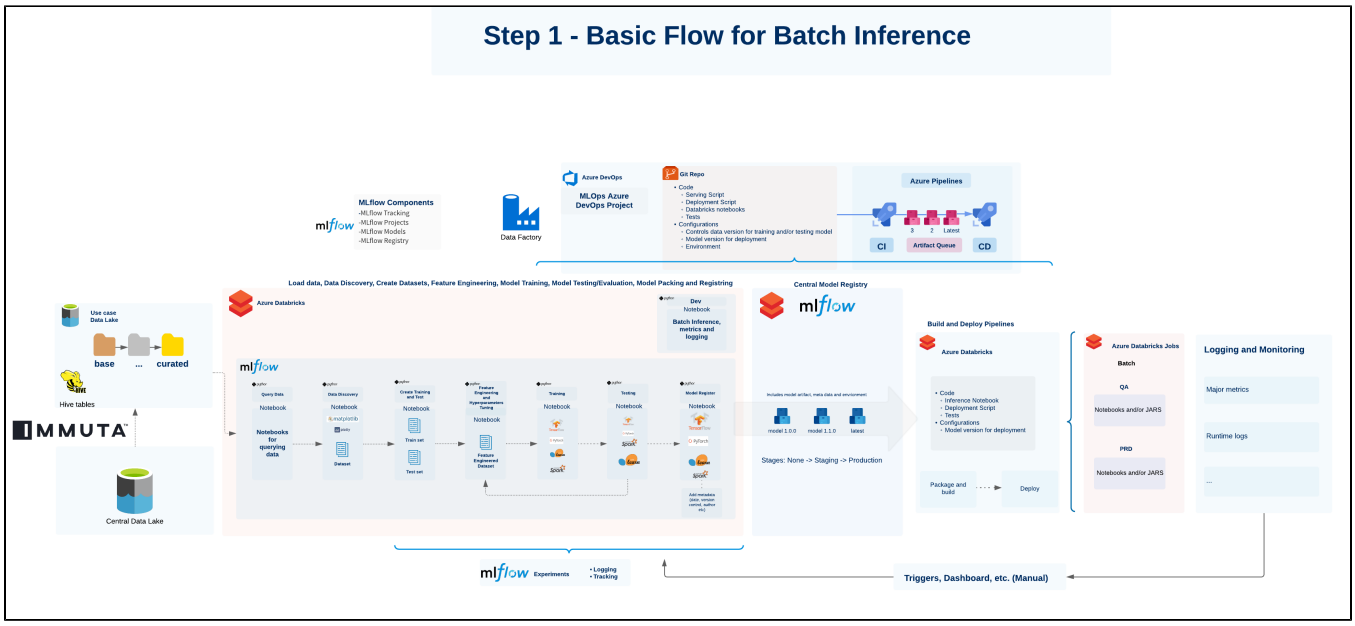
MLOps Two-Steps Implementation - Basic to Enhanced

The content of this page is the conclusion of the MLOps design meeting on 2021-06-11, which is based on the existing design, new meetings and discussion with the team. The intent is to NOT break anything that has been done, and NOT to put a lot of effort on additional exploring and developing, while improving the decided architecture with a more simplistic flow.

The implementation of the MLOps can be divided into 2 steps.

- Step 1 focuses on batch model deployment, and the MLOps workflow will reach the Level 3 - Automated Model Deployment.
- Step 2 adds on the workflow from the previous step, enhancements including (can be discussed) Delta Lake, Feature Store, real-time model deployment via AzureML to Kubernetes, better logging and monitoring. The ambition for the 2nd step is to enable Continuous Training and Continuous Monitoring, to reach the Level 5 - Automated Operations.

Step 1 - Batch Inference



Major parts of the workflow

- The process of EDA, data preprocessing, model training/evaluation will be using Azure Databricks.
- The experiments will be tracked by Databricks managed MLflow
- A separate Databricks workspace will be using as a central MLflow model registry.
- A model will be deployed with a Databricks Notebook for batch inference in QA/Prod env.
- Major metrics and runtime logs will be collected in QA/Prod env (this to be discussed further and is concerned in separate pages).
- DevOps Pipelines will cover sanity tests/unit tests, model training, model deployment.
- In QA/Prod environments, Databricks job scheduling via ADF
- Using the new Databricks repo feature which is already implemented as of the current implementation

Topics that are not covered in this steps are

- Real-time model deployment via AzureML and Kubernetes
- Customer container for Databricks

MLOps Workflow (Differences from Current Design)

Development

- A batch inference notebook is developed before model deployment.

CI Pipeline

- No changes.

Model Training

- The model is registered to a central Databricks managed MLflow model registry. (The current design mentions uploading the model artifact to JFrog).

Model Deployment / Release Pipeline

- Should the batch inference notebook be tested before QA?

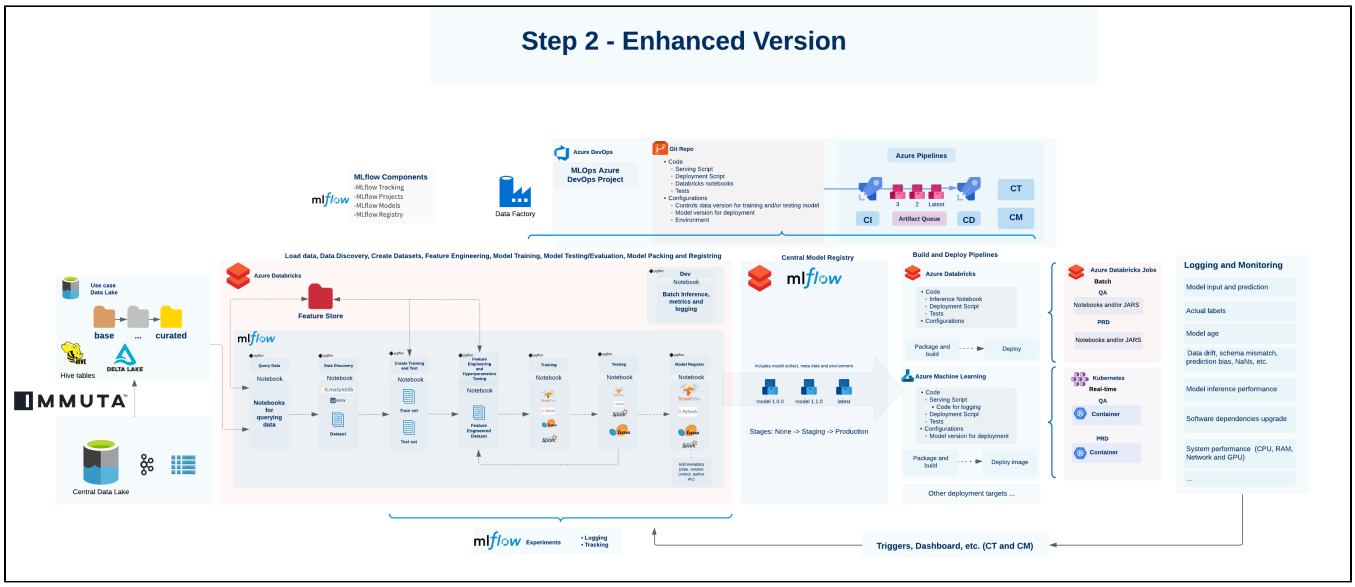
Model Re-Training

- Model retraining is triggered manually.

Model Monitoring

- Simple logging and metrics in PROD batch inference
  - only major metrics, stats of training data and prod input/prediction data (data drift), ...
  - options to be discussed and decided
    - mlflow
    - application insights
    - open source pkg
    - custom code
- Data scientists monitor the metrics (including self-calculated stats of data for data drift monitoring), and trigger re-training based on the metrics and the amount of new data.

Step 2 - Enhanced Solution



Building on top of step 1, enhanced solution introduced new components

- Introduction of Delta Lake to improve query performance with larger data and provide versioning of the data
- Utilizing Databricks Feature Store to ease sharing features across models
- Introduction of Azure ML with its powerful SDK and Kurbernetes for web service / real-time inference
- Improved loggings of metrics and monitoring
- Continuous training (CT) and continuous monitoring (CM)