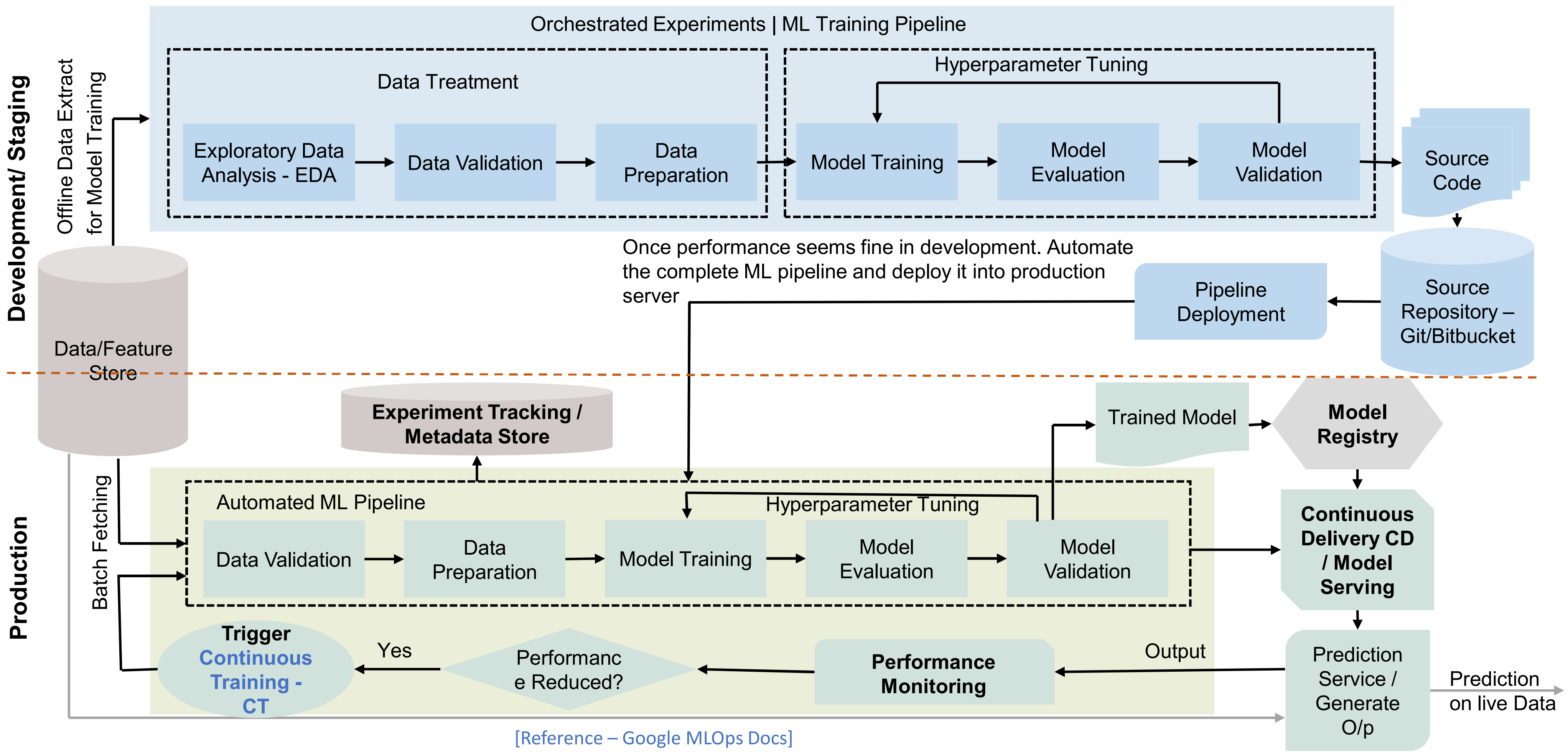
MLOps – Machine Learning Operations in Production

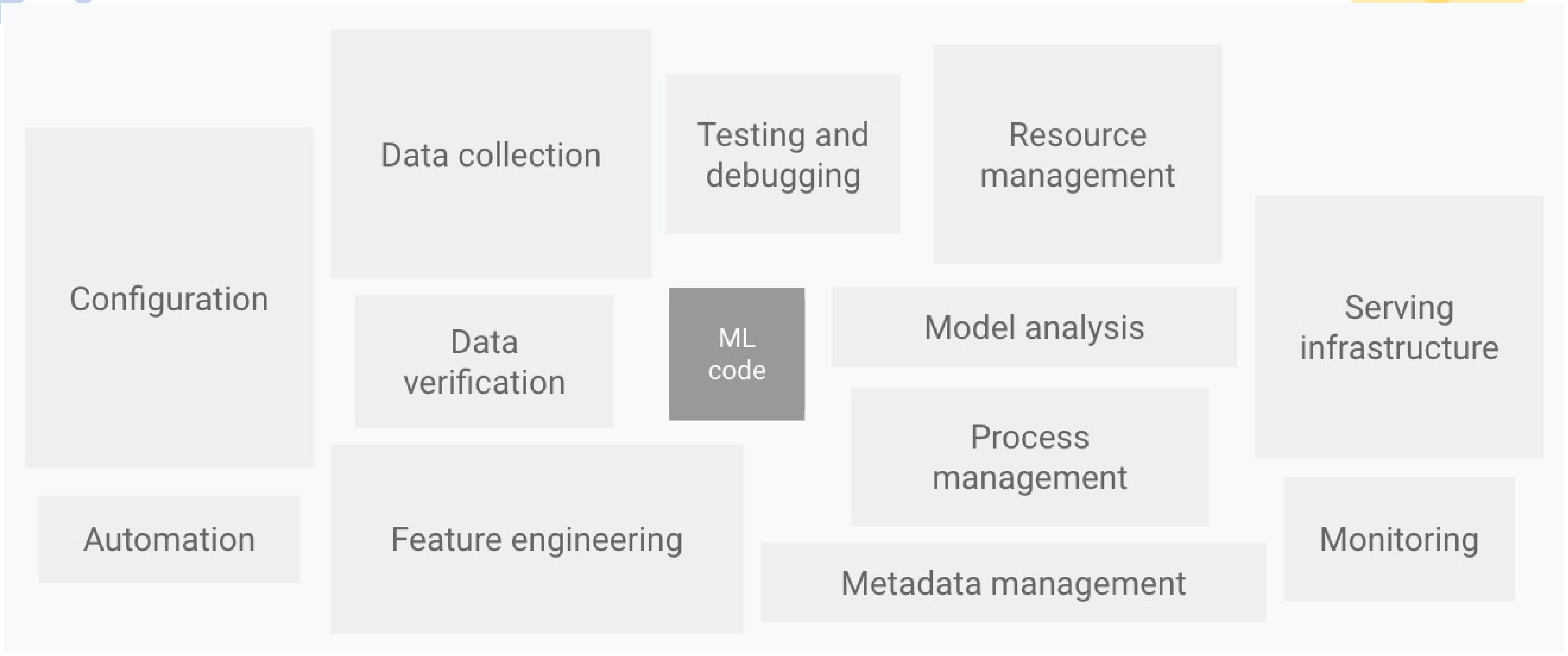
Data Science Duniya
https://www.ashutoshtripathi.com

MLOps – Automated Production Pipeline



[Reference – Google MLOps Docs]

How MLOps came into the picture!



Hidden Technical Debt in Machine Learning Systems

MLOps maturity levels

Google Model

Google has its own model of MLOps maturity levels. It appeared as one of the first models, is concise, and consists of three levels:

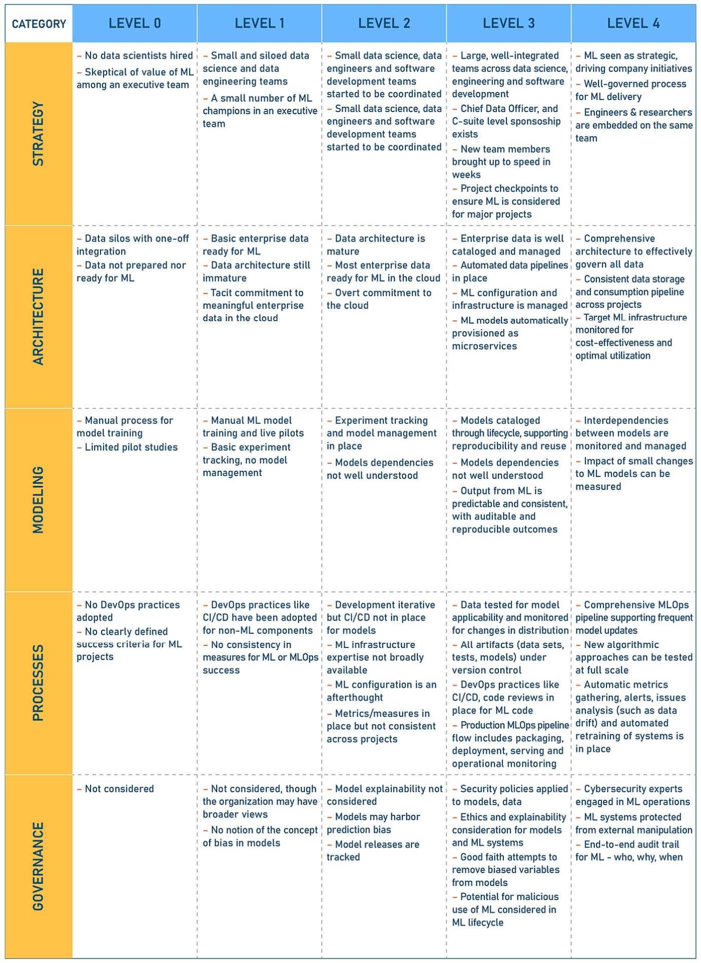
1. Level 0: Manual process
2. Level 1: ML pipeline automation
3. Level 2: CI/CD pipeline automation

First, do everything manually, then build an ML pipeline, and then automate MLOps.

Azure Model

Today, many large companies using ML have created their own maturity models. Azure also has a similar approach to identifying levels. However, they have more levels than Google:

1. Level 0: No MLOps
2. Level 1: DevOps but no MLOps
3. Level 2: Automated Training
4. Level 3: Automated Model Deployment
5. Level 4: Full MLOps Automated Operations



CATEGORY	LEVEL 0	LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 4
STRATEGY	– No data scientists hired – Skeptical of value of ML among an executive team	– Small and siloed data science and data engineering teams – A small number of ML champions in an executive team	– Small data science, data engineers and software development teams started to be coordinated – Small data science, data engineers and software development teams started to be coordinated	– Large, well-integrated teams across data science, engineering and software development – Chief Data Officer, and C-suite level sponsoship exists – New team members brought up to speed in weeks – Project checkpoints to ensure ML is considered for major projects	– ML seen as strategic, driving company initiatives – Well-governed process for ML delivery – Engineers & researchers are embedded on the same team
ARCHITECTURE	– Data silos with one-off integration – Data not prepared nor ready for ML	– Basic enterprise data ready for ML – Data architecture still immature – Tacit commitment to meaningful enterprise data in the cloud	– Data architecture is mature – Most enterprise data ready for ML in the cloud – Overt commitment to the cloud	– Enterprise data is well cataloged and managed – Automated data pipelines in place – ML configuration and infrastructure is managed – ML models automatically provisioned as microservices	– Comprehensive architecture to effectively govern all data – Consistent data storage and consumption pipeline across projects – Target ML infrastructure monitored for cost-effectiveness and optimal utilization
MODELING	– Manual process for model training – Limited pilot studies	– Manual ML model training and live pilots – Basic experiment tracking, no model management	– Experiment tracking and model management in place – Models dependencies not well understood	– Models cataloged through lifecycle, supporting reproducibility and reuse – Models dependencies not well understood – Output from ML is predictable and consistent, with auditable and reproducible outcomes	– Interdependencies between models are monitored and managed – Impact of small changes to ML models can be measured
PROCESSES	– No DevOps practices adopted – No clearly defined success criteria for ML projects	– DevOps practices like CI/CD have been adopted for non-ML components – No consistency in measures for ML or MLOps success	– Development iterative but CI/CD not in place for models – ML infrastructure expertise not broadly available – ML configuration is an afterthought – Metrics/measures in place but not consistent across projects	– Data tested for model applicability and monitored for changes in distribution – All artifacts (data sets, tests, models) under version control – DevOps practices like CI/CD, code reviews in place for ML code – Production MLOps pipeline flow includes packaging, deployment, serving and operational monitoring	– Comprehensive MLOps pipeline supporting frequent model updates – New algorithmic approaches can be tested at full scale – Automatic metrics gathering, alerts, issues analysis (such as data drift) and automated retraining of systems is in place
GOVERNANCE	– Not considered	– Not considered, though the organization may have broader views – No notion of the concept of bias in models	– Model explainability not considered – Models may harbor prediction bias – Model releases are tracked	– Security policies applied to models, data – Ethics and explainability consideration for models and ML systems – Good faith attempts to remove biased variables from models – Potential for malicious use of ML considered in ML lifecycle	– Cybersecurity experts engaged in ML operations – ML systems protected from external manipulation – End-to-end audit trail for ML - who, why, when

Reference: https://medium.com/@NickHystax/mlops-maturity-levels-the-most-well-known-models-5b1de94ea285

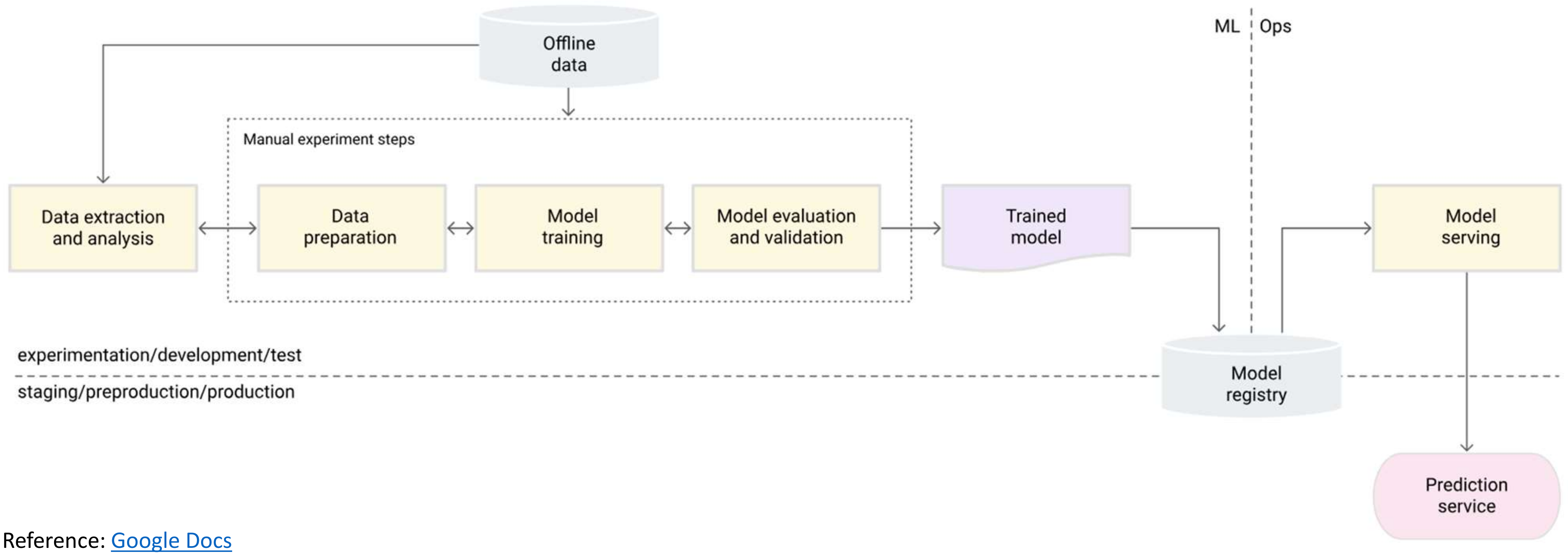
- **GigaOm Model**

Also, one of the most detailed and understandable models is from the analytical firm GigaOm. Of all the models, it is the closest to Capability Maturity Model Integration (CMMI). This is a set of process improvement methodologies in organizations, which also consists of five levels from 0 to 4.

Google Model

MLOps level 0: Manual process

Many teams have data scientists and ML researchers who can build state-of-the-art models, but their process for building and deploying ML models is entirely manual. This is considered the *basic* level of maturity, or level 0.



Reference: [Google Docs]()

MLOps level 0 Characteristics

Manual, script-driven, and interactive process:

- Every step is manual, including data analysis, data preparation, model training, and validation.
- It requires manual execution of each step, and manual transition from one step to another.
- This process is usually driven by experimental code that is written and executed in notebooks by data scientists interactively until a workable model is produced.

Disconnection between ML and operations:

- The process separates data scientists who create the model and engineers who serve the model as a prediction service.
- The data scientists hand over a trained model as an artefact to the engineering team to deploy on their API infrastructure.
- This handoff can include putting the trained model in a storage location, checking the model object into a code repository, or uploading it to a model registry. Then engineers who deploy the model need to make the required features available in production for low-latency serving, which can lead to *training-serving skew*.

Infrequent release iterations:

- The process assumes that your data science team manages a few models that don't change frequently—either changing model implementation or retraining the model with new data.
- A new model version is deployed only a couple of times per year.

No CI:

- Because few implementation changes are assumed, CI is ignored.
- Usually, testing the code is part of the notebooks or script execution.
- The scripts and notebooks that implement the experiment steps are source - controlled, and they produce artifacts such as trained models, evaluation metrics, and visualizations.

No CD:

- Because there aren't frequent model version deployments, CD isn't considered.

Deployment refers to the prediction service:

- The process is concerned only with deploying the trained model as a prediction service (for example, a microservice with a REST API), rather than deploying the entire ML system.

Lack of active performance monitoring:

- The process doesn't track or log the model predictions and actions, which are required in order to detect model performance degradation and other model behavioural drifts.

Reference: Google Docs

MLOps level 0 Challenges

MLOps level 0 is common in many businesses that are beginning to apply ML to their use cases.
This manual, data-scientist-driven process might be sufficient when models are rarely changed or trained.

In practice, models often break when they are deployed in the real world.
The models fail to adapt to changes in the dynamics of the environment or changes in the data that describes the environment. For more information, see Why Machine Learning Models Crash and Burn in Production.

To address these challenges and to maintain your model's accuracy in production, you need to do the following:

Actively monitor the quality of your model in production:
- Monitoring lets you detect performance degradation and model staleness.
- It acts as a cue to a new experimentation iteration and (manual) retraining of the model on new data.

Frequently retrain your production models:
- To capture the evolving and emerging patterns, you need to retrain your model with the most recent data.
- For example, if your app recommends fashion products using ML, its recommendations should adapt to the latest trends and products.

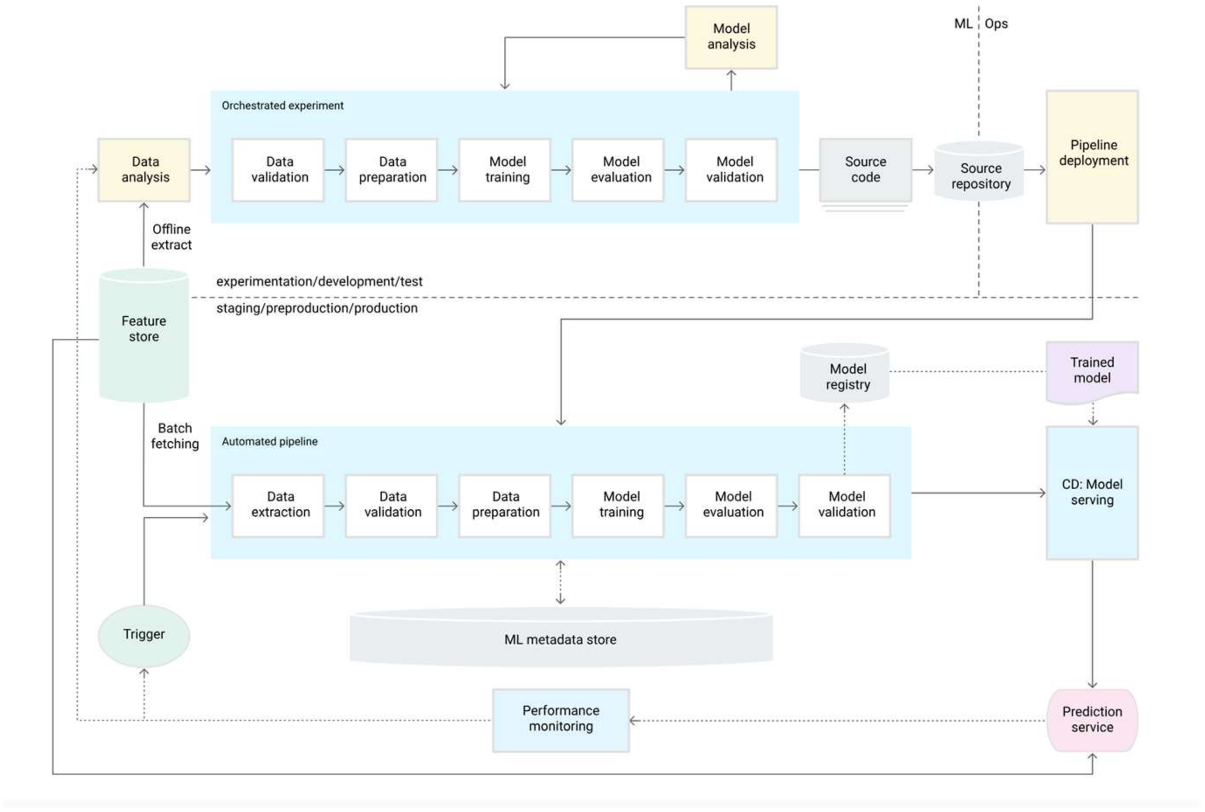
Continuously experiment with new implementations to produce the model:
- To harness the latest ideas and advances in technology, you need to try out new implementations such as feature engineering, model architecture, and hyperparameters.

To address the challenges of this manual process, MLOps practices for CI/CD and CT are helpful.
By deploying an ML training pipeline, you can enable CT, and you can set up a CI/CD system to rapidly test, build, and deploy new implementations of the ML pipeline.

Reference: Google Docs

MLOps level 0: Manual process

The goal of level 1 is to perform continuous training of the model by automating the ML pipeline; this lets you achieve continuous delivery of model prediction service.



Reference: Google Docs

MLOps level 1 Characteristics

Rapid experiment:

- The steps of the ML experiment are orchestrated. The transition between steps is automated, which leads to rapid iteration of experiments and better readiness to move the whole pipeline to production.

CT of the model in production:

- The model is automatically trained in production using fresh data based on live pipeline triggers.

Experimental-operational symmetry:

- The pipeline implementation that is used in the development or experiment environment is used in the preproduction and production environment, which is a key aspect of MLOps practice for unifying DevOps.

Modularized code for components and pipelines:

- To construct ML pipelines, components need to be reusable, composable, and potentially shareable across ML pipelines.
- Therefore, while the EDA code can still live in notebooks, the source code for components must be modularized.
- In addition, components should ideally be containerized to do the following:
  - Decouple the execution environment from the custom code runtime.
  - Make code reproducible between development and production environments.
  - Isolate each component in the pipeline. Components can have their own version of the runtime environment, and have different languages and libraries.

Continuous delivery of models:

- An ML pipeline in production continuously delivers prediction services to new models that are trained on new data.
- The model deployment step, which serves the trained and validated model as a prediction service for online predictions, is automated.

Pipeline deployment:

- In level 0, you deploy a trained model as a prediction service to production. For level 1, you deploy a whole training pipeline, which automatically and recurrently runs to serve the trained model as the prediction service.

Reference: Google Docs

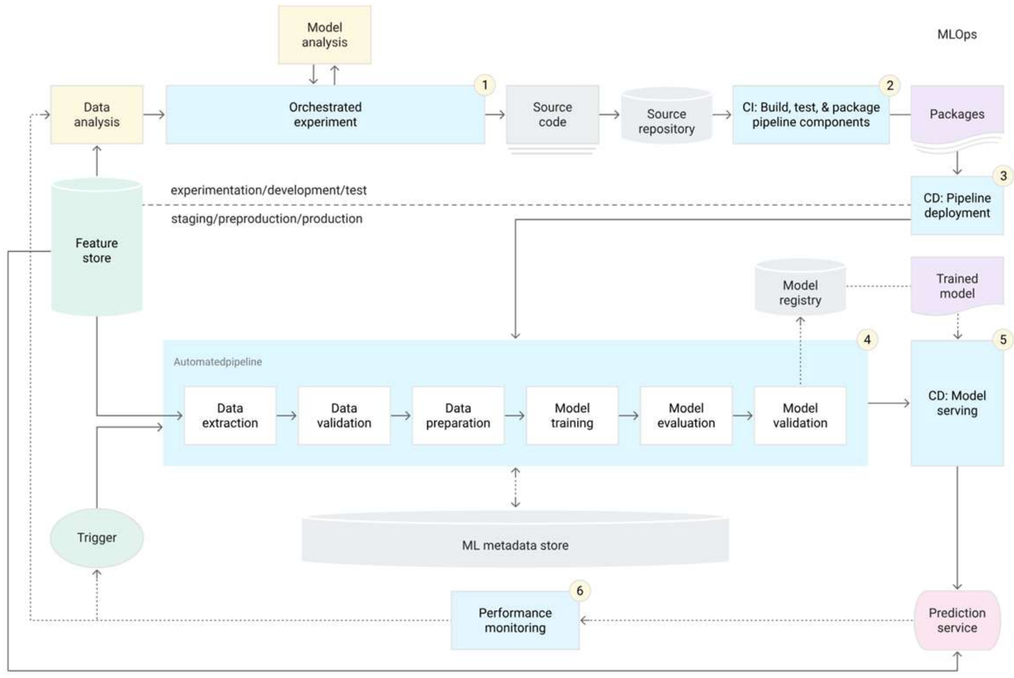
Challenges

- Assuming that new implementations of the pipeline aren't frequently deployed and you are managing only a few pipelines, you usually **manually test** the pipeline and its components.
- In addition, you **manually deploy** new pipeline implementations.
- You also submit the tested source code for the pipeline to the IT team to deploy to the target environment.
- This setup is suitable when you deploy new models based on new data, rather than based on new ML ideas.
- However, you need to try new ML ideas and rapidly deploy new implementations of the ML components.
- If you manage many ML pipelines in production, you need a CI/CD setup to automate the build, test, and deployment of ML pipelines.

Reference: Google Docs

MLOps level 2: CI/CD pipeline automation

For a rapid and reliable update of the pipelines in production, you need a robust automated CI/CD system. This automated CI/CD system lets your data scientists rapidly explore new ideas around feature engineering, model architecture, and hyperparameters. They can implement these ideas and automatically build, test, and deploy the new pipeline components to the target environment.



Reference: Google Docs

MLOps level 2 Characteristics

Development and experimentation:

- You iteratively try out new ML algorithms and new modelling where the experiment steps are orchestrated.
- The output of this stage is the source code of the ML pipeline steps that are then pushed to a source repository.

Pipeline continuous integration:

- You build source code and run various tests.
- The outputs of this stage are pipeline components (packages, executables, and artifacts) to be deployed in a later stage.

Pipeline continuous delivery:

- You deploy the artifacts produced by the CI stage to the target environment.
- The output of this stage is a deployed pipeline with the new implementation of the model.

Automated triggering:

- The pipeline is automatically executed in production based on a schedule or in response to a trigger.
- The output of this stage is a trained model that is pushed to the model registry.

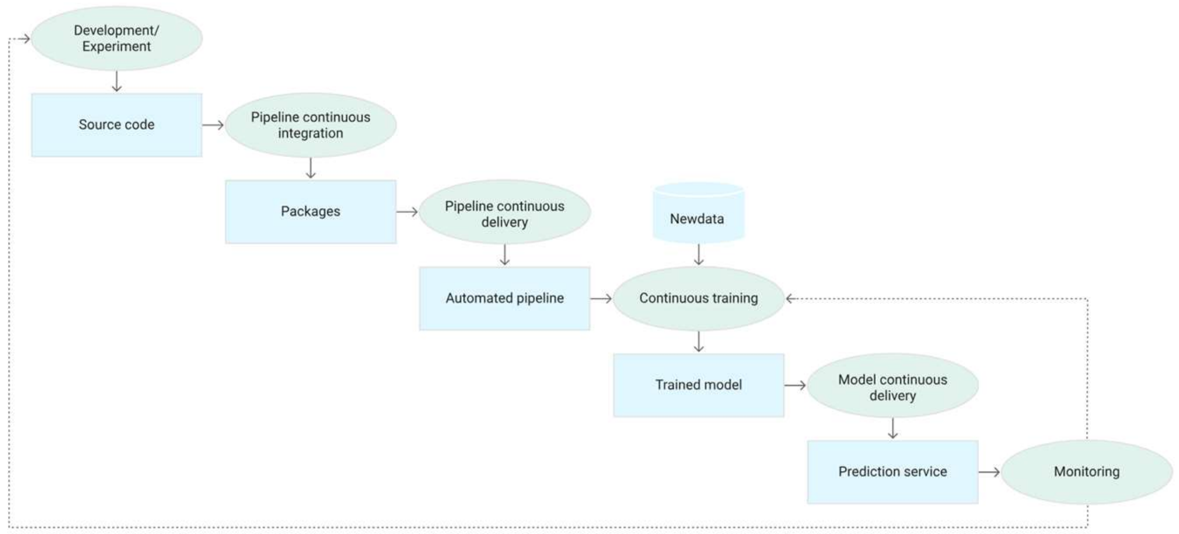
Model continuous delivery:

- You serve the trained model as a prediction service for the predictions.
- The output of this stage is a deployed model prediction service.

Monitoring:

- You collect statistics on the model performance based on live data.
- The output of this stage is a trigger to execute the pipeline or to execute a new experiment cycle.

Reference: Google Docs



*The data analysis step is still a manual process for data scientists before the pipeline starts a new iteration of the experiment. The model analysis step is also a manual process.

Azure MLOps Maturity Model

The MLOps maturity model helps clarify the Development Operations (DevOps) principles and practices necessary to run a successful MLOps environment. It's intended to identify gaps in an existing organization's attempt to implement such an environment. It's also a way to show you how to grow your MLOps capability in increments rather than overwhelm you with the requirements of a fully mature environment.

Use it as a guide to:

- Estimate the scope of the work for new engagements.
- Establish realistic success criteria.
- Identify deliverables you'll hand over at the conclusion of the engagement.

Azure MLOps Maturity Model

Level	Description	Highlights	Technology
0	No MLOps	• Difficult to manage full machine learning model lifecycle • The teams are disparate and releases are painful • Most systems exist as "black boxes," little feedback during/post deployment	• Manual builds and deployments • Manual testing of model and application • No centralized tracking of model performance • Training of model is manual
1	DevOps but no MLOps	• Releases are less painful than No MLOps, but rely on Data Team for every new model • Still limited feedback on how well a model performs in production • Difficult to trace/reproduce results	• Automated builds • Automated tests for application code
2	Automated Training	• Training environment is fully managed and traceable • Easy to reproduce model • Releases are manual, but low friction	• Automated model training • Centralized tracking of model training performance • Model management
3	Automated Model Deployment	• Releases are low friction and automatic • Full traceability from deployment back to original data • Entire environment managed: train > test > production	• Integrated A/B testing of model performance for deployment • Automated tests for all code • Centralized training of model training performance
4	Full MLOps Automated Operations	• Full system automated and easily monitored • Production systems are providing information on how to improve and, in some cases, automatically improve with new models • Approaching a zero-downtime system	• Automated model training and testing • Verbose, centralized metrics from deployed model

Reference: Azure MLOps Maturity Model

Level 0: No MLOps

People	Model Creation	Model Release	Application Integration
• Data scientists: siloed, not in regular communications with the larger team • Data engineers (if exists): siloed, not in regular communications with the larger team • Software engineers: siloed, receive model remotely from the other team members	• Data gathered manually • Compute is likely not managed • Experiments aren't predictably tracked • End result may be a single model file manually handed off with inputs/outputs	• Manual process • Scoring script may be manually created well after experiments, not version controlled • Release handled by data scientist or data engineer alone	• Heavily reliant on data scientist expertise to implement Manual releases each time

Level 1: DevOps no MLOps

People	Model Creation	Model Release	Application Integration
• Data scientists: siloed, not in regular communications with the larger team • Data engineers (if exists): siloed, not in regular communication with the larger team • Software engineers: siloed, receive model remotely from the other team members	• Data pipeline gathers data automatically • Compute is or isn't managed • Experiments aren't predictably tracked • End result may be a single model file manually handed off with inputs/outputs	• Manual process • Scoring script may be manually created well after experiments, likely version controlled • Is handed off to software engineers	• Basic integration tests exist for the model • Heavily reliant on data scientist expertise to implement model • Releases automated • Application code has unit tests

Level 2: Automated Training

People

- Data scientists: Working directly with data engineers to convert experimentation code into repeatable scripts/jobs
- Data engineers: Working with data scientists
- Software engineers: siloed, receive model remotely from the other team members

Model Creation

- Data pipeline gathers data automatically
- Compute managed
- Experiment results tracked
- Both training code and resulting models are version controlled

Model Release

- Manual release
- Scoring script is version controlled with tests
- Release managed by Software engineering team

Application Integration

- Basic integration tests exist for the model
- Heavily reliant on data scientist expertise to implement model
- Application code has unit tests

Level 3: Automated Model Deployment

People	Model Creation	Model Release	Application Integration
• Data scientists: Working directly with data engineers to convert experimentation code into repeatable scripts/jobs • Data engineers: Working with data scientists and software engineers to manage inputs/outputs • Software engineers: Working with data engineers to automate model integration into application code	• Data pipeline gathers data automatically • Compute managed • Experiment results tracked • Both training code and resulting models are version controlled	• Automatic release • Scoring script is version controlled with tests • Release managed by continuous delivery (CI/CD) pipeline	• Unit and integration tests for each model release • Less reliant on data scientist expertise to implement model • Application code has unit/integration tests

Level 4: Full MLOps Automated Retraining

People	Model Creation	Model Release	Application Integration
• Data scientists: Working directly with data engineers to convert experimentation code into repeatable scripts/jobs. Working with software engineers to identify markers for data engineers • Data engineers: Working with data scientists and software engineers to manage inputs/outputs • Software engineers: Working with data engineers to automate model integration into application code. Implementing post-deployment metrics gathering	• Data pipeline gathers data automatically • Retraining triggered automatically based on production metrics • Compute managed • Experiment results tracked • Both training code and resulting models are version controlled	• Automatic Release • Scoring Script is version controlled with tests • Release managed by continuous integration and CI/CD pipeline	• Unit and Integration tests for each model release • Less reliant on data scientist expertise to implement model • Application code has unit/integration tests