

The background features abstract geometric shapes in various shades of blue. On the left, a solid light blue triangle points upwards. On the right, a complex arrangement of overlapping triangles in light blue, medium blue, and dark blue creates a dynamic, layered effect. The word "MLOPS" is centered in the white space between these blue elements.

MLOPS

# Model Lifecycle Management

- ▶ MLOps emphasizes managing the full ML lifecycle, including data ingestion, model training, evaluation, deployment, monitoring, and retraining.
- ▶ A well-structured ML lifecycle with clearly defined stages helps ensure models are developed, deployed, and maintained efficiently and transparently.

# Version Control for Code and Models

- Just as version control (e.g., Git) is crucial in software development, **versioning code and models** is fundamental in MLOps.
- This involves tracking different versions of datasets, model code, configurations, and trained models, so you can reproduce results, roll back to previous versions, or retrain models with updated data.

# Automated Model Training Pipelines

- Automated training pipelines make it easy to regularly retrain models, especially when new data becomes available.
- These pipelines use tools like **Airflow**, **Kubeflow Pipelines**, or **MLflow** to automate workflows, reduce manual errors, and ensure reproducibility.
- CI/CD principles are extended to ML through **Continuous Integration (CI)** (automating testing and validation) and **Continuous Delivery (CD)** (automating deployment of models).

# Feature Engineering and Feature Stores

- **Feature engineering** is a critical part of ML that involves creating meaningful input variables (features) from raw data.
- MLOps uses **feature stores** to store and manage features for reuse across models and teams. This ensures consistency in feature calculations and speeds up the development process by providing a shared library of features.

# Model Deployment and Serving

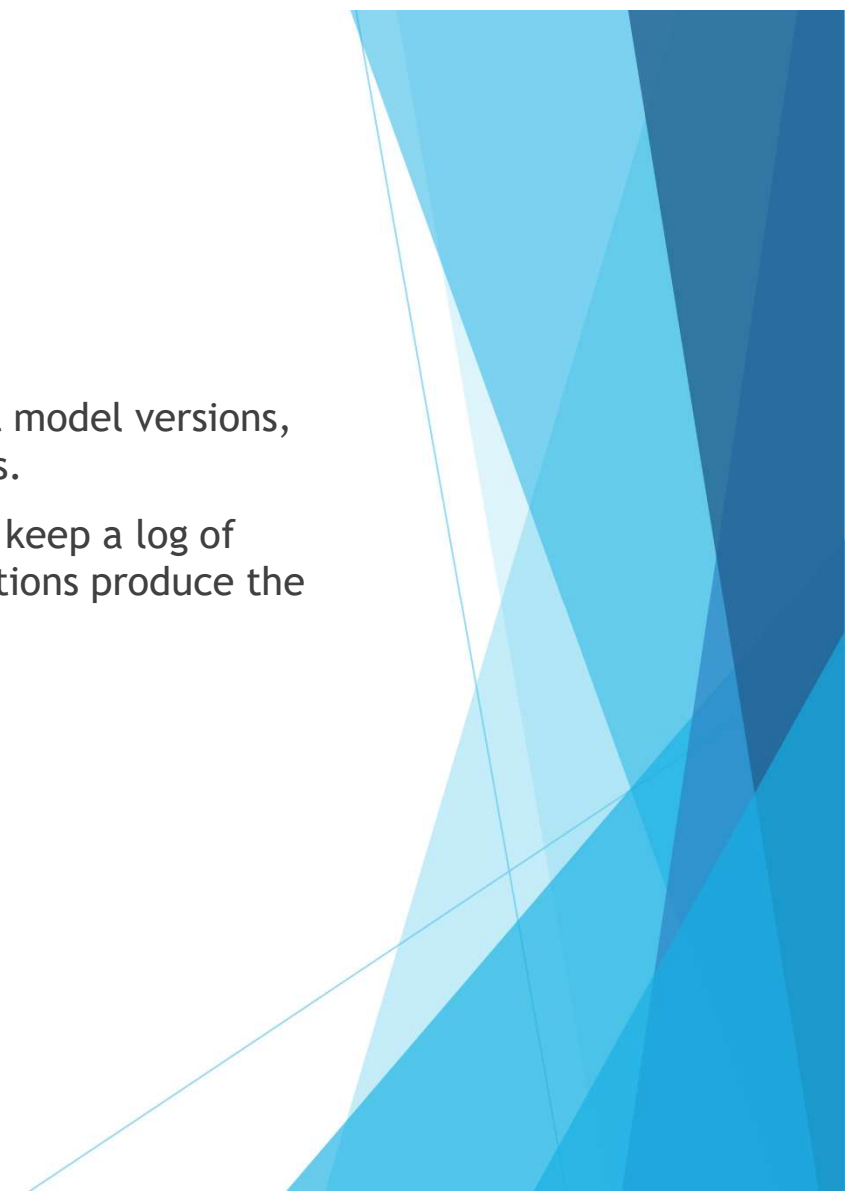
- **Deployment** involves putting a model into a production environment where it can make real-time predictions or batch predictions.
- **Model serving** is the process of exposing the model so it can be accessed by applications or end users, typically via REST APIs or other interfaces.
- Deployment strategies include **A/B testing**, **canary deployments**, and **blue-green deployments** to validate model performance safely before scaling it fully.

# Monitoring and Model Drift Detection

- MLOps frameworks continuously monitor deployed models to ensure they perform as expected over time.
- **Model drift** happens when the statistical properties of input data or relationships change, causing model accuracy to degrade.
- Monitoring tools and dashboards track model accuracy, prediction confidence, latency, and other metrics to detect and address performance degradation or data drift.

# Experiment Tracking

- MLOps includes tools to track experiments, such as different model versions, hyperparameters, feature selections, and evaluation metrics.
- **Experiment tracking** helps data scientists and ML engineers keep a log of experiments, making it easier to understand which combinations produce the best results.





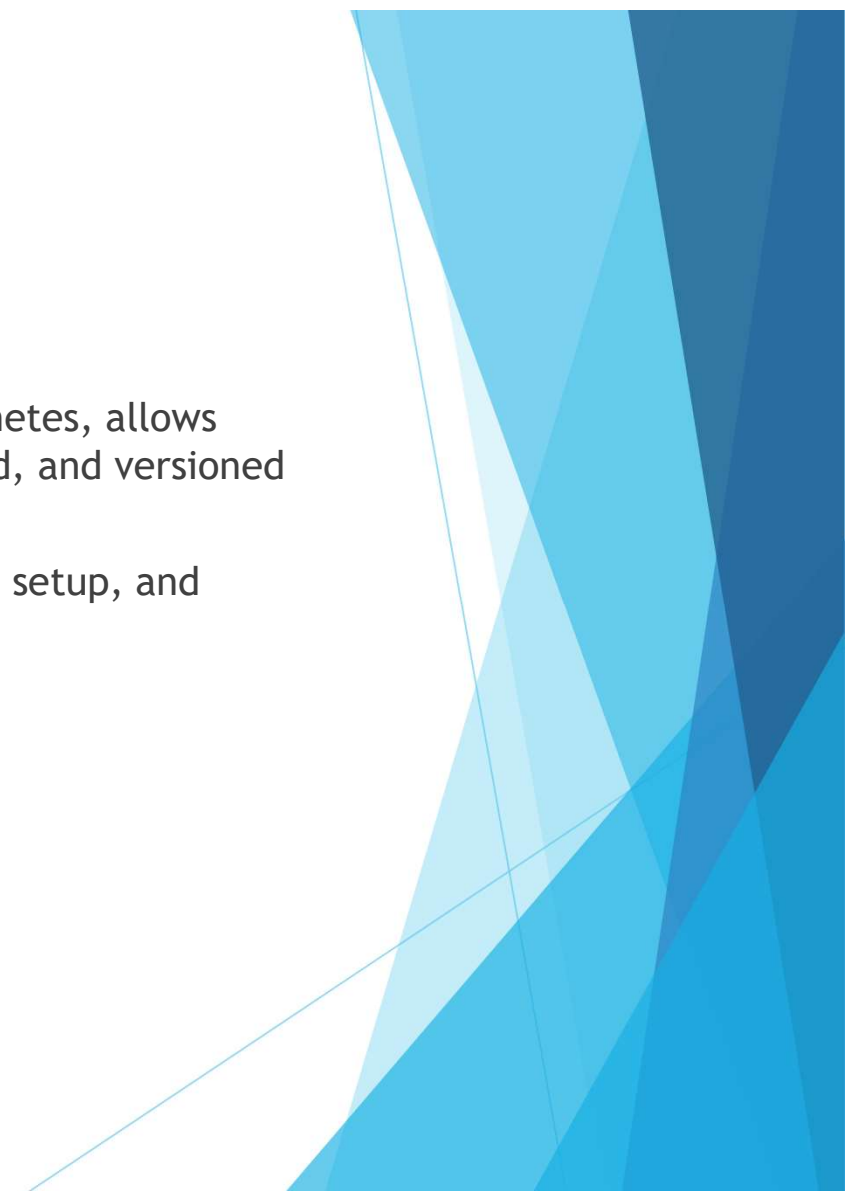
# Data and Model Lineage

- **Data lineage** tracks the origin, transformations, and flow of data through the pipeline to maintain transparency and regulatory compliance.
- **Model lineage** allows tracking of which datasets, features, and hyperparameters were used to train each model version, supporting reproducibility and debugging.



# Infrastructure as Code (IaC)

- Using **Infrastructure as Code (IaC)**, like Terraform or Kubernetes, allows infrastructure setup for ML pipelines to be defined, managed, and versioned as code.
- IaC makes it easier to scale resources up or down, automate setup, and maintain consistency across different environments.



# Security and Compliance

- MLOps considers **data security** (e.g., data encryption, access control) and **model security** (e.g., vulnerability testing) to protect models and data.
- **Compliance** requirements, such as GDPR for data privacy, are also integrated into MLOps practices, especially in regulated industries.

# Tools and Technologies in MLOps

- MLOps relies on a variety of tools to manage different lifecycle stages:
  - **Data management:** Data versioning tools like DVC, Delta Lake, and feature stores like Feast.
  - **Experiment tracking:** MLflow, Weights & Biases, or TensorBoard.
  - **Pipeline orchestration:** Apache Airflow, Kubeflow Pipelines, and Prefect.
  - **Deployment:** Docker, Kubernetes, Seldon Core, and TensorFlow Serving.
  - **Monitoring:** Prometheus, Grafana, and monitoring platforms like Arize or Evidently.

# Collaborative and Reproducible Workflows

- MLOps emphasizes collaborative, reproducible workflows that enable data scientists and ML engineers to work together seamlessly.
- Through shared pipelines, experiment tracking, and reproducible environments (e.g., Docker, Conda), MLOps promotes collaboration and reduces dependency on individual team members.

