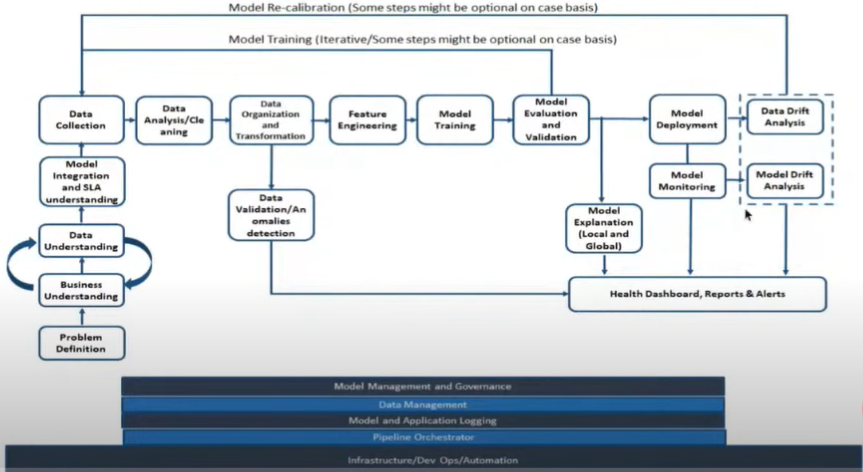
MLOps

# **An Introduction to MLOps**

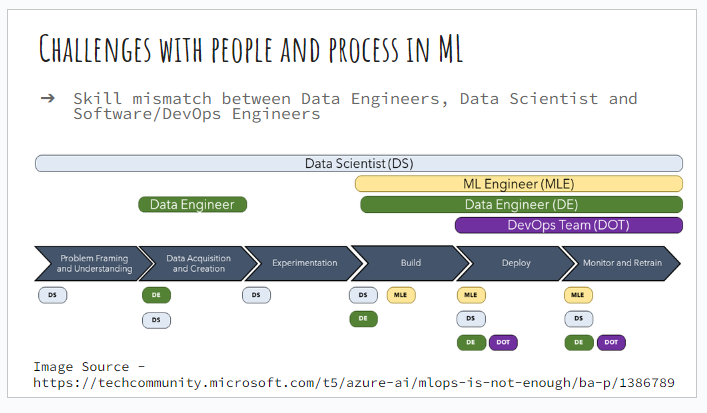
* In real world ML systems there are multiple components, ML code is just a small part of entire ecosystem.
* **The ML Lifecycle contains following steps like:**



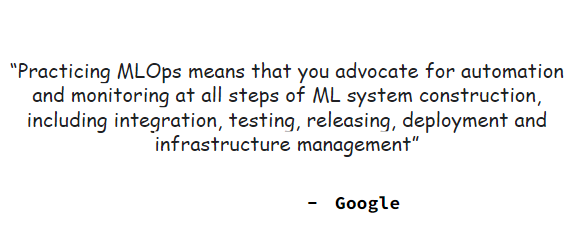
* We have to track data as well as the model.
* **Challenges faced during ML development**
* Development, training and deployment environment can be different
* Tools, libraries, and dependencies can complicate model deployment
* Tracking and analysing experiment can become tedious to handle
* Difficult 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.

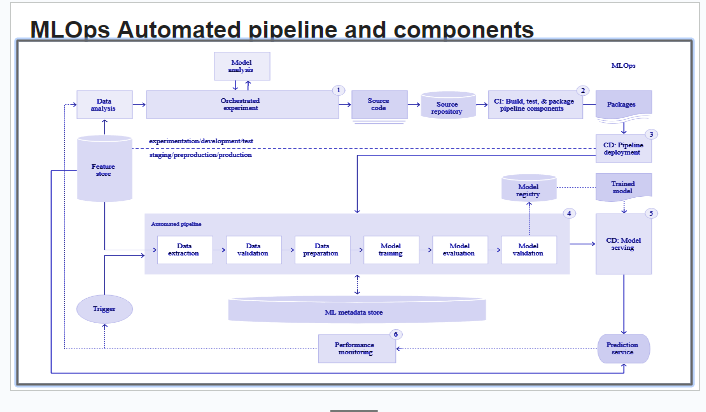


* **How’s MLOps different than Devops?**
* Data/ schema versioning apart from code versioning
* All the Artifacts must be added apart from your code
* Experimentation tracking (Model hyper parameters, Data distribution, Model performance, feature importance etc.)
* Model Artifacts versioning
* Monitor 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.

Q. How can we measure data drift for NLP text?

* It is similar to ML model process.
* We use tf-idf vector to know which features are being selected.
* NLP data drift is complex
* Find contribution and similarity vectors.

Q. Data storing

* We want to freeze the data, so in the initial stages no need to store the data.
* Make data read only.

Q. How do we retrain the model since we don’t have ground truth for production data?

* Retraining is only on the ground truth.
* Retraining is not based on the prediction
* We train the model till we get the ground truth.
* Never retrain on the production ground truth.

Q. The trigger scenario would usually be on streaming or interval basis?

* It can be either way. We can use event based architecture, as soon as the model is drifted we can put event in layer & the streaming layer can trigger the pipeline.

Q. How feature store is being deployed in the production?

* We have data elements, SOR, we can define use case specificity layer. In development we cannot have all the data. We can use production feature store into development feature store.

Q. How do we create explainability package?

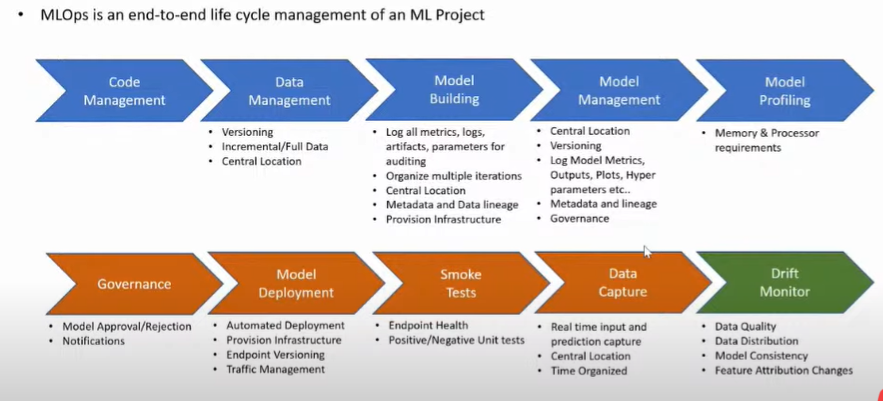
* When we develop the model, we know what features are contributing to the network, we also create our framework for doing sensitivity analysis.

# **MLOps with Azure**

Here we are going to understand what is MLOps? What are the capabilities of MLOps in Azure?

How to do fully automated training & deployment pipeline on azure?

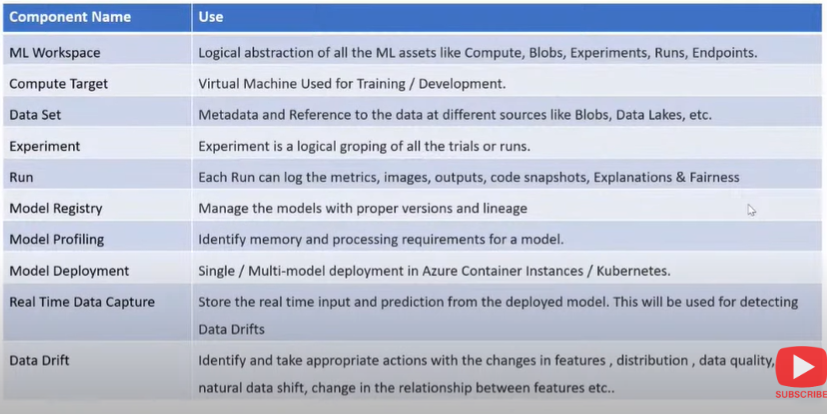
* **MLOps (ML operations)**



* MLOps is an end-to-end lifecycle management of an ML project.
* It is difficult to manage code management, data management, model building in local directory.
* We can manage the infrastructure required to train the model in MLOps.
* Model management
* Model profiling will have- Memory & processor requirements.
* The tasks code management>>>>>>>Data Management >>>>>>>> Model building >>>>>>>Model Management >>>>>> Model profiling

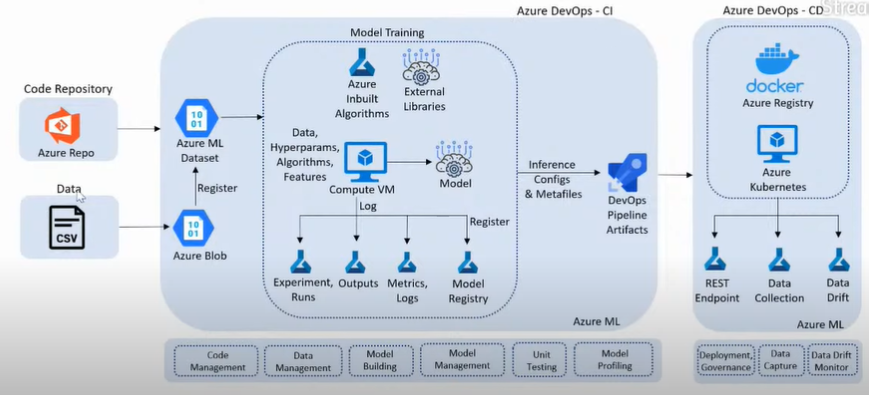
Talks about MLOps on the continuous integration side. (CI phase)

* The next phase would be deployment phase.
* Traffic management is not required all the time. If multiple versions of the model are not present, then it is not required.
* Data drift monitor is post deployment activity.
* We check model performance.
* ML workspace- logical abstraction of all the ML assets like compute, Blobs, Experiments, Runs, Endpoints.
* Compute target- virtual machine used for training/ deployment. We can refer it in our CI/CD pipeline.
* Data Set – Abstract API/ Metadata & reference to the data at different sources like Blobs, Data lakes etc.
* By using Data set we can manage all the connectivity, & reference to external data sources.
* Azure ML Components:



* Dataset will help to bring the data without understanding on external sources.
* Experiment is nothing but logical grouping of all the trials and runs.
* One iteration is known as Run, when we organize multiple runs together we call it as experiment.
* Model registry – manage the models with proper versions & lineage. This is the last stage of CI pipeline.
* Model profiling is used to identify memory and processing requirements for a model before going to production.
* Model deployment – single/multi model deployment in azure container instances/ kubernetes.
* Real time data capture: store the real time input & prediction from the deployed model. This will be used for data drifts.
* Data drift- Identify & take appropriate actions with the changes in features, distribution, data quality, natural data shift, changes in the relationship between features etc.

**Architecture used:**



* We get the data and keep it in azure blob storage, data source can be CSV file, data lake, blob storage, snaps or anything.
* Any data service supported by azure is supported here.
* From azure blob storage we create an Azure ML dataset & refer that dataset directly in the ML experiment.
* We don’t have any connectivity to azure blob storage directly. We create abstract of the dataset.
* We have to create compute VM in model training in which we can create/use Azure inbuilt algorithms, external libraries.
* Everything can be stored in Azure ML Workspace & can be retrieved.
* We have divided our pipeline into CI/CD.
* We have the model training, model training done we got the best model & that is stored in model registry. Then we store all that configuration which is required for the deployment pipeline in the Devops pipeline Artifacts.
* Once we have the CI pipeline we go with the CD pipeline.
* In CD pipeline we take Devops pipeline Artifacts which are stored in the CI pipeline & we deploy it into Azure kubernetes or Azure registry (container registry)
* Always go with the azure kubernetes, but we won’t be able to use it for free tier.
* CD pipeline will create a REST endpoint & enable the data collection & data drift post deployment part. (Data collection & Data drift are enabled in Azure kubernetes)
* Go to Azure Devops- create free account

1. Dev.azure.com – login to account
2. Create a project
3. Build a pipeline – create a pipeline – pipeline (here we are going to integrate the code)
4. Here we can get out code from any repository (we will be using Azure repos git for demo)
5. Azure repos git (note: not able to click on continue as there is no branch available to select)
6. Azure repos are a set of version control tools that you can use to manage your code

**Q. How to create azure pipeline?**

**Q. what is Azure pipelines?**

- azure pipelines automatically builds & tests code projects.

- it supports all major languages & projects types & combines continuous integration, continuous delivery, continuous testing to build, test & deliver your code to any destination. (CI, CD, & CT)

**Q. Build Azure Repos Git or TFS GIT repositories?**

Check how to do it?

Back to the demo practice:

* For demo purpose imported repository from my GitHub account in **Repos** section.
* Then in pipeline able to select continue.
* Create an empty job
* We can add tasks in that
* Do not save the changes & go to project setting:

Service connection – create service connection – azure resource manager

(having an issue with this check again)

