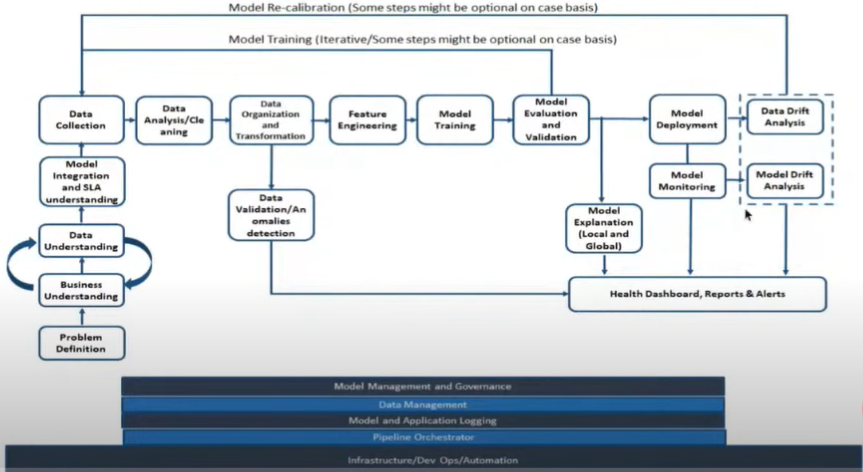
MLOps

# **An Introduction to MLOps**

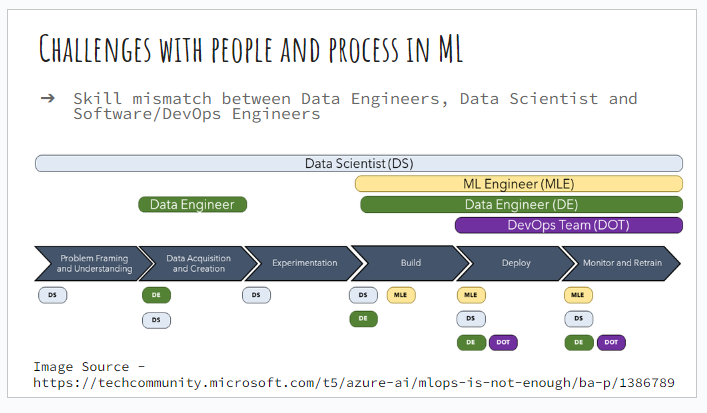
* In real world ML systems there are multiple components, ML code is just a small part of entire ecosystem.
* **The ML Lifecycle contains following steps like:**



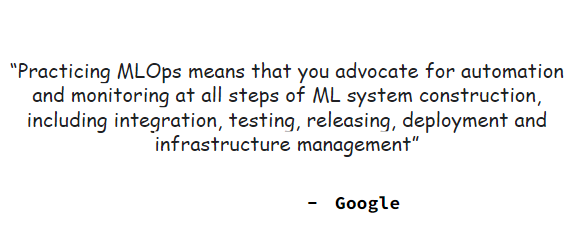
* We have to track data as well as the model.
* **Challenges faced during ML development**
* Development, training and deployment environment can be different
* Tools, libraries, and dependencies can complicate model deployment
* Tracking and analysing experiment can become tedious to handle
* Difficult to reproduce experiment as input data changes
* ML code end up in a spaghetti jungle
* **Challenges as ML in production**
* Live data is not equal to training data
* Feature engineering pipeline must match between training and serving infrastructure

(There must be model tracking pipeline)

* Seamlessly scale up and scale down deployed model
* Continuous training and champion challenger model deployment
* Different technology landscape between development and deployment
* **Challenges with people and process with ML**
* Skill mismatch between Data engineers, Data Scientists, and software/Devops Engineers



* **What is MLOps?**
* MLOps is not about throwing a product and everything is fine.
* It’s a process change in an organization.
* How to develop, package the model
* MLOps in simple terms is DevOps for Machine Learning
* MLOps enables data science (data engineers and ML engineers ) and IT teams ( software engineers) to collaborate and increase pace at which ML models can be developed, deployed ,scaled, monitored and re-trained.

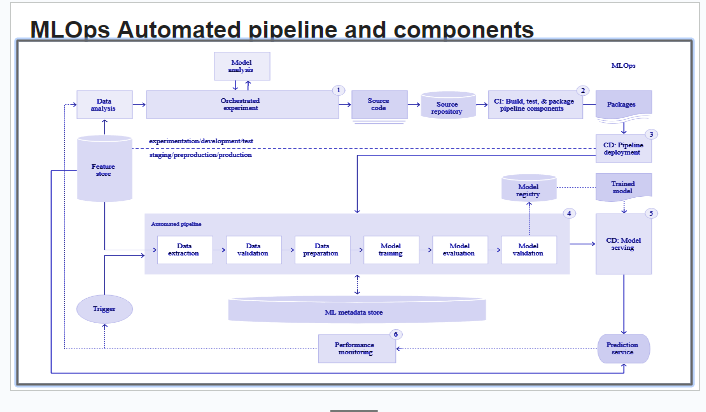


* **How’s MLOps different than Devops?**
* Data/ schema versioning apart from code versioning
* All the Artifacts must be added apart from your code
* Experimentation tracking (Model hyper parameters, Data distribution, Model performance, feature importance etc.)
* Model Artifacts versioning
* Monitor continuously for data and model drift
* Continuous re-training of model
* Capture sensitivity of features to target
* Continuous training is speciality of MLOps
* **ML + DevOps:**
* **CI**

Continuous Integration (CI) – Build, Test and Validate code + data + schema + models

* We need to create unit testing scripts, data scripts
* **CD**
* Continuous deployment (CD) – in continuous deployment we are training our ML pipeline + serving Component + deploy the model
* **CT**
* Continuous training – automatically re-train and serve the model
* **MLOps Architecture:**

MLOps automated pipelines and components:



* Data engineers collect the data from different sources and go through data quality checks, data cleaning, aggregation of the data and store it in feature store
* Data aggregation, data transformation etc. in feature store, new defined feature will be added to feature store and all these features must be searchable and sharable.
* Feature store – no duplicate features must be there, governed and controlled, generate customer lifetime value.
* We use features for training and deployment, technical mapping comes into picture, it should support both training and deployment pipeline. Non-functional requirements must also be satisfied.
* Where the feature store is kept is very important based on the business requirement.
* ML metadata store – tracks your experiments, it also stores the computed statistic of the data, model and data version details. We are just tracking the metadata.
* We have development test pipeline and production pipeline, what we do is we take the features, do data analysis, test data, analysis and performance the model, we check the source code and source repository it can be anything like GitHub, cloud tec.
* We need to have CI pipeline from source repository so to understand we have done in the testing environment.
* How do you want to create the infrastructure, how to deploy it and what are the dependencies? Continuous integration will package your code and move to continuous deployment (CD)
* CD reads your code, reads what you have created, reads your definition, once all this is done it will take data from the feature store and start implementing it in the production environment.
* Sometimes we don’t have access to production environment, so we create an automated pipeline, which will have Data Extraction, Data validation, Data preparation, Model training, Model evaluation, Model validation
* Everything cannot be automated.
* There must be folder structure defined, and pipelines for feature selection, pre-processing, training
* CD model------ Prediction service------Performance monitoring-----trigger



**Q. In continuous training (CT), how we define the feature & how it infer the previous results?**

**Ans:** we already have a set of features to develop the model. We retrain the data on the same features, same algorithm, we need to have a development cycle for redeveloping the model. Monitor performance of the model.

Q. What is CI? What do we do in CI stage?

Ans:

* We are packaging the code which has some dependencies
* We are not just creating the code but we need to create an environment to test the code
* Sample data might fit in pandas memory but when we go to production we might need a new ecosystem, DASK, pyspark etc.
* CI is taking your source code & packaging it to a format in pipeline deployment to spin up the infrastructure continuously for you depending on the environment.

Q. How to stitch all these pipelines stages together? Is Airflow widely used? Or other tools like MLFlow or Kubeflow?

Ans:

* Airflow is a very good tool to stitch your code framework as DAG file by creating operators.
* Kubeflow has some challenges, experimentation part becomes slow
* MLflow is good for project tracking
* Airflow gives a good orchestration & customization
* We can create custom/ partial deployment pipeline or data pipeline
* Model deployment can be separate pipeline
* When we try to stitch everything we might get lost
* We can have event based mechanism, we can have kafka queue and the trigger.