MLOPS

By: Ashish Pal

MODEL DEPLOYMENT

When designing an ML system architecture, the following should be borne in mind.

- **Modularity**: the code used in preprocessing/feature engineering should be arranged in comprehensive pipelines.

- **Reproducibility**: the output of each component must be replicable for any version in time.

- **Scalability**: the model must be serveable to a large number of customers with minimal response time.

- **Extensibility**: it should be easy to modify for future tasks.

- **Testing**: the ability to test variation between model versions.

- **Automation**: eliminating manual steps wherever possible to reduce error chances.

DEPLOYMENT PIPELINES

Deployment Type	Training Approach	Prediction Approach - Batch	Prediction Approach - Live
On-Premises	Training is done locally on dedicated hardware or servers.	Batch predictions are run periodically on the same infrastructure.	Live predictions are served from the on-premises infrastructure.
Cloud	Training is performed on cloud-based resources.	Batch predictions can be scheduled using cloud services (e.g., AWS Batch).	Live predictions are served via cloud-based APIs or services (e.g., AWS Lambda).
Hybrid	A combination of local and cloud training resources.	Batch predictions can use cloud services for scalability (e.g., Azure Data Factory).	Live predictions can be a mix of on-premises and cloud APIs.
Edge	Training can occur on edge devices or remote servers.	Batch predictions might not be common due to limited resources.	Live predictions are executed directly on edge devices with optimized models.
Serverless	Training might involve serverless functions for specific tasks.	Batch predictions can be triggered by events using serverless functions.	Live predictions are served through serverless functions or APIs.
Containerized	Training is done within Docker containers for consistency.	Batch predictions can be executed within containerized environments.	Live predictions are served using containerized applications or microservices.

Model as a Service

- Model-as-Service is a common pattern for wrapping an ML model as an independent service. We can wrap the ML model and the interpreter within a dedicated web service that applications can request through a REST API or consume as a gRPC service.
- This pattern can be used for various ML workflows, such as Forecast, Web Service, Online Learning.
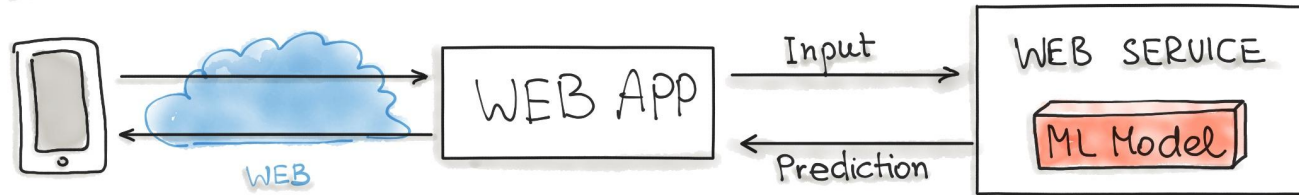
Model as a Dependency

Model-as-Dependency is probably the most straightforward way to package an ML model. A packaged ML model is considered as a dependency within the software application.
For example, the application consumes the ML model like a conventional jar dependency by invoking the prediction method and passing the values. The return value of such method execution is some prediction that is performed by the previously trained ML model.

The Model-as-Dependency approach is mostly used for implementing the Forecast pattern.
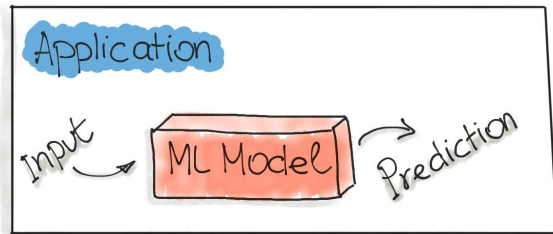
Precompute Serving Pattern

This type of ML model serving is tightly related to the Forecast ML workflow. With the Precompute serving pattern, we use an already trained ML model and precompute the predictions for the incoming batch of data.
The resulting predictions are persisted in the database. Therefore, for any input request, we query the database to get the prediction result.
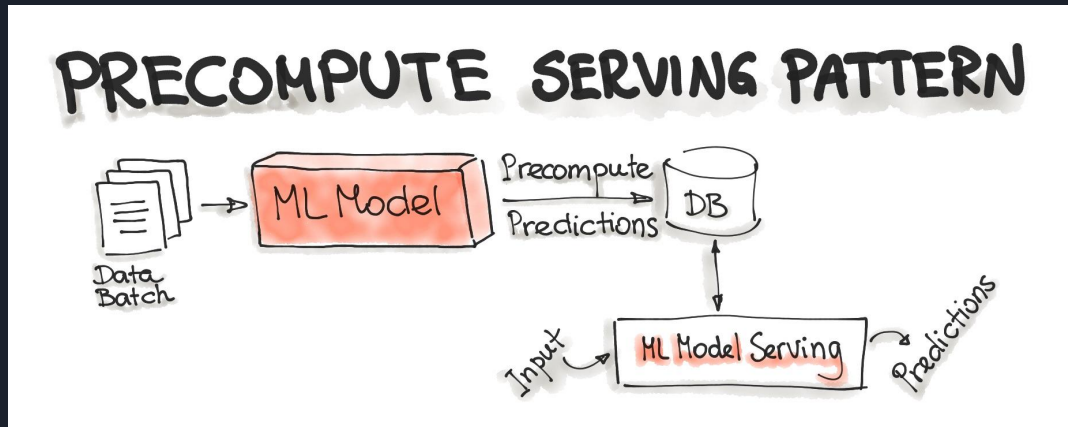
Deploying ML Models as Docker Containers

As of now, there is no standard, open solution to ML model deployment. As ML model inference being considered stateless, lightweight, and idempotent, containerization becomes the de-facto standard for delivery.

This means we deploy a container that wraps an ML model inference code. For on-premise, cloud, or hybrid deployments, Docker is considered to be de-facto standard containerization technology.

One ubiquitous way is to package the whole ML tech stack (dependencies) and the code for ML model prediction into a Docker container. Then Kubernetes or an alternative (e.g. AWS Fargate) does the orchestration.

The ML model functionality, such as prediction, is then available through a REST API (e.g. implemented as Flask application)
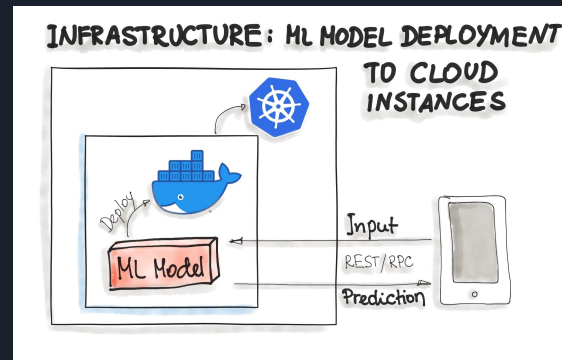
Deploying ML Models as Serverless Functions

Various cloud vendors already provide machine-learning platforms, and you can deploy your model with their services. Examples are Amazon AWS Sagemaker, Google Cloud AI Platform, Azure Machine Learning Studio, and IBM Watson Machine Learning, to name a few.

Commercial cloud services also provide containerization of ML models such as AWS Lambda and Google App Engine servlet host.

In order to deploy an ML model as a serverless function, the application code and dependencies are packaged into .zip files, with a single entry point function.

This function then could be managed by major cloud providers such as Azure Functions, AWS Lambda, or Google Cloud Functions.
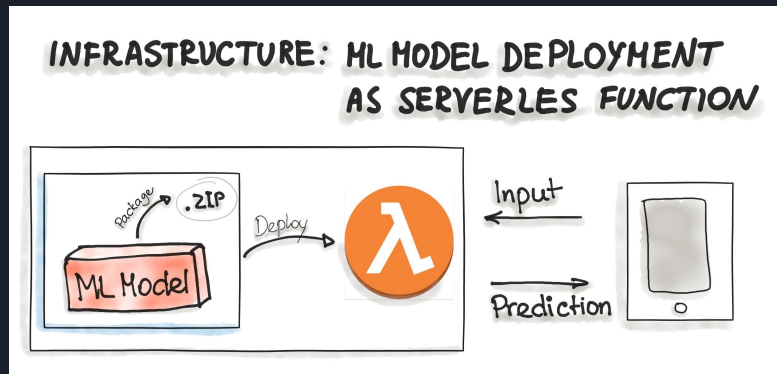
MODEL MONITORING
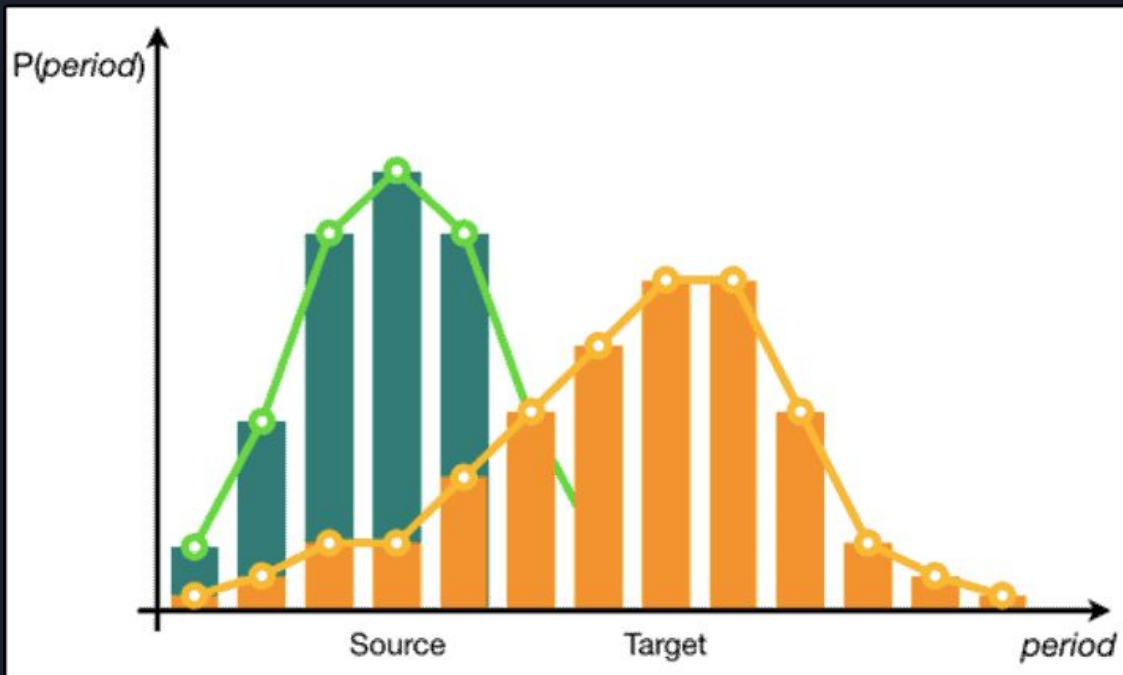
Data Drift

Data drift refers to the situation where the statistical properties of the incoming data used for model prediction change over time.

This change can be due to various factors such as shifts in the data distribution, changes in user behavior, or external events.

When data drift occurs, the data that the model encounters in production may differ significantly from the data it was trained on.
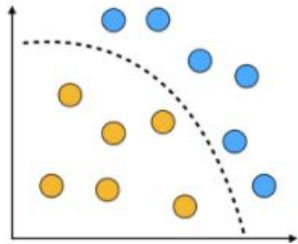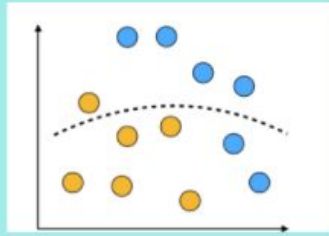
Concept Drift

Concept drift is closely related to data drift but focuses on the underlying relationships between features and labels rather than just statistical properties. It occurs when the relationships between input features and the target variable change over time.

This means that the assumptions the model was built upon are no longer valid. Concept drift can be more subtle and challenging to detect than data drift. Continuous monitoring is necessary to identify concept drift and to retrain or update the model as needed to maintain its

MLOPS PIPELINE

MLOps level 0: Manual process
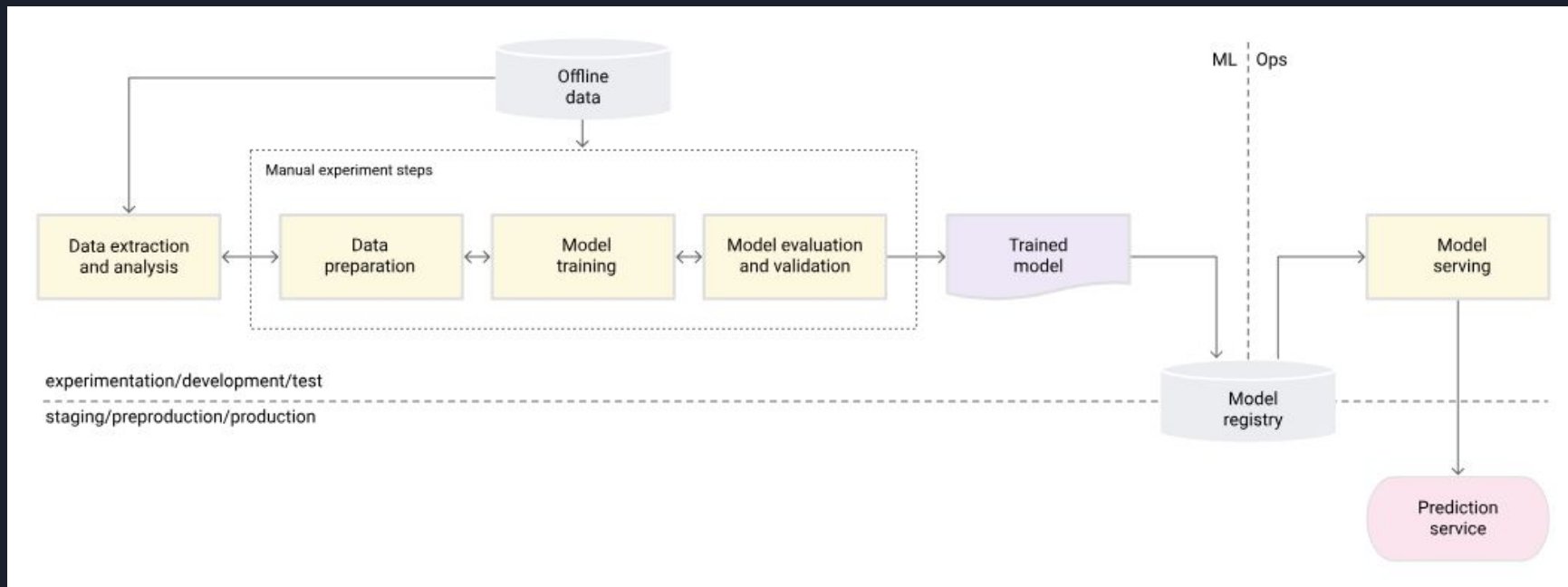
MLOps level 1: ML pipeline automation
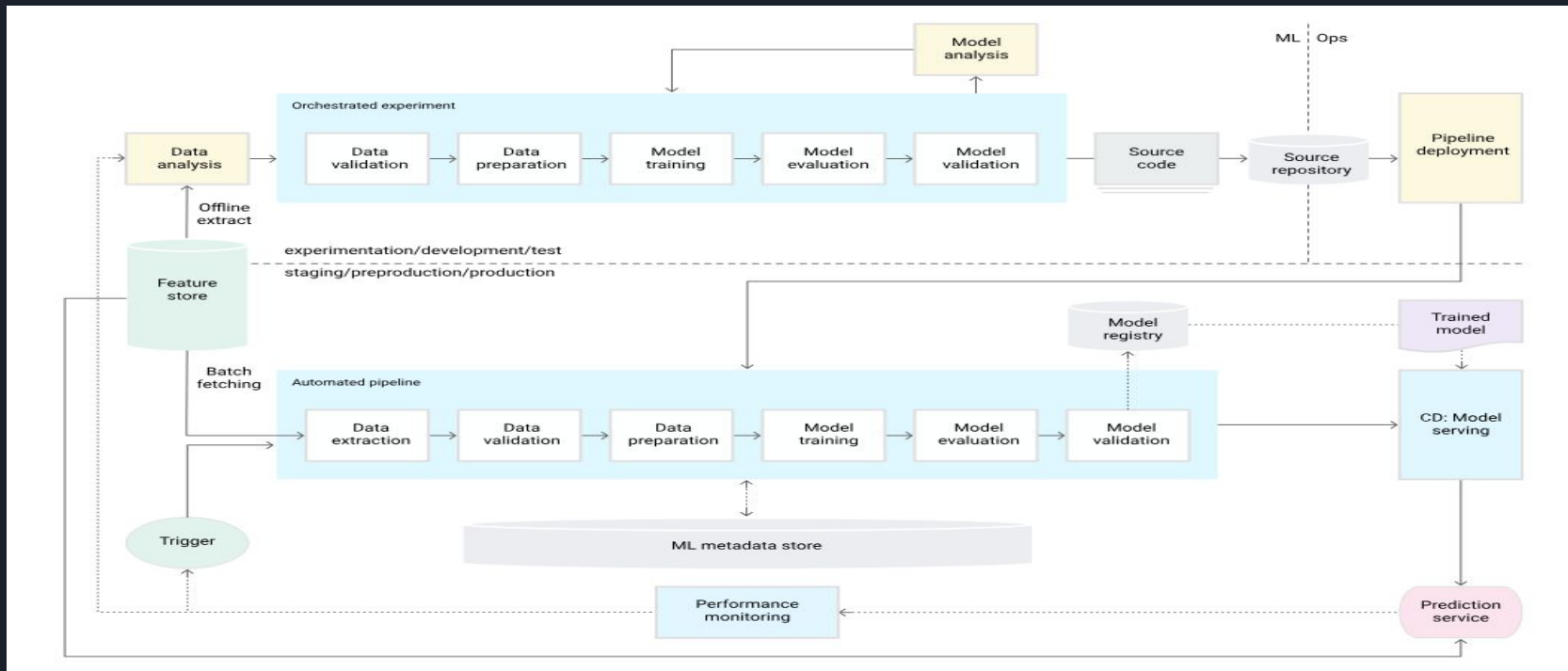
MLOps level 2: CI/CD pipeline automation

Google Cloud MLops

https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning
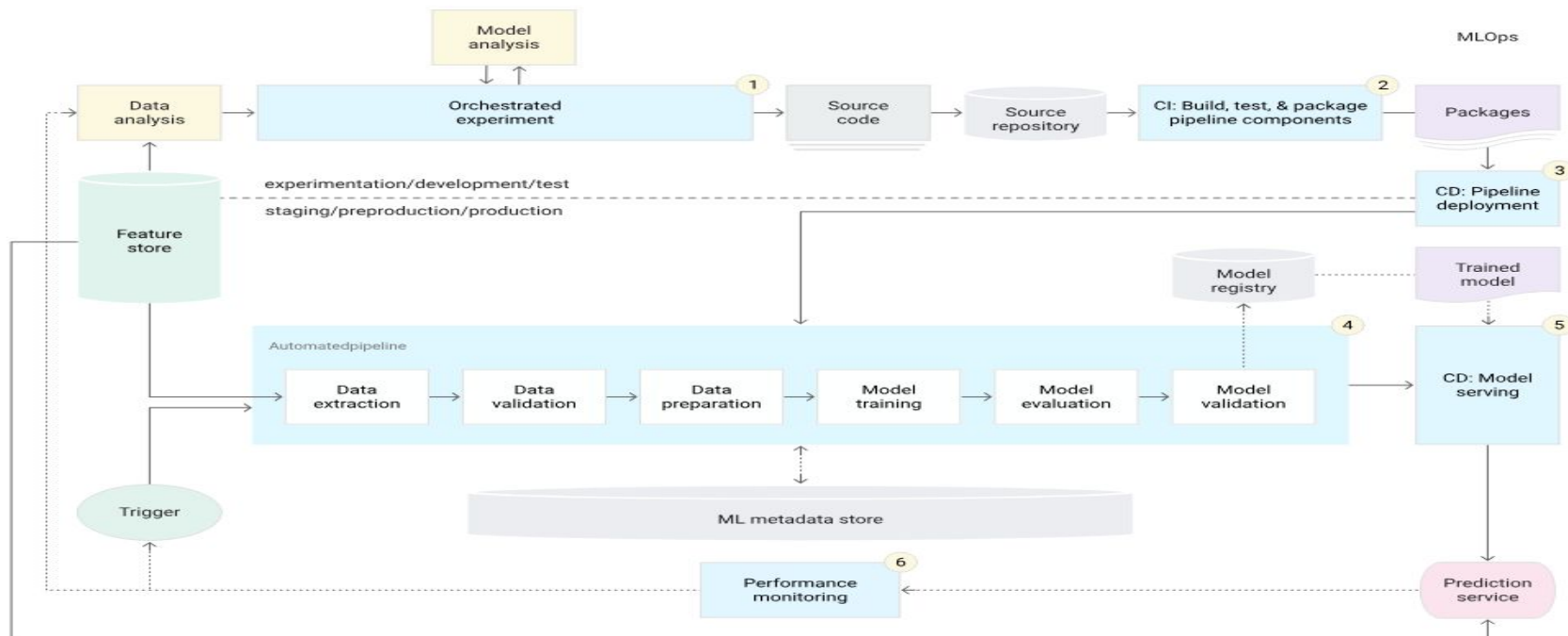
Machine Learning Operations (MLOps)

MLOps = ML + Dev + Ops
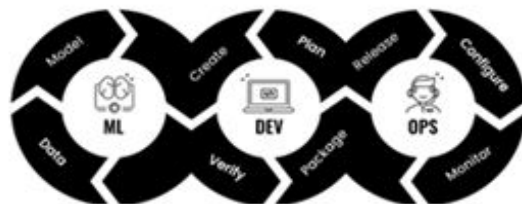
Line of Business

Business requirement

Machine Learning Lifecycle

Data Collection
- Collecting data from various data sources

Data Pipeline
- Data Pipeline Prep
- Data Clean up
- Data Transformation
- Data management Data drift monitoring
- Data support automation scripts/Tools

Model Building
- Model selection
- Model training
- Evaluation of Scoring
- Experimentation on accuracy
- Model validation

Model Deployment
- Scaling deployment model
- Model monitoring
- Model drift monitoring
- Model validation
- Assess the streaming and batch process

DevOps
- CI/CD pipeline
- Testing model
- Monitoring
  - Logging
  - Notification
  - Scaling deployment model
- Versioning
  - Model
  - Code
  - Data
- Model drift monitoring

Catalogs
- Dataset
- Feature
- Model

Governance
- Access control
- Security
- GDPR/CCPA

SMEs
Data Scientists

Data Engineer

ML Engineer

ML Engineer

MLOps Engineers

ML Archives

ML Governance

ML Architect

A MLOps Project Initiation

Business problem analysis (define goal) → designs architecture and technologies to be used → derive ML problem from business goal (e.g., classification, regression) → Understand required data to solve problem (data available?, where is it?, how to get it?) AND → Connect to raw data for initial data analysis (distribution analysis, data quality checks, validation checks)

MLOps Project Initiation Zone

B1 Requirements for feature engineering pipeline

define transformation & cleaning rules → define feature engineering rules

Feedback Loop feature requirements (iterative)

data pipeline code

transformation rules

feature engineering rules

orchestration component

Data Sources

streaming data

batch data

cloud storage

(labeled data)

B2 Feature Engineering Pipeline

Connect to raw data → data extraction → data transformation & cleaning → feature engineering (e.g., calc. of new features) → data Ingestion job (batch or streaming) AND

data preprocessing

data processing computation infrastructure

artifact store

CI/CD component

Data Engineering Zone

C Experimentation

model engineering (best algorithm selection, hyperparameter tuning)

data analysis → data preparation & validation → model training → model validation → export model

model training code

ML model

ML workflow pipeline code

model serving code

Repository

versioned feature data

ML metadata store

model training computation infrastructure

ML Experimentation Zone

versioned artifacts: **model** + ML training & workflow code

CI / CD component

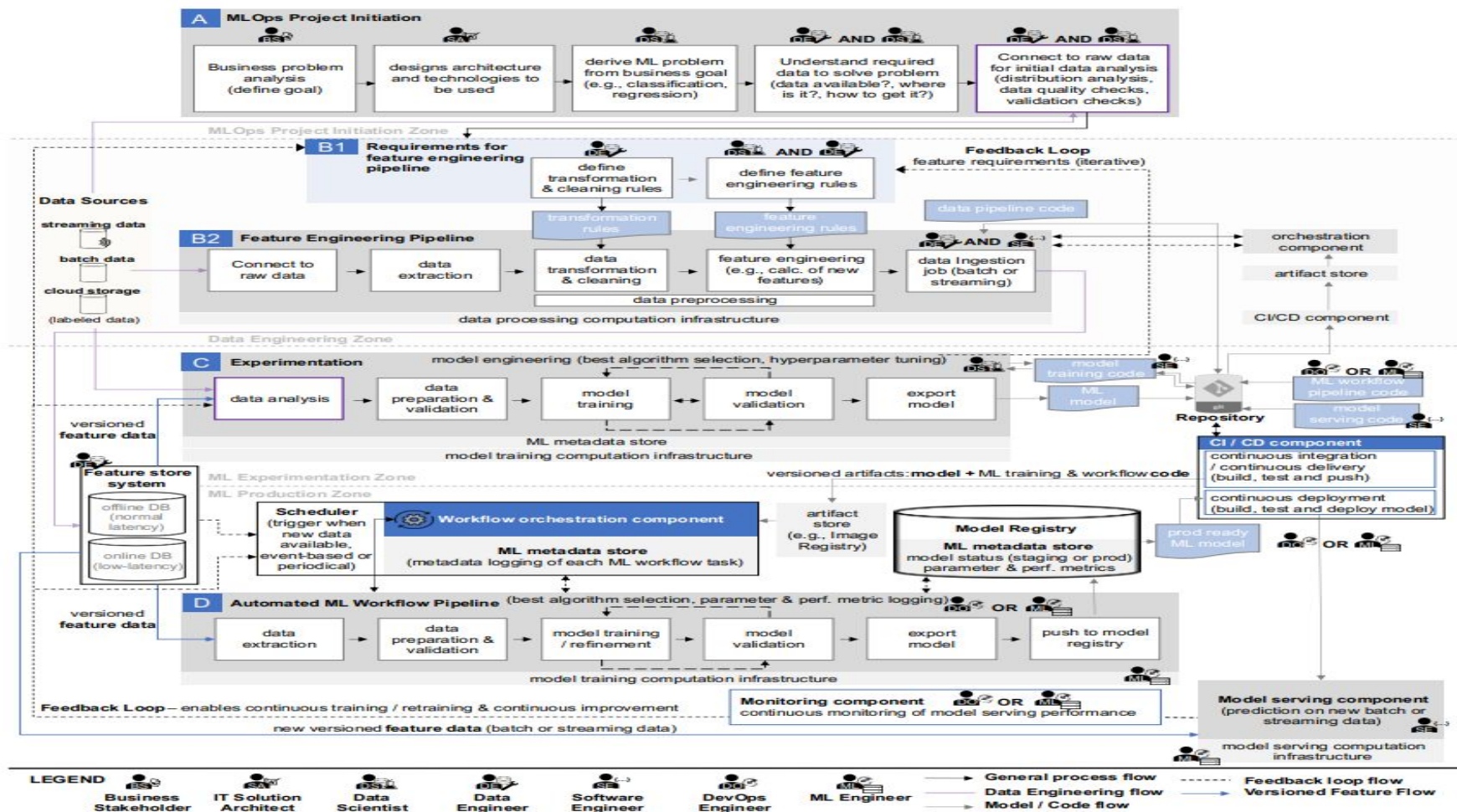continuous integration / continuous delivery (build, test and push)

continuous deployment (build, test and deploy model)

ML Production Zone

Feature store system

offline DB (normal latency)

online DB (low-latency)

Scheduler (trigger when new data available, event-based or periodical) → **Workflow orchestration component**

ML metadata store (metadata logging of each ML workflow task)

artifact store (e.g., Image Registry)

Model Registry

ML metadata store model status (staging or prod) parameter & perf. metrics

prod ready ML model

OR

D Automated ML Workflow Pipeline (best algorithm selection, parameter & perf. metric logging)

versioned feature data

data extraction → data preparation & validation → model training / refinement → model validation → export model → push to model registry

model training computation infrastructure

OR

Feedback Loop – enables continuous training / retraining & continuous improvement

Monitoring component OR continuous monitoring of model serving performance

Model serving component (prediction on new batch or streaming data)

new versioned **feature data** (batch or streaming data)

model serving computation infrastructure

LEGEND

Business Stakeholder | IT Solution Architect | Data Scientist | Data Engineer | Software Engineer | DevOps Engineer | ML Engineer

General process flow
Data Engineering flow
Model / Code flow
Feedback loop flow
Versioned Feature Flow
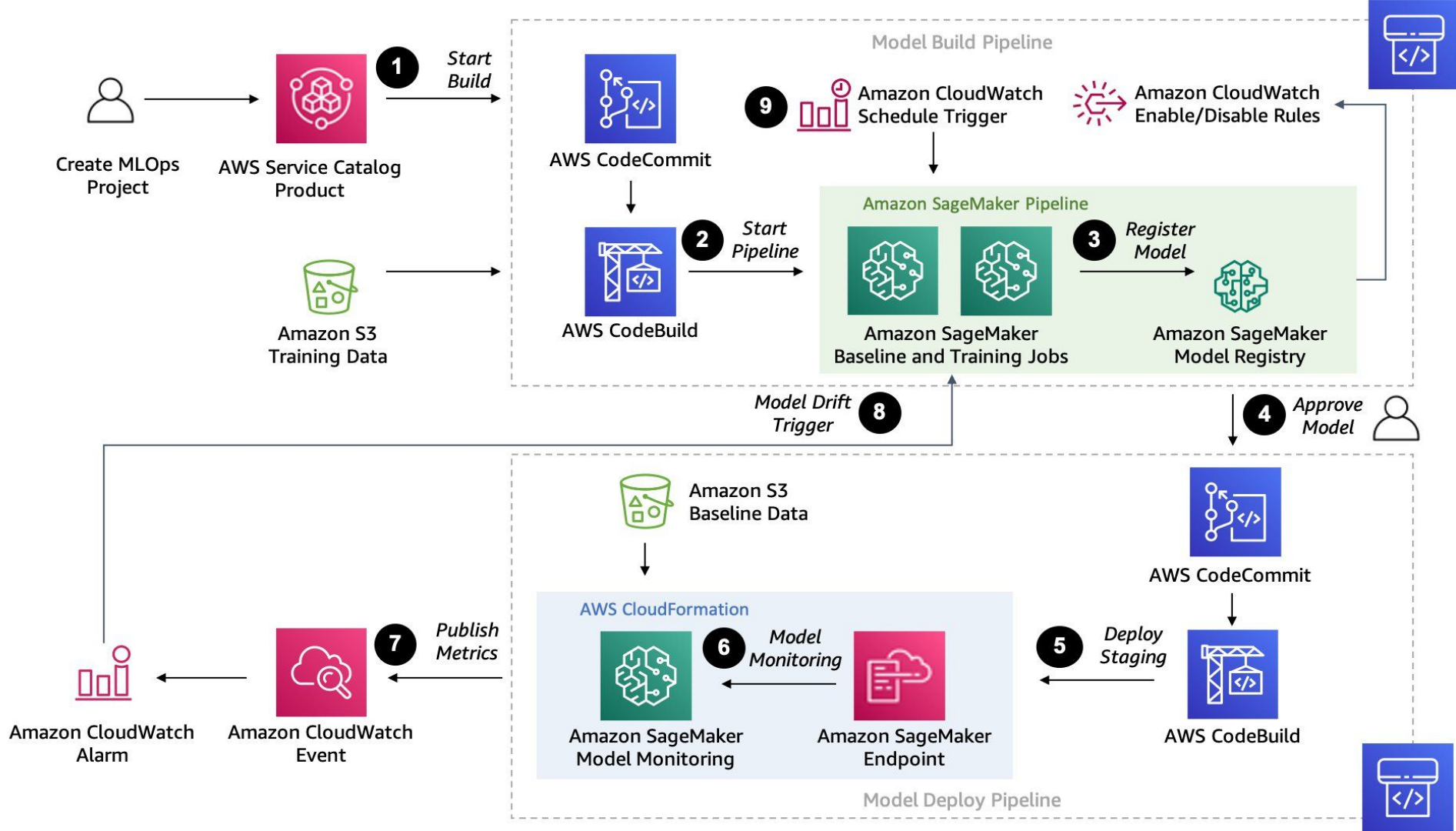
Azure Storage

Triggers
- Data-driven
- Schedule-driven
- Metrics-driven

(New data location)

Azure Machine Learning pipeline endpoint

Azure DevOps build pipeline

Data sanity test　Unit test

Runs for new code

Azure Machine Learning

Create machine learning workspace　Create machine learning compute　Publish machine learning pipeline

One-time run

Publish machine learning pipeline

Azure Machine Learning retraining pipeline

Train model

Evaluate model

Register model

Azure Machine Learning Compute

Operationalize model

Model Artifact Trigger

AML Model Management

Azure DevOps release pipeline

Azure Machine Learning

Package model into image　Deploy model as web service on ACI　Test web service

Staging, QA

Manual release gate

Azure Machine Learning

Deploy model as web service on AKS　Test web service

Production

Scoring services

Monitoring

Application Insights

Container Instances

Kubernetes Service

Model performance data

Microsoft Azure

Google Cloud Platform

Intelligent Products Essentials

Data warehouse
BigQuery

Raw storage
Cloud Storage

Data scientist

Machine Learning Engineer

Continuous integration
Cloud Build

ML pipeline config
Kubeflow Pipeline

Continuous deployment
Cloud Build

ML pipeline
Vertex AI Pipeline

Triggers
Pub/Sub

Feature store
Vertex AI Feature Store

ML pipeline (detailed)

Data extraction

Data validation

Data preparation

Model training

Model evaluation

Model validation

Model registry
Vertex AI

ML metadata repository
Vertex ML Metadata

For batch prediction

Trained model
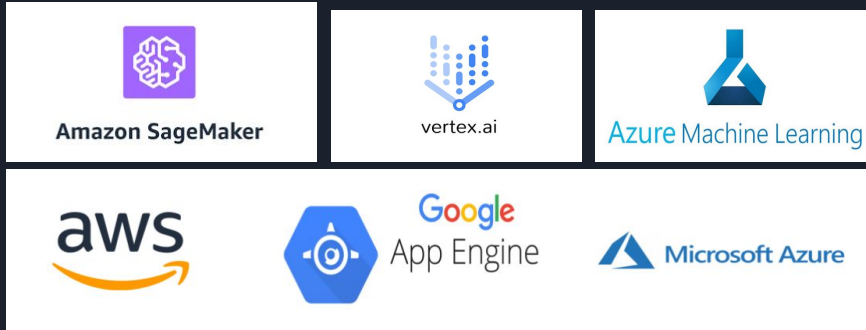Vertex AI

Batch prediction
Vertex AI

Version Control

CI/CD

Containerization and Orchestration

Model Deployment

Monitoring and Observability

www.pathtoai.com

Thank You

ASHISH PAL

PATH TOAI

SUBSCRIBE

www.pathtoai.com