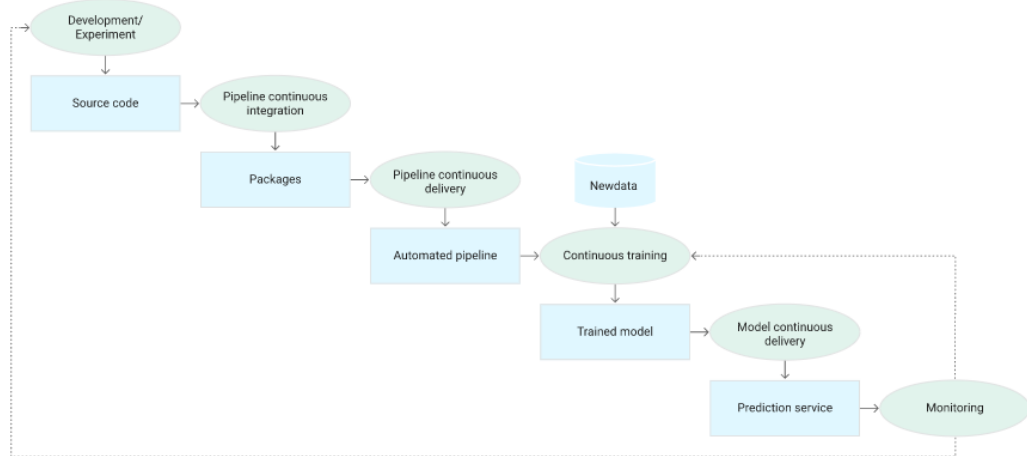


MLOps level 2: CI/CD pipeline automation



Characteristics

- ⦿ Development and experimentation
- ⦿ Pipeline continuous integration
- ⦿ Pipeline continuous delivery/deployment
- ⦿ Automated triggering
- ⦿ Model continuous delivery
- ⦿ Monitoring



Continuous integration

- ① Unit tests for feature engineering and model methods in implementation
- ① Tests for model convergence and avoiding NaN values
- ① Testing that each component in the pipeline produces the expected artifacts
- ① Testing integration between pipeline components.

Continuous delivery

- ⦿ Verify the compatibility of the model with the target infrastructure before deployment
- ⦿ Test the prediction service by calling the service API with expected inputs
- ⦿ Test prediction service performance by load testing to capture metrics such as QPS and model latency
- ⦿ Validate data for retraining or batch prediction
- ⦿ Verify that models meet predictive performance targets before deployment
- ⦿ Automate deployment to a test environment triggered by code push to development branch

Principles & Technical Components



COMPONENT

Principles

- ◎ **P1 CI/CD automation:** Build, Test, Delivery, Deployment
fast feedback to developers regarding the success or failure of certain steps
- ◎ **P2 Workflow orchestration (containerization)**
coordinates the tasks of an ML workflow pipeline according to DAGs (tasks and dependencies)
- ◎ **P3 Reproducibility**
reproduce an ML experiment and obtain the exact same results
- ◎ **P4 Versioning:** data, model, code
reproduce and traceability
- ◎ **P5 Collaboration:**
possibility of work collaboratively on data, model, and code

Principles

◎ P6 Continuous ML training (CT) & evaluation:

Periodic retraining of the ML model based on new feature data

Assess the change in model quality

Managing costs of retraining:

- Update frequency (e.g. daily vs. weekly)
- Online learning in large scale web application

◎ P7 ML metadata tracking/logging:

- Training job iteration (e.g. training code and time, duration)
- Model specific metadata (e.g. used parameters and resulting performance metrics)

Principles

◎ P8 Continuous monitoring

periodic assessment of data, model, code, infrastructure resources, and model serving performance (e.g., prediction accuracy)

◎ P9 Feedback loops

- the monitoring component to the scheduler to enable the retraining
- quality assessment step into the development or engineering process

Technical components

◎ C1 CI/CD Component (P1, P6, P9)

- Jenkins, GitLab CI/CD

◎ C2 Source Code Repository (P4, P5)

- GitLab, GitHub, FastDS, DagsHub

◎ C3 Workflow Orchestration Component (P2, P3, P6)

- Apache Airflow, Kubeflow,
- AWS SageMaker Pipelines, Azure Pipeline

PRINCIPLES

P1 CI/CD automation

P2 Workflow orchestration

P3 Reproducibility

P4 Versioning of data, code, model

P5 Collaboration

P6 Continuous ML training & evaluation

P7 ML meta data tracking

P8 Continuous monitoring

P9 Feedback loops

Technical components

◎ C4 Feature Store System (P3, P4)

- Feast, Hopswork, Tecton, Gojek's Feast
- AWS Feature Store

Central storage of commonly used features

Two databases configured:

- Offline feature store to serve features with normal latency for experimentation
- Online feature store to serve features with low latency for predictions in production

◎ C5 Model training Infrastructure (P6)

- Kubernetes, Red Hat OpenShift

CPU, RAM, GPU

Distributed or non-distributed

PRINCIPLES

- P1 CI/CD automation
- P2 Workflow orchestration
- P3 Reproducibility
- P4 Versioning of data, code, model
- P5 Collaboration
- P6 Continuous ML training & evaluation
- P7 ML meta data tracking
- P8 Continuous monitoring
- P9 Feedback loops

Technical components

◎ C6 Model Registry (P3, P4)

Store trained ML models with their metadata

- MLFlow, ModelDB
- AWS SageMaker Model Registry, Azure Model Registry, Neptune.ai

◎ C7 ML Metadata Stores (P4, P7)

- Kubeflow pipelines,
- AWS SageMaker Pipelines, Azure ML

PRINCIPLES

- P1 CI/CD automation
- P2 Workflow orchestration
- P3 Reproducibility
- P4 Versioning of data, code, model
- P5 Collaboration
- P6 Continuous ML training & evaluation
- P7 ML meta data tracking
- P8 Continuous monitoring
- P9 Feedback loops

Technical components

◎ C8 Model Serving Component (P1) ?????

- KServing of Kubeflow, Tensorflow Serving, Seldon
- Azure ML REST API, AWS SageMaker Endpoints
- online inference for real-time predictions
- Batch inference for predictions using large volumes of input data
- Use Kubernetes and docker technology to container the ML model

◎ C9 Monitoring Component (P8, P9)

- Prometheus with Grafana, Kubeflow, Mlflow, ELK stack
- AWS SageMaker model monitor

continuous monitoring of:

the model serving performance (e.g., prediction accuracy)

the ML infrastructure, CI/CD, and orchestration

PRINCIPLES

- P1 CI/CD automation
- P2 Workflow orchestration
- P3 Reproducibility
- P4 Versioning of data, code, model
- P5 Collaboration
- P6 Continuous ML training & evaluation
- P7 ML meta data tracking
- P8 Continuous monitoring
- P9 Feedback loops

Tools

Name	Status	Launched in	Use
iMerit	Private	2012	Data Preprocessing
Pachyderm	Private	2014	Data Versioning
Labelbox	Private	2017	Data Preprocessing
Prodigy	Private	2017	Data Preprocessing
Comet	Private	2017	Data Versioning
Data Version Control	Open Source	2017	Data Versioning
Qri	Open Source	2018	Data Versioning
Weights and Biases	Private	2018	Data Versioning
Delta Lake	Open Source	2019	Data Versioning
Doccano	Open Source	2019	Data Preprocessing
Snorkel	Private	2019	Data Preprocessing
Supervisely	Private	2019	Data Preprocessing
Segments.ai	Private	2020	Data Preprocessing
Dolt	Open Source	2020	Data Versioning
LakeFs	Open Source	2020	Data Versioning

Table I
DATA PREPROCESSING TOOLS.

Name	Status	Launched in	Use
Hyperopt	Open Source	2013	Hyperparameter Optimization
SigOpt	Public	2014	Hyperparameter Optimization
Iguazio Data Science Platform	Private	2014	Feature Engineering
Tsfresh	Private	2016	Feature Engineering
Featuretools	Private	2017	Feature Engineering
Comet	Private	2017	Experiment Tracking
Neptune.ai	Private	2017	Experiment Tracking
TensorBoard	Open source	2017	Experiment Tracking
Google Vizier	Public	2017	Hyperparameter Optimization
Scikit-Optimize	Open source	2017	Hyperparameter Optimization
dotData	Private	2018	Feature Engineering
Weight and Biases	Private	2018	Experiment Tracking
CML	Open source	2018	Experiment Tracking
MLFlow	Open source	2018	Experiment Tracking
Optuna	Open source	2018	Hyperparameter Optimization
Talos	Open Source	2018	Hyperparameter Optimization
AutoFet	Open Source	2019	Feature Engineering
Feast	Private	2019	Feature Engineering
GuildAi	Open Source	2019	Experiment Tracking
Rasgo	Private	2020	Feature Engineering
ModelDB	Open source	2020	Experiment Tracking
HopsWork	Private	2021	Feature Engineering
Aim	Open source	2021	Experiment Tracking

Table II
MODELING TOOLS.

Name	Status	Launched in	Use
Google Cloud Platform	Public	2008	end-to-end
Microsoft Azure	Public	2010	end-to-end
H2O.ai	Open source	2012	end-to-end
Unravel Data	Private	2013	Model Monitoring
Algorhythmia	Private	2014	Model Deployment / Serving
Iguazio	Private	2014	end-to-end
Databricks	Private	2015	end-to-end
TensorFlow Serving	Open source	2016	Model Deployment / Serving
Featuretools	Private	2017	Feature Engineering
Amazon SageMaker	Public	2017	end-to-end
Kubeflow	Open Source	2018	Model Deployment / Serving
OpenVino	Open source	2018	Model Deployment / Serving
Triton Inference Server	Open source	2018	Model Deployment / Serving
Fiddler	Private	2018	Model Monitoring
Losswise	Private	2018	Model Monitoring
Alibaba Cloud ML Platform for AI	Public	2018	end-to-end
Mlflow	Open source	2018	end-to-end
BentoML	Open Source	2019	Model Deployment / Serving
Superwise.ai	Private	2019	Model Monitoring
MLrun	Open source	2019	Model Monitoring
DataRobot	Private	2019	end-to-end
Seldon	Private	2020	Model Deployment / Serving
Torch Serve	Open source	2020	Model Deployment / Serving
KFServing	Open source	2020	Model Deployment / Serving
Syndicai	Private	2020	Model Deployment / Serving
Arize	Private	2020	Model Monitoring
Evidently AI	Open Source	2020	Model Monitoring
WhyLabs	Open source	2020	Model Monitoring
Cloudera	Public	2020	end-to-end
BodyWork	Open source	2021	Model Deployment / Serving
Cortex	private	2021	Model Deployment / Serving
Sagify	Open source	2021	Model Deployment / Serving
Aporia	Open source	2021	Model Monitoring
Deep checks	Private	2021	Model Monitoring

Table III
OPERATIONALIZATION TOOLS.

DATA MANAGEMENT

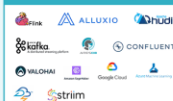
Data Exploration & Management



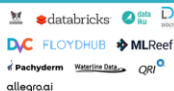
Data Labelling



Data Streaming



Data Version Control



Data Generation



Data Privacy



Data Quality Checks



MODELLING

Notebook / Code Management



Data Processing & Visualization



Feature Engineering



Model Training



Experiment Tracking



Model (Hyperparameter) Optimization



Auto ML



Model Management



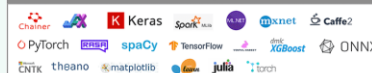
Model Evaluation



Model Explainability



Frameworks & major libraries



CONTINUOUS DEPLOYMENT

Data Flow Management



Feature Transformation



Monitoring



Model Compliance & Audit



Model Deployment & Serving



Model Validation



Model Compatibility



COMPUTING MANAGEMENT

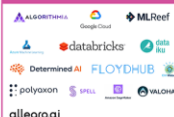
Computing & Data Infrastructure



Environment Management



Resource Allocation



SCALING



SECURITY & PRIVACY



Roles

◎ R1 Business Stakeholder

- Define business goal

◎ R2 Solution Architect

- Design the architecture and define the technologies

◎ R3 Data Scientist

- Translate business problem into an ML problem
- model engineering (best-performing algorithm + hyperparameter)

Roles

◎ R4 Data Engineer

- Build up and manage data and feature engineering pipelines
- proper data ingestion to the databases of the feature store system

◎ R5 Software Engineer

- Apply software design patterns
- widely accepted coding guidelines

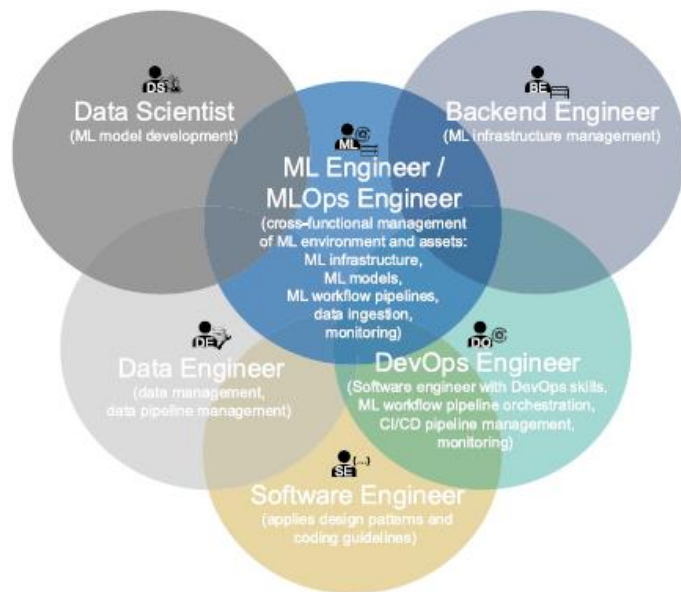
◎ R6 DevOps Engineer

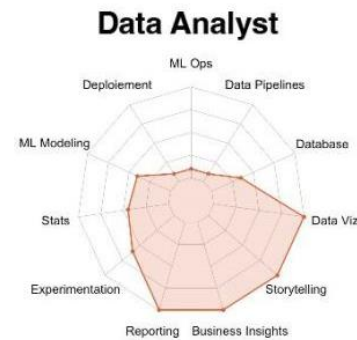
- CI/CD automation
- ML workflow orchestration
- model deployment to production
- monitoring

Roles

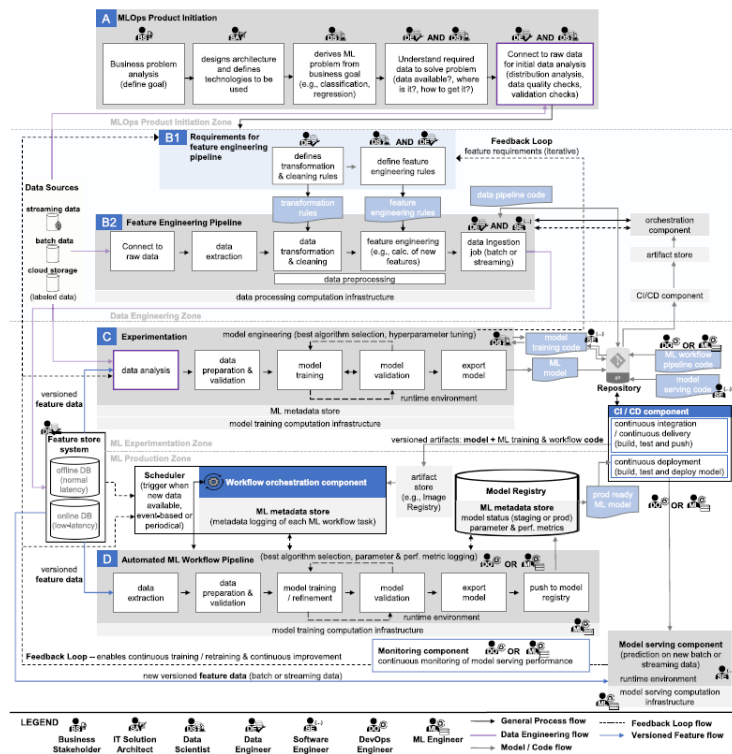
◎ R7 ML/MLOps Engineer:

- build up and operate the ML infrastructure
- manage the automated ML workflow pipelines
- model deployment to production
- monitor model and ML infrastructure

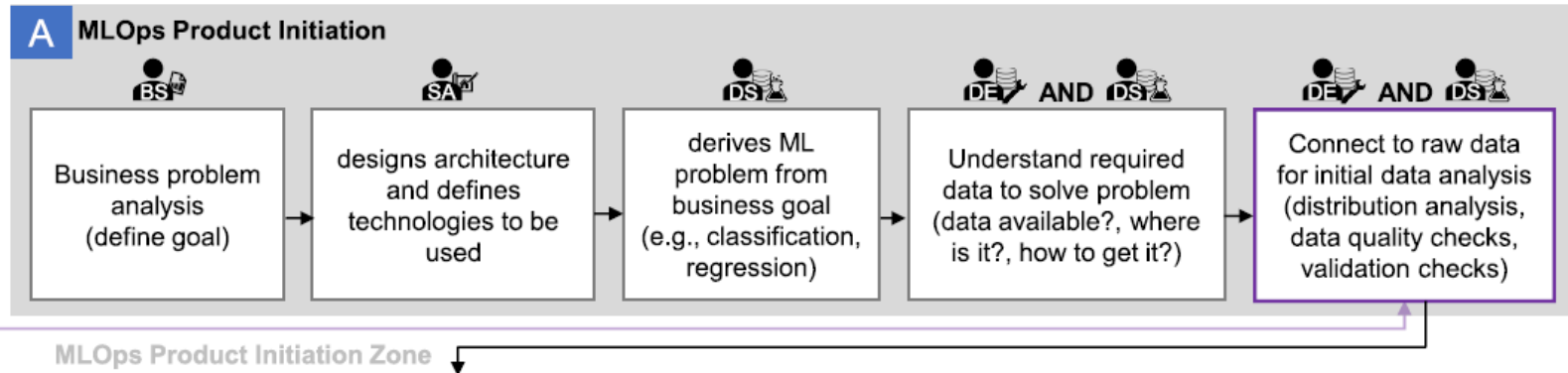




Architecture and Workflow

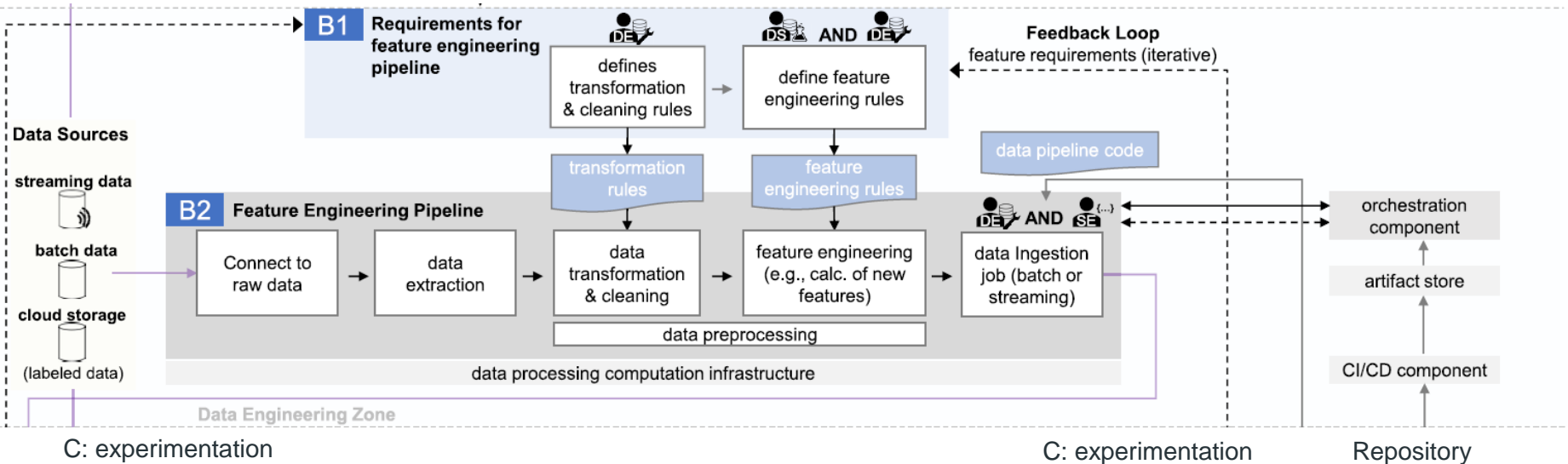


(A) MLOps Product Initiation

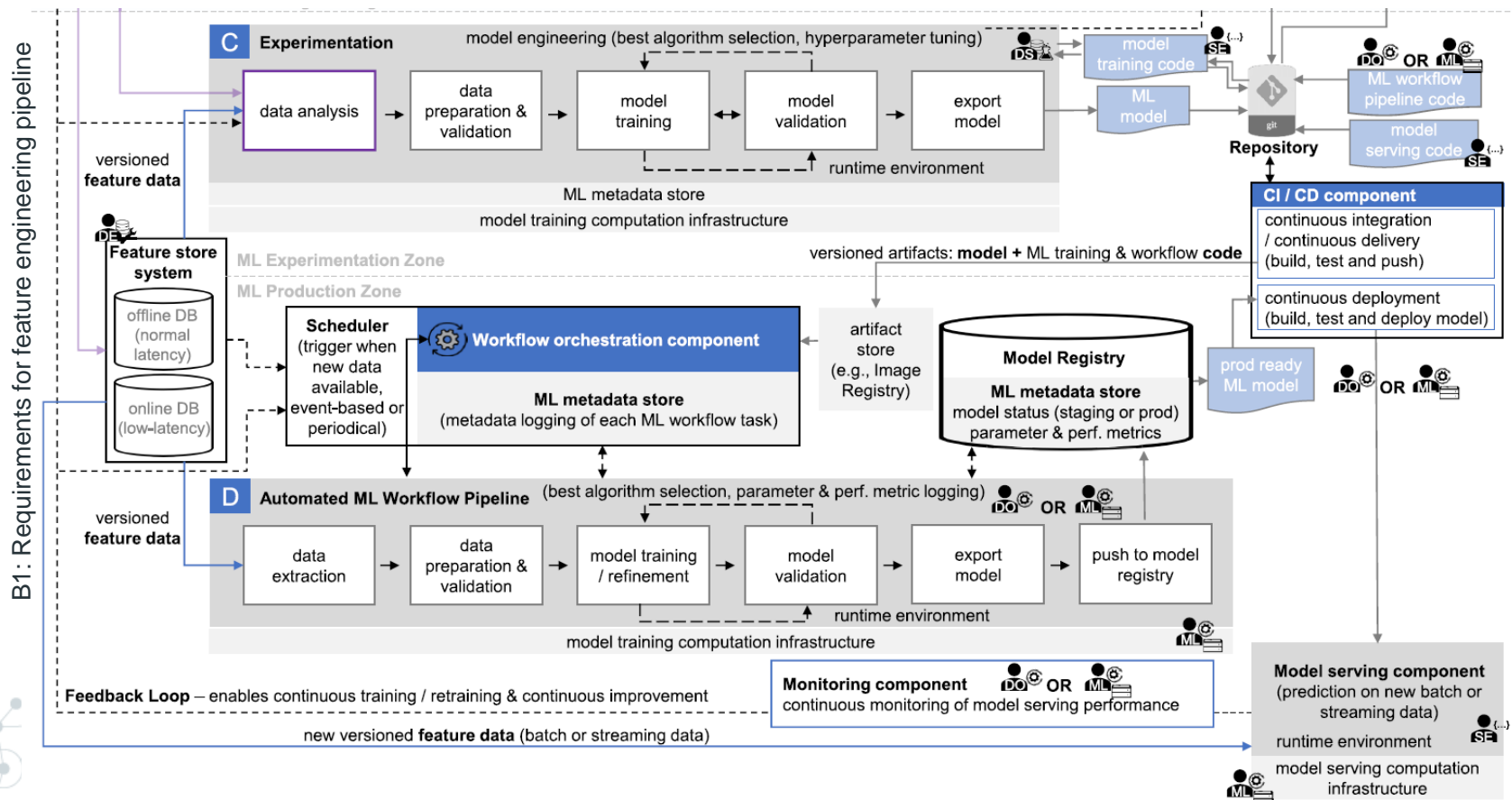


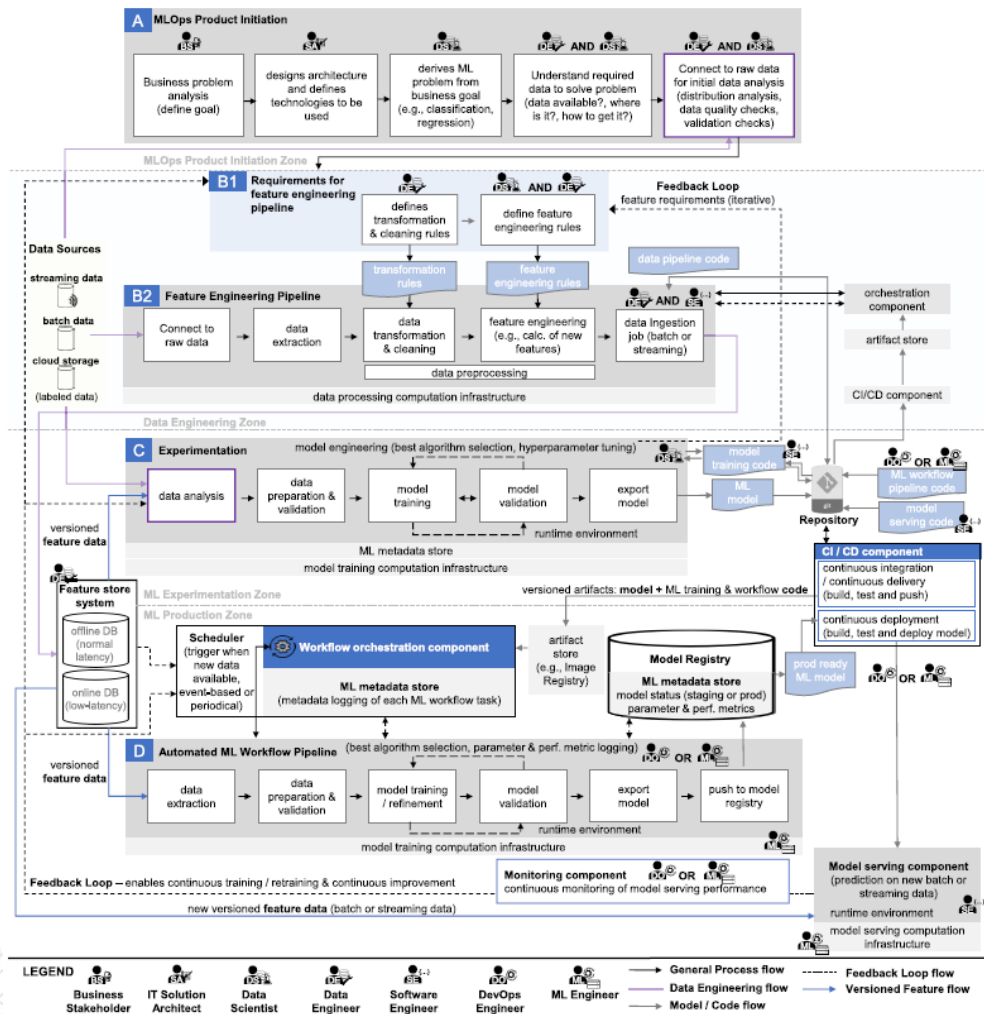
(B) Feature Engineering Pipeline

A: initial data analysis

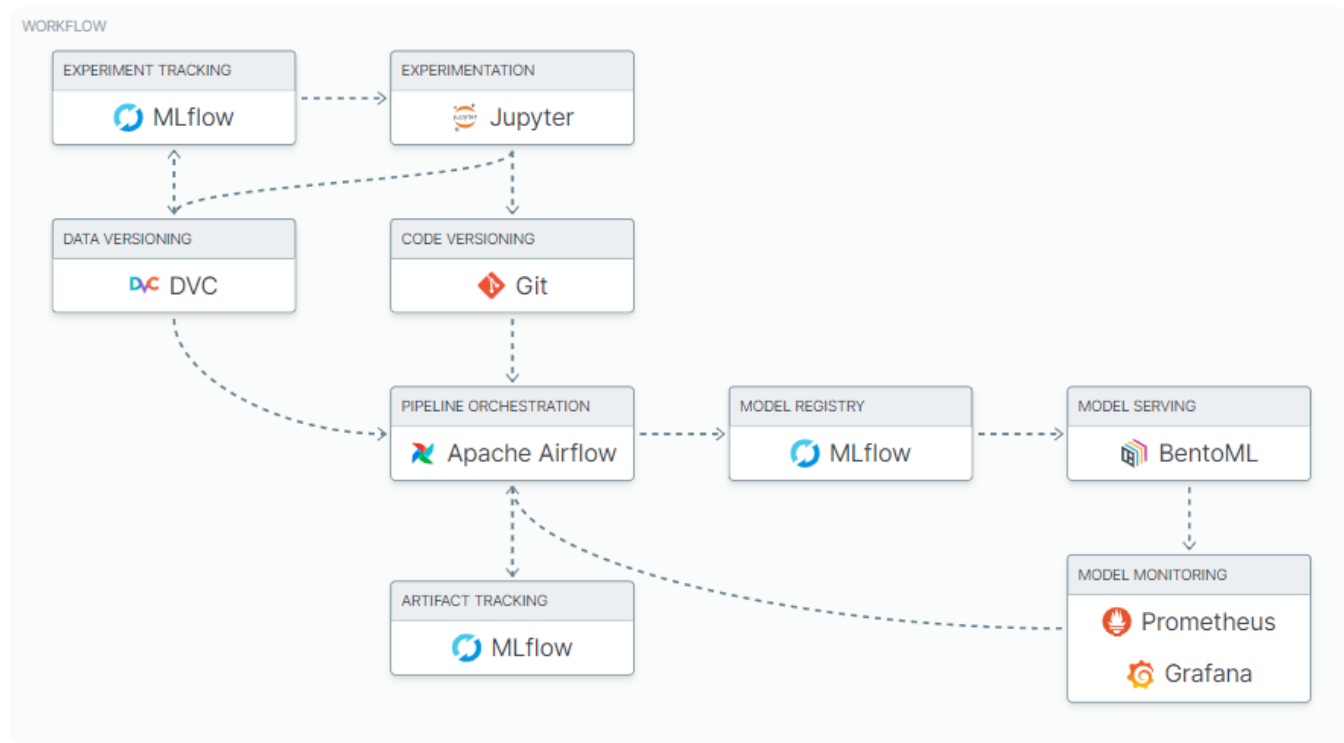


(C,D) Experimentation & Automated ML Workflow Pipeline












Example stacks



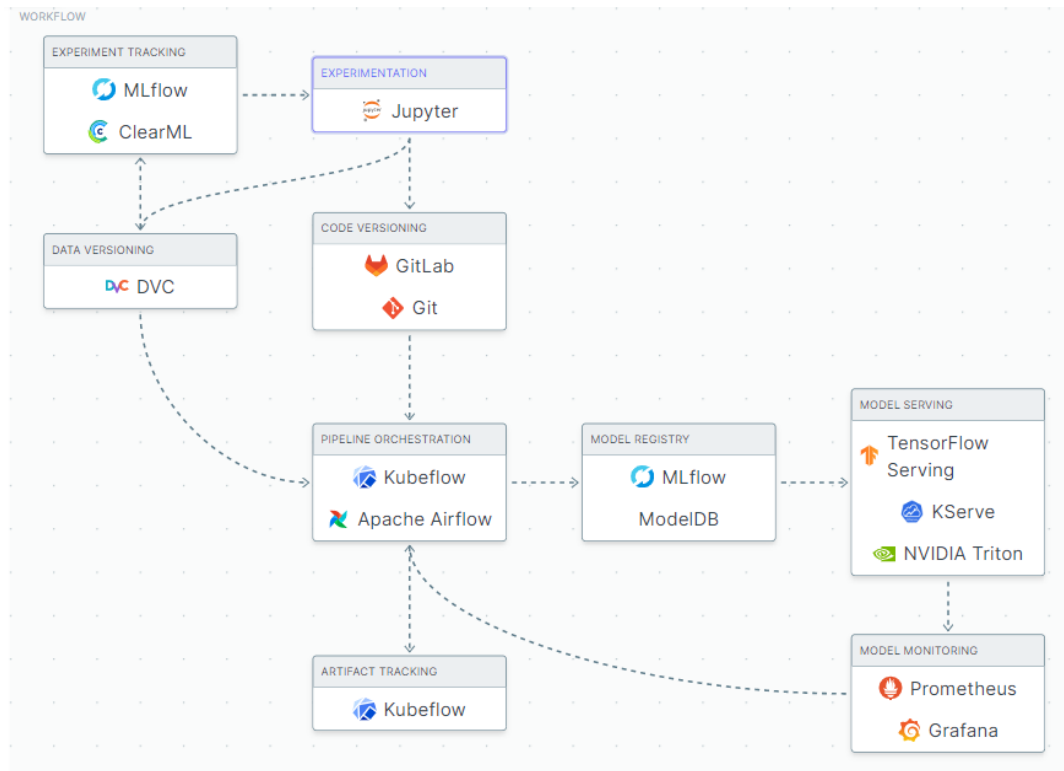
Example stacks

Tools	Function	Developer	License
IBM Watson ML	Deploying model as API	IBM	Proprietary
IBM Watson OpenScale	Monitoring Model in production	IBM	Proprietary
DVC	Data and Model Versioning	Iterative	Apache License 2.0
CML	Pipeline Automation	Iterative	Apache License 2.0
Terraform	Setups IBM infrastructure with script	HashiCorp	Mozilla Public License v2.0
Github	Code versioning	Github	Proprietary
Github Actions	CI/CD Automation	Github	Proprietary
Pytest	Python script testing	Pytest-dev	MIT
Pre-commit	Running tests on local commit	Pre-commit	MIT
Cookiecutter	Creating folder structure and files	Cookiecutter	BSD 3-Clause

Example stacks

MLOps Stage	Open-source Tool	Alternatives
Source Code	 Github	Bitbucket
Feature Store	 Feast	Hopsworks
ML Pipeline	 Kubeflow	Polyaxon
Model Validation Testing/Maintenance	 Deepchecks	Etiq AI, Great Expectations
Model Registry	 MLflow	Neptune
Model Serving	 Cortex	Seldon Core
Model Monitoring	 Deepchecks	Prometheus, Grafana

Example stacks



Example stacks

- Gathering and analysis data: Apache nifi
- End-to-End Platform for Continuous ML: ClearML (Alegro)
- Monitoring: Prometheus, Grafana, ELK Stack

The background of the slide is a light gray network pattern. It consists of numerous small circles, some of which are solid gray and others are hollow with a gray outline. These circles are interconnected by thin, light gray lines, creating a complex, web-like structure that fills the entire background.

Thanks!

Any questions?