**MLOPs (Machine Learning Operations)** is a set of practices that combines machine learning, DevOps, and data engineering to automate and manage the entire ML lifecycle — from data preparation and model training to deployment, monitoring, and updating. It ensures models are production-ready, scalable, reproducible, and continuously improving, enabling faster and more reliable delivery of ML solutions.

Below are the problems MLOPS solves or this are the aspects of the MLOPS.

1. Data Management

a. Data Collection

b. Data Pre-processing

c. Data Validation

d. Data Security

e. Data Compliance

f. Feature Store

2. Development Practices: Modular coding, developing the code modules for different parts and then combining.

3. Version Control:

- Data versioning

- Code Versioning

- Model versioning

4. Experimental Tracking:

- Tracking Experiments

- Test and validation

- Model registry

**5. Model Serving and CI/CD:**

- Continuous integration

- Containerization

- Continuous deployment

**6. Automation:**

- **Pipeline automation** [Data ingestion pipeline, model training pipeline, model validation and testing, model deployment, model monitoring and retraining].

- **Orchestration:** coordinating and automating the various steps involved in the ML lifecycle.

**7. Monitoring and Retraining:**

- Monitoring the metrics of the deployed model.

- Drift Detection

- Retraining

**8. Infrastructure Management:**

- Cloud based solutions to handle scalability concerns

- Cost management

- Managing multiple vendors

**9. Collaboration and operations:**

- Unified workspace

- Role based access

**10. Governance and Ethics**

MLOps refers to the practice and discipline within machine learning that aims to unify and streamline the machine learning system development (Dev) and machine learning system operation (Ops). It involves collaboration between data scientists, ML engineers, and IT professionals to automate and optimize the end-to-end lifecycle of machine learning applications.

**Benefit of MLOps:**

1. Scalability

2. Improved performance

3. Reproducibility

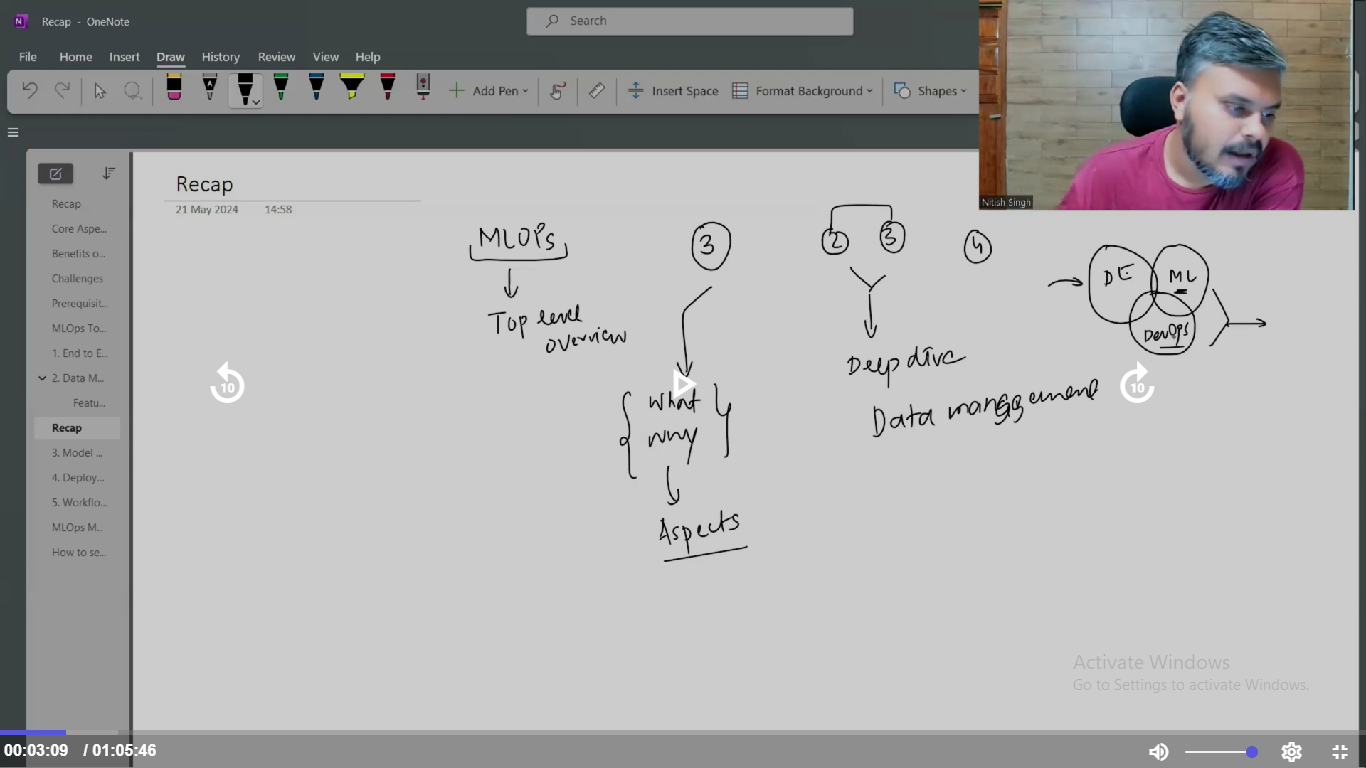
4. Collaboration and efficiency

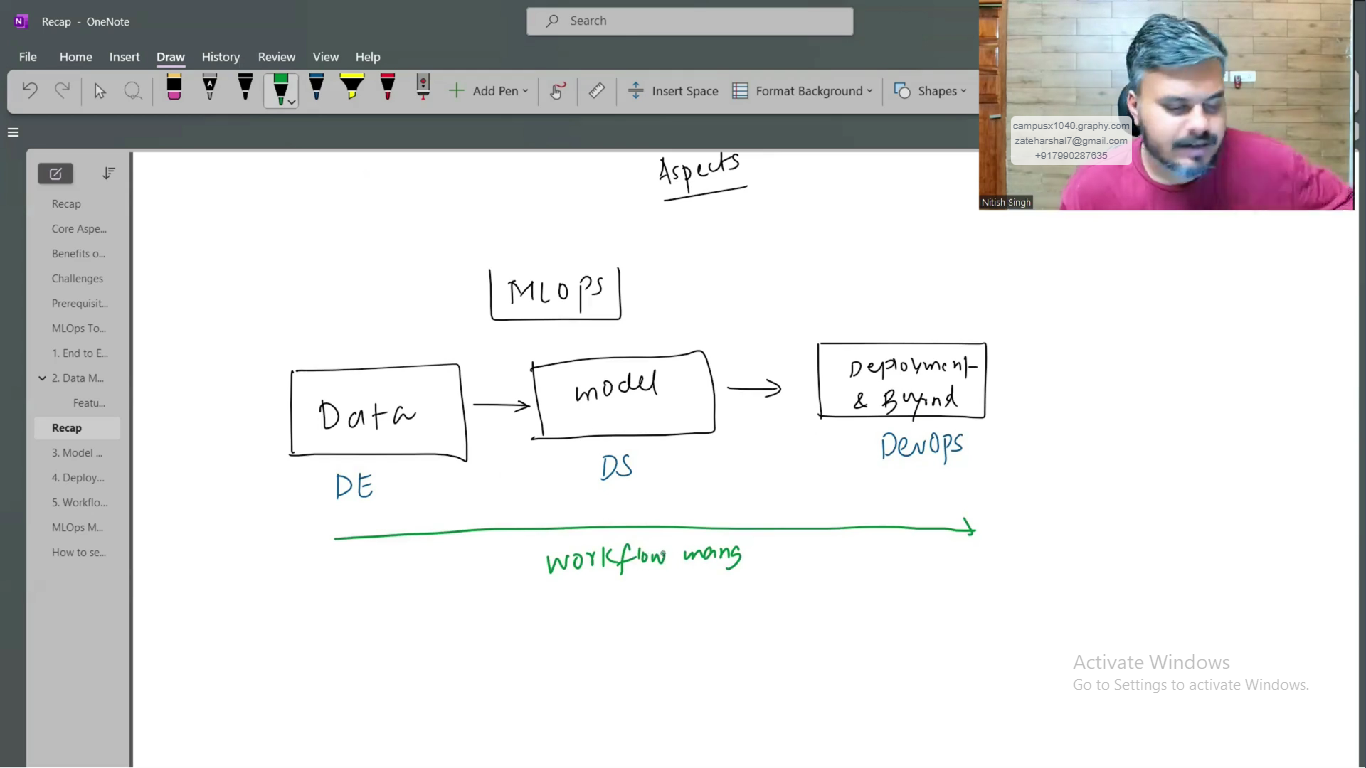
5. Risk reduction

6. Cost Savings

7. Faster time to market

8. Better compliance and governance





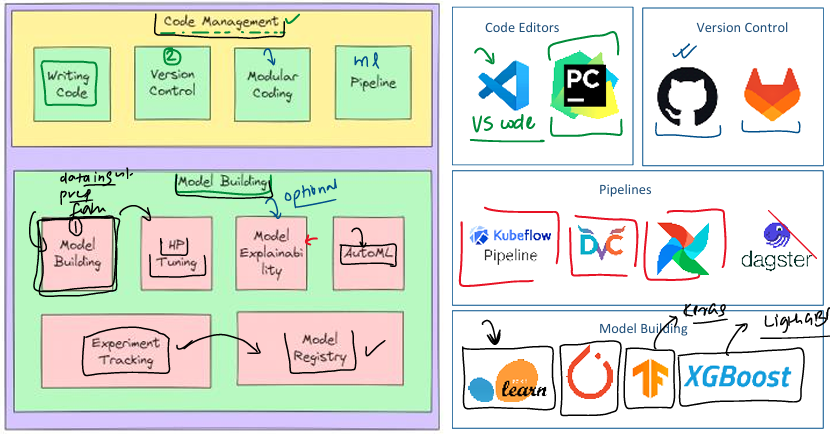
1. **Data Engineering**

**ETL (Extract, Transform, and Load)** is a data integration process used to move data from multiple sources into a centralized system, typically a data warehouse. It involves three key steps:

* **Extract**: Pulling raw data from various sources (e.g., databases, APIs, files)
* **Transform**: Cleaning, filtering, and converting the data into a desired format or structure
* **Load**: Storing the transformed data into a target system for analysis and reporting

Rest is to be refereed from the CampusX notes.

1. **Model Building:**



* **Modular coding** is a software development practice where a program is divided into small, self-contained units called modules, each responsible for a specific task. This approach enhances code **reusability**, **readability**, and **maintainability**, as modules can be developed, tested, and updated independently without affecting the entire system. It also promotes **scalability**, allows for **faster development** through parallel work, and improves **debugging and testing** by isolating functionality. By organizing code into logical, well-defined components, modular coding supports clean architecture, better collaboration among developers, and more robust, flexible software systems.
* **Cookiecutter** supports modular coding by automating the creation of structured, modular project templates. It allows developers to scaffold projects with predefined folders and files for separate modules (e.g., core/, utils/, api/), ensuring consistent architecture and separation of concerns. This promotes reusability, easier maintenance, and faster development by eliminating repetitive setup tasks. By enforcing a standard modular layout across projects and teams, Cookiecutter helps streamline development while aligning with clean, maintainable coding practices.
* **Pipelines:** A **machine learning pipeline** is a structured and automated sequence of steps that takes raw data through various stages—like pre-processing, training, evaluation, and deployment—to produce a final model. Instead of running each step manually, the pipeline handles everything in the correct order, ensuring consistency and saving time. It tracks inputs, outputs, and changes, so only the necessary steps are re-executed when something is updated. This makes the workflow **reproducible**, **efficient**, and easy to **collaborate** on, helping teams build and manage ML models in a reliable, scalable way. Tools use are Kubeflow, DVC, dagstar
* ML Model development: Everything is known and stuff is similar.
* Hyper parameter tuning: Process is similar, tools are Optuna and HyperOpt
* **Model Explainability:** is a set of techniques that make AI model decisions **understandable and transparent** to humans. It helps users see **why** a model made a prediction, **which features influenced it**, and ensures **trust, accountability**, and **fairness**, especially in critical fields like healthcare, finance, and law.

Many powerful AI models (like deep learning) are "black boxes" — they make accurate predictions but **don’t clearly show how or why** they reached a conclusion. XAI helps answer questions like:

* Why did the model make this prediction?
* What features influenced the decision?
* Can I trust this output?

Tools like SHAPLEY, Lime are used.

* **AutoML (Automated Machine Learning)** is the process of automating the end-to-end steps of building machine learning models — including **data preprocessing**, **feature selection**, **model selection**, **training**, and **hyperparameter tuning** — with minimal human intervention. It helps both experts and non-experts quickly build high-performing models, saving time and reducing manual effort while ensuring consistency and scalability. Tools, Cloub AutoML (google), PYCARET, Katib, Auto Sklearn etc.
* **Experiment tracking** in machine learning is the process of logging and managing all details of model training runs — such as datasets used, code version, hyperparameters, metrics, and results. It helps ensure **reproducibility**, **easy comparison**, and **collaboration** by keeping a clear record of what was tried and what worked best. Tools like **MLflow** or **Weights & Biases** automate this process, making model development more organized and efficient.
* **Model registry** in MLOps is a centralized repository or system that manages the storage, versioning, and lifecycle of machine learning models. It serves as a catalog where models are tracked from development to deployment, ensuring that the best-performing models are readily available for production. The model registry is essential for managing the complexity of deploying and maintaining models in a reliable and reproducible manner.

1. **Model Versioning:** Track different versions of a model as it evolves over time.
2. **Metadata Management:** Store and manage metadata associated with each model.
3. **Model Lineage Tracking**: Track the lineage and provenance of models to understand their development history.
4. **Stage Management:** Manage the lifecycle stages of a model (e.g., development, staging, production).
5. **Model Storage and Retrieval:** Store and provide access to model artifacts, It means the model registry **saves all the files needed to use or redeploy the model,** and makes them **easily accessible** to others or to automated systems.