The Hoang

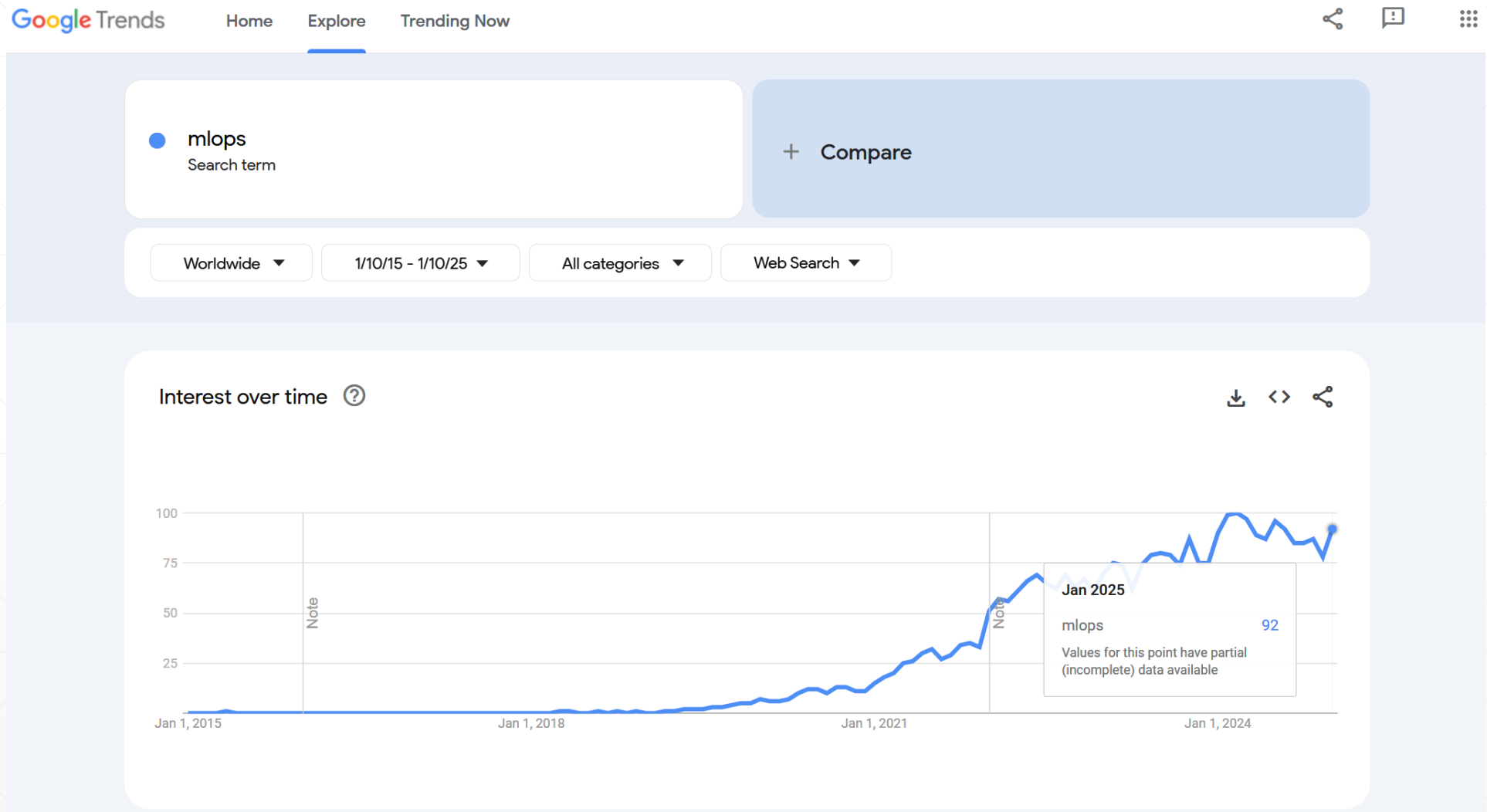
MLOps Introduction

Understand and Implement Production-Grade Machine Learning Operations

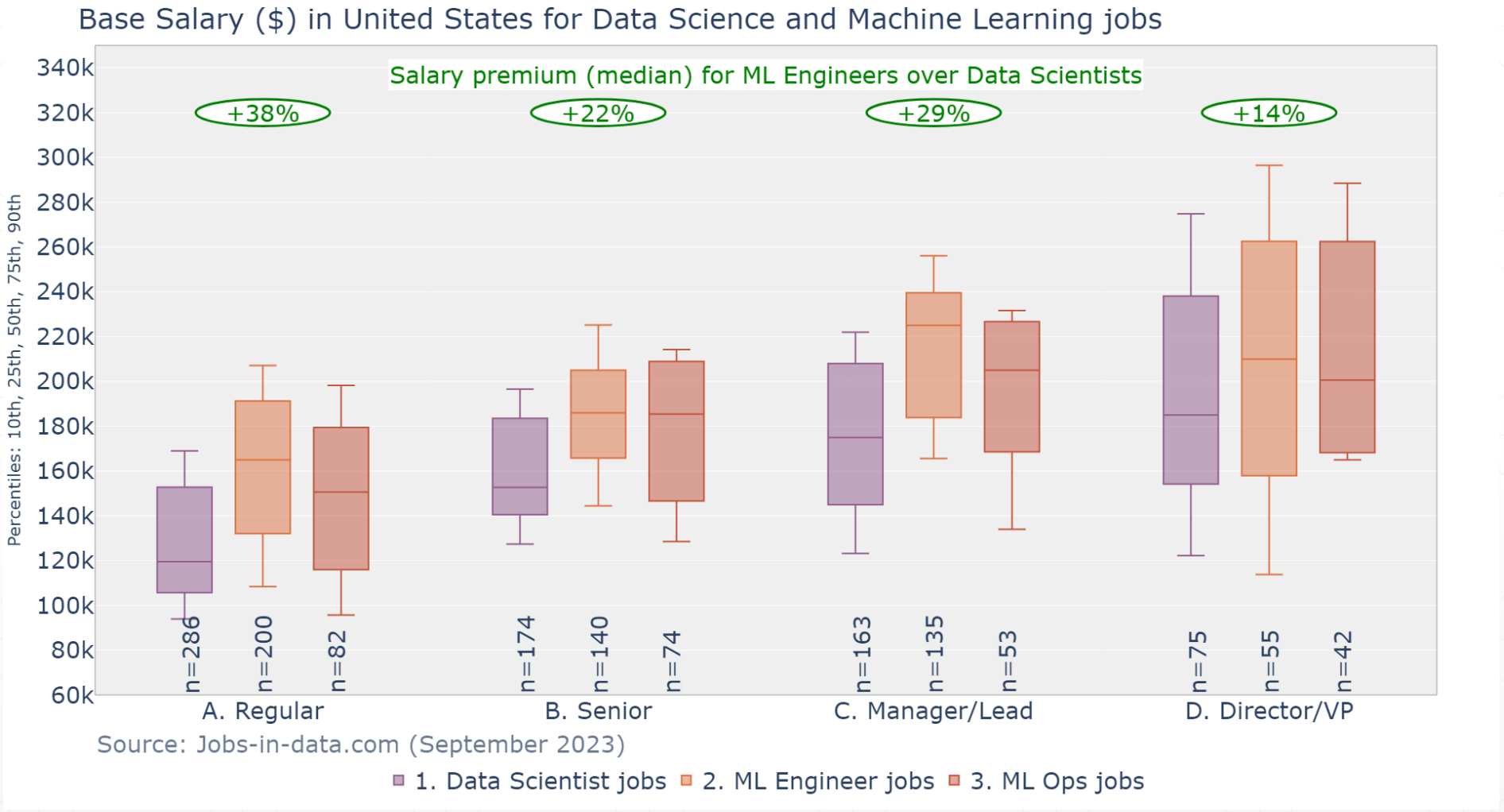
Lecture

- Why MLOps?
- What is MLOps?
- Why is it important?
- Overview of the MLOps lifecycle.
- The MLOps Principles.
- Key roles: Data Scientists, ML Engineers, DevOps.
- Tools and Technologies in MLOps.

Why MLOps? – Propensity

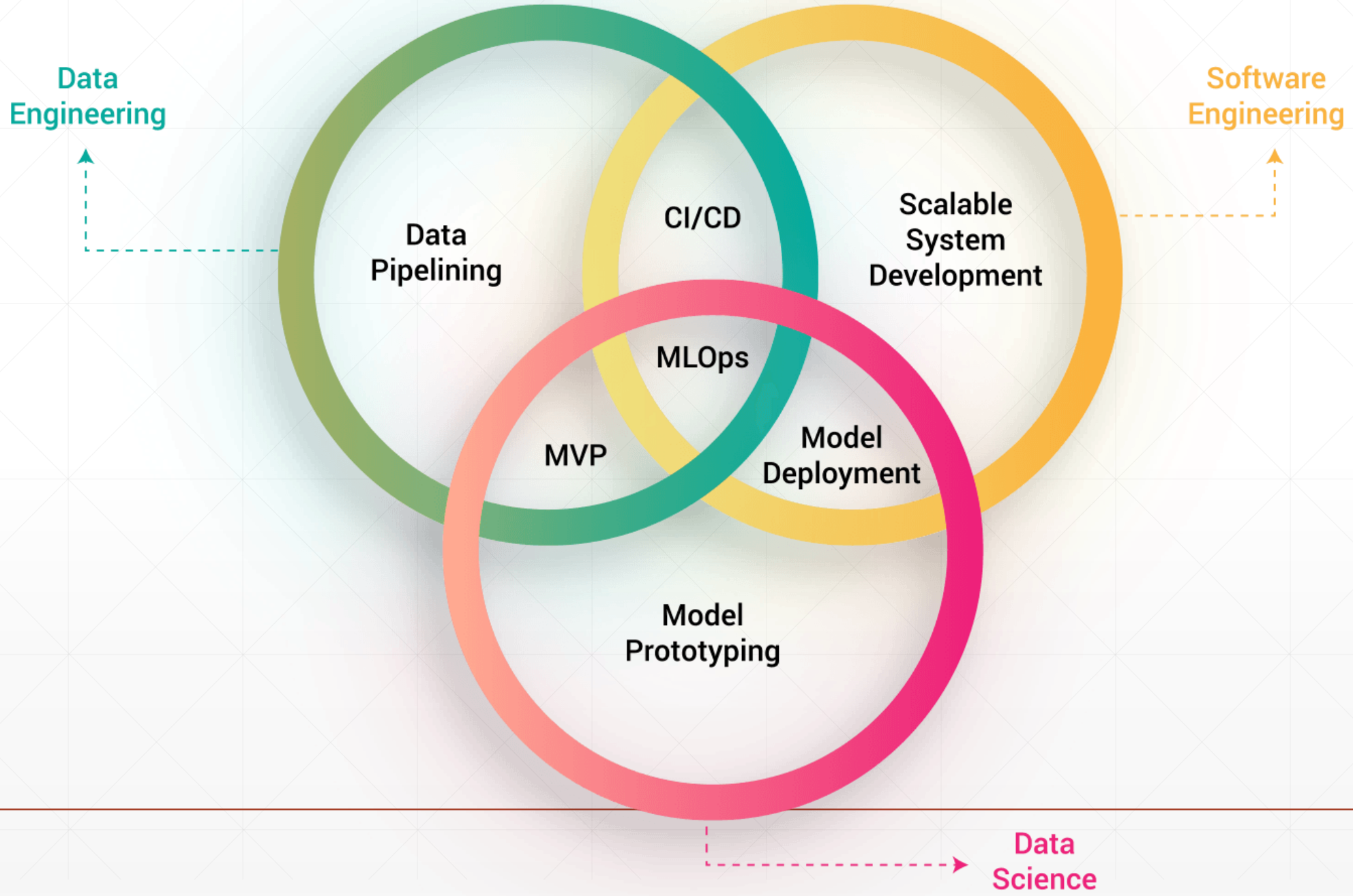


Why MLOps? – Career Prospects



What is MLOps?

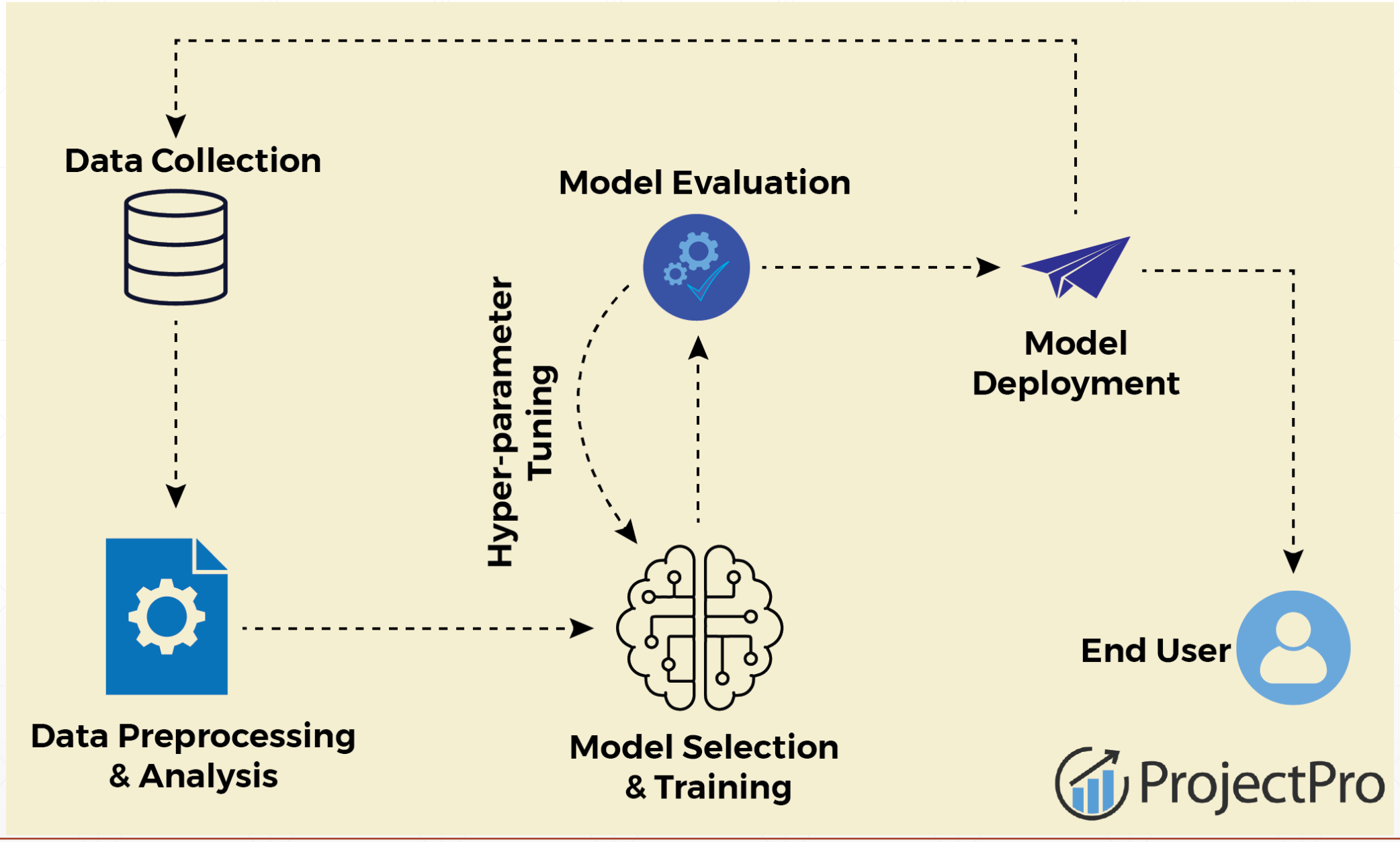
- MLOps (**M**achine **L**earning **Op**eration**s**) is a set of **practices**, **tools**, and **frameworks** designed to streamline and automate the deployment, monitoring, and management of machine learning (ML) models in production environments**.**
- It is an extension of DevOps (**Dev**elopment + **Op**eration**s**) but specifically tailored for ML systems, which involve unique challenges, such as managing data, retraining models, and handling versioning of both code and models.

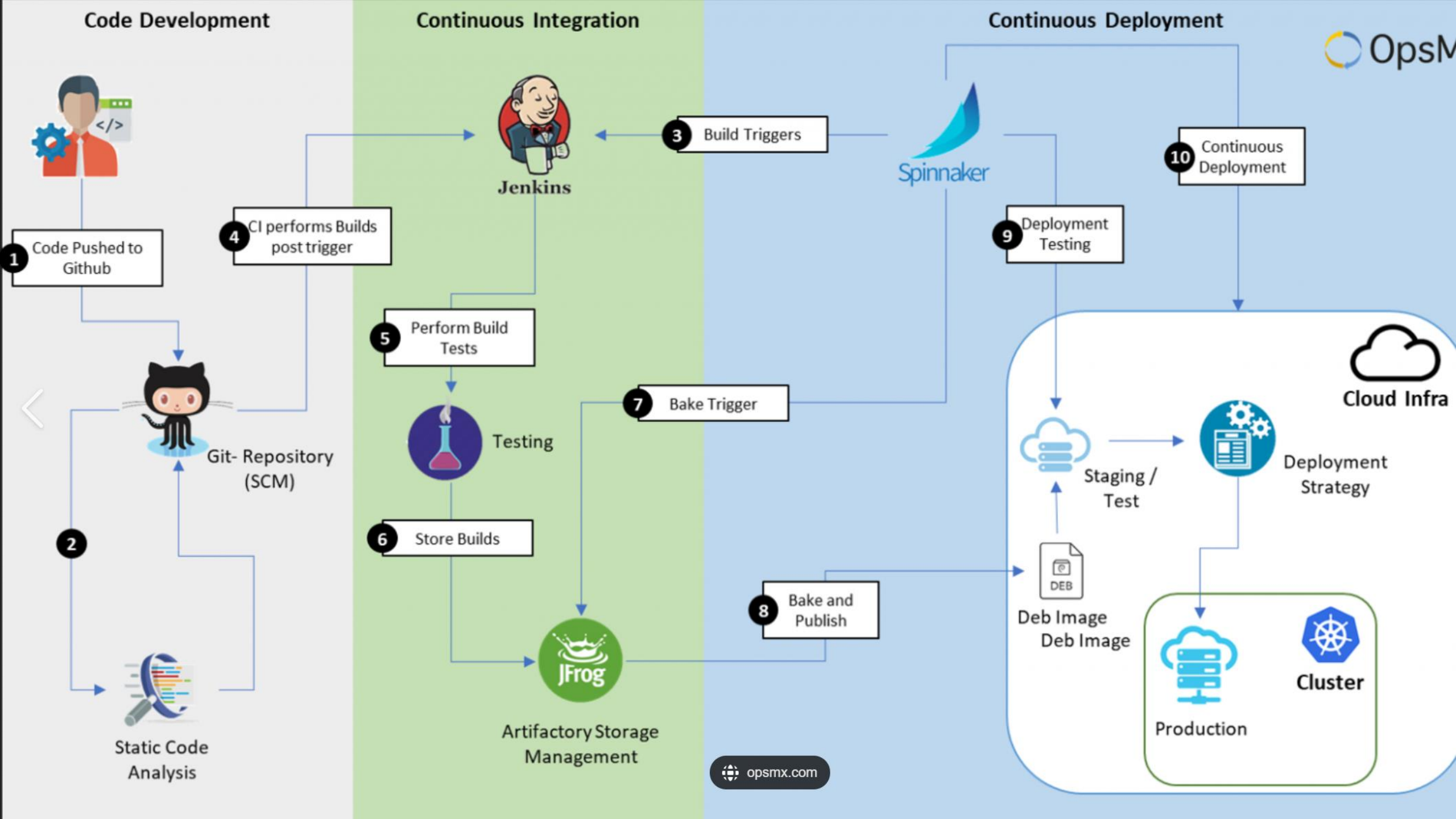

Data
Engineering
Software
Engineering
Data
Pipelining
CI/CD
Scalable
System
Development
MLOps
MVP
Model
Deployment
Model
Prototyping
Data
Science

Why is it important? (1/10)

Streamlined Deployment Process

- MLOps automates the deployment of machine learning models, reducing manual intervention and errors.
- Ensures CI/CD pipelines for both code (software) and models, enabling faster and more reliable releases.

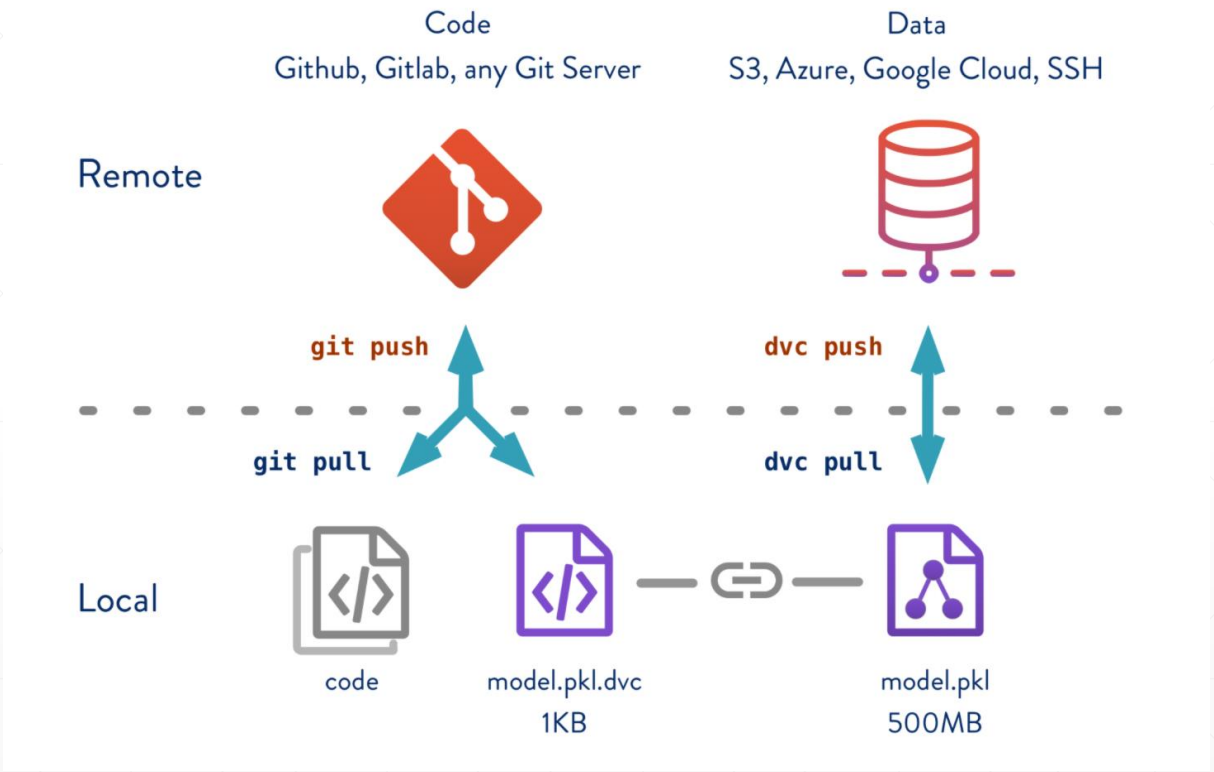

Data Collection
Model Evaluation
Hyper-parameter Tuning
Model Deployment
Data Preprocessing & Analysis
Model Selection & Training
End User
ProjectPro


Code Development
Continuous Integration
Continuous Deployment
OpsM
1 Code Pushed to Github
2
Git- Repository (SCM)
Static Code Analysis
4 CI performs Builds post trigger
Jenkins
3 Build Triggers
5 Perform Build Tests
Testing
6 Store Builds
JFrog
Artifactory Storage Management
7 Bake Trigger
8 Bake and Publish
Spinnaker
9 Deployment Testing
10 Continuous Deployment
Cloud Infra
Staging / Test
Deployment Strategy
DEB
Deb Image
Deb Image
Cluster
Production
opsmx.com

Why is it important? (2/10)

Model Versioning and Reproducibility

- Tracks and manages different versions of models, datasets, and code to ensure reproducibility.
- Allows teams to roll back to previous versions of models and experiments if issues arise.



Why is it important? (3/10)

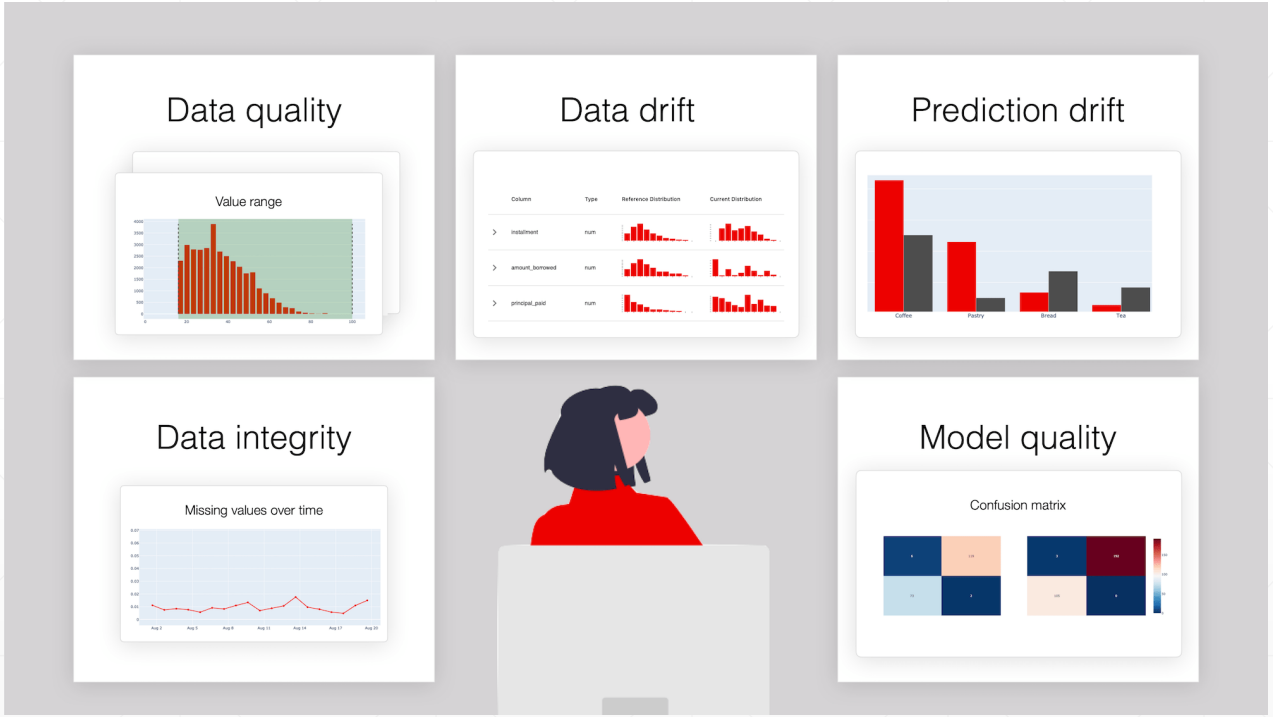
Scalability

- Provides tools and frameworks to scale ML workflows as data and user demands grow.
- Supports distributed training, large-scale data processing, and deployment across multiple environments (cloud, edge, or on-premises).

Why is it important? (4/10)

Monitoring and Model Performance Management

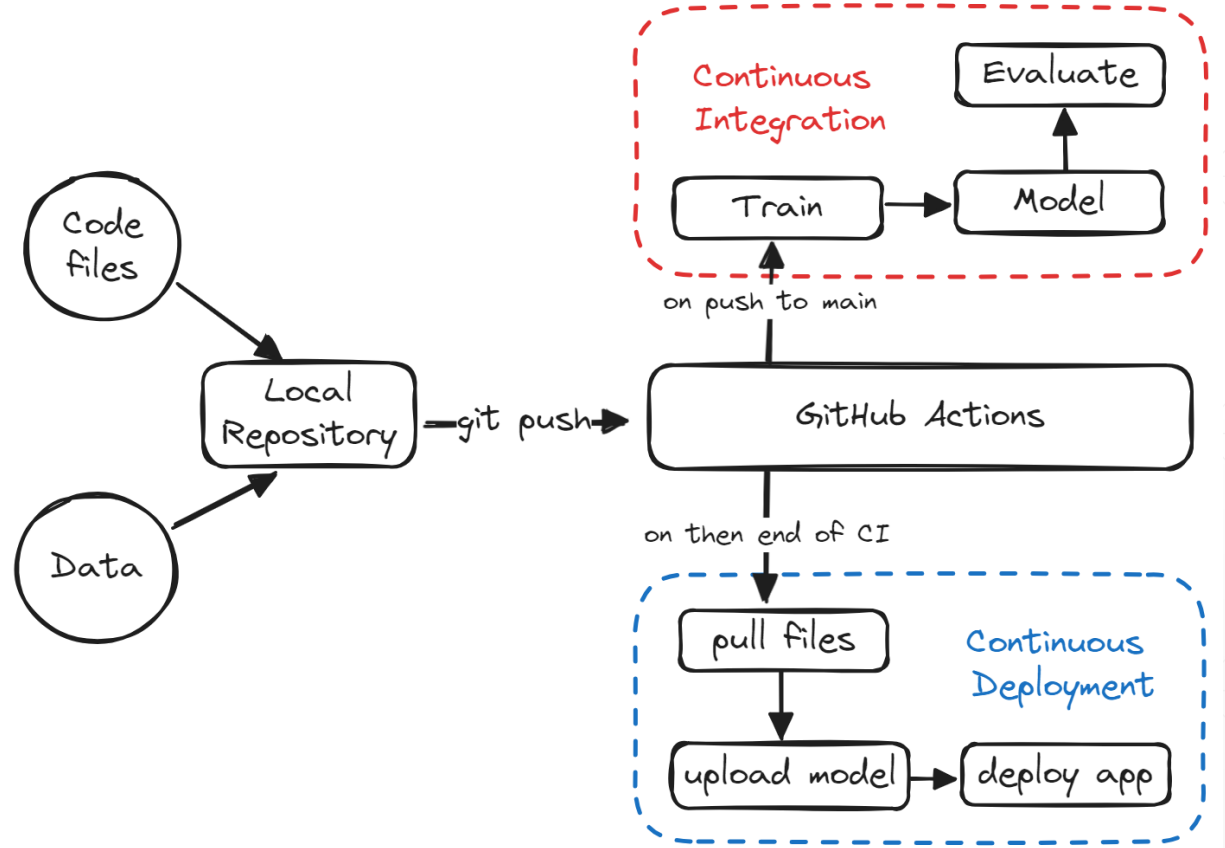
- Enables real-time monitoring of model performance in production.
- Tracks metrics like accuracy, latency, drift, and errors to ensure models are delivering expected results.
- Detects concept drift or data drift, allowing for timely retraining or updates.



Why is it important? (5/10)

Automation of Retraining and Continuous Learning

- Automates retraining pipelines when new data becomes available or when performance degrades.
- Creates an ecosystem for continuous learning, keeping models up to date with changing patterns in the data.



Why is it important? (6/10)

Collaboration Between Teams

- Promotes collaboration between data scientists, ML engineers, and DevOps teams.
- Creates standard processes and tools that make it easier for teams to work together and share knowledge.

Why is it important? (7/10)

Efficient Resource Management

- Optimizes infrastructure usage by dynamically allocating resources for training, testing, and serving models.
- Reduces operational costs by integrating with cloud platforms and leveraging tools like Kubernetes or serverless computing.

Why is it important? (8/10)

Compliance and Governance

- Ensures that machine learning workflows comply with regulations (e.g., GDPR, HIPAA) by tracking data lineage, audit logs, and model explainability.
- Provides a framework for ethical AI, ensuring transparency and accountability in decision-making.

Why is it important? (9/10)

Improved Reliability and Fault Tolerance

- Enhances the robustness of ML systems through redundancy, rollback mechanisms, and error-handling strategies.
- Reduces downtime by enabling quick recovery from failures or bugs.

Why is it important? (10/10)

Faster Time-to-Market

- Speeds up the transition from experimentation to production by automating repetitive tasks like feature engineering, model training, and deployment.
- Shortens the feedback loop for model improvements, enabling businesses to iterate faster.

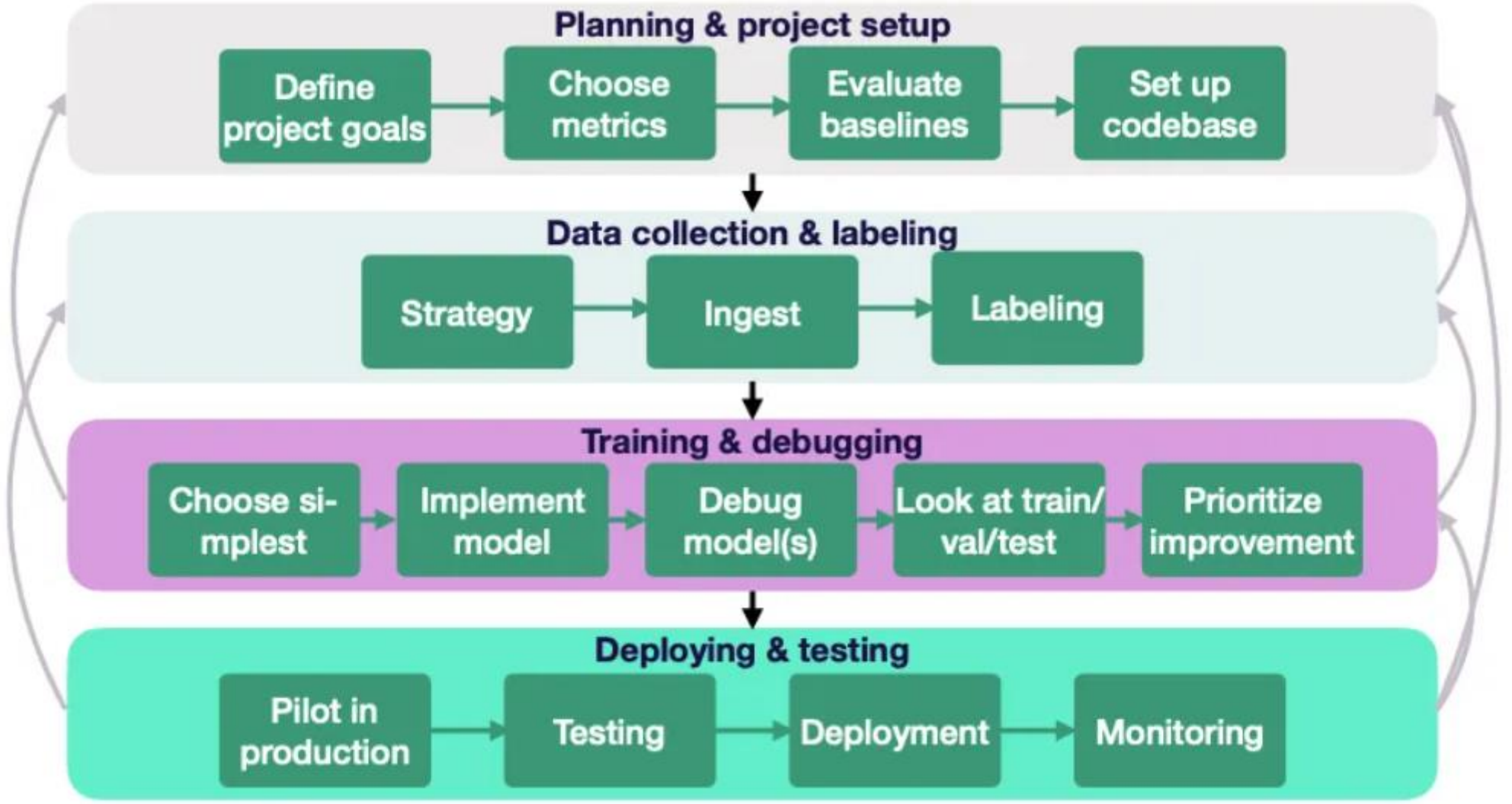
Example

- **Without MLOps**: The data scientist trains the model, hands it over to the engineering team, and manually deploys it. Six months later, the model's performance deteriorates because of changes in fraud patterns, but there's no automated way to detect or fix it.
- **With MLOps**: The bank uses an MLOps pipeline to continuously monitor the model, retrain it with fresh fraud data, and redeploy it automatically, ensuring consistent performance.

Overview of the MLOps Lifecycle – 1/

- The MLOps lifecycle consists of several interconnected stages that ensure ML models are successfully developed, deployed, and maintained.

Overview of the MLOps Lifecycle – 2/



MLOps Lifecycle – 1/7

Problem Definition and Data Understanding

- Collaborate with stakeholders to define the problem.
- Gather and analyze data to understand its structure and quality.

Example: A retailer wants to predict customer churn. The data team collects transaction histories, demographics, and customer feedback.

MLOps Lifecycle – 2/7

Data Preparation

- Clean, preprocess, and transform data into a usable format.
- Split the data into training, validation, and test sets.

Example: Remove null values, encode categorical variables, and normalize numerical data for the churn dataset.

MLOps Lifecycle – 3/7

Model Development

- Train multiple ML models (e.g., decision trees, neural networks) using the prepared data.
- Evaluate models using metrics like accuracy, precision, recall, or F1 score.

Example: A data scientist trains a logistic regression model on customer data and evaluates it using AUC-ROC scores.

MLOps Lifecycle – 4/7

Model Packaging

- Package the chosen model with its dependencies for deployment (e.g., using Docker or model serialization like pickle or ONNX).

Example: Save the trained churn prediction model as a .pkl file and build a Docker container to run it.

MLOps Lifecycle – 5/7

Model Deployment

- Deploy the model to a production environment using APIs, batch systems, or streaming platforms.

Example: Host the churn prediction model on a REST API so customer data can be sent in real-time for predictions.

MLOps Lifecycle – 6/7

Monitoring and Maintenance

- Track the model’s performance in production.
- Detect and fix data or concept drift (when data patterns change over time).
- Retrain the model as needed.

Example: Monitor the churn model's accuracy over time. If the accuracy drops below 90%, trigger a retraining pipeline.

MLOps Lifecycle – 7/7

Governance and Compliance

- Ensure the system adheres to ethical standards, data privacy laws, and organizational policies.

Example: Document how the churn model makes predictions and verify compliance with GDPR.

MLOps Principles

Reproducible

Re-executing the exact same training run should be possible at any time.

Accountable

All decisions should be recorded

Collaborative

Exploring the work of others and extending it should be possible

Continuous

Automated execution of tasks like building the source code or deploying the model

Scalable

The system should have greater granular elasticity based on demand

Trustworthy

Establish trust by adhering to the ML Principles

Key Roles in MLOps – 1/3

Data Scientists

Focus: Building machine learning models.

Skills: Data analysis, statistics, ML algorithms, Python/R, tools like Jupyter Notebook, TensorFlow, or Scikit-learn ...

Role in MLOps:

- Data exploration and preparation.
- Training and fine-tuning models.
- Handing off models to ML engineers for deployment.

Example Task: Train a recommendation system for an e-commerce site.

Key Roles in MLOps – 2/3

ML Engineers

Focus: Deploying and maintaining ML models in production.

Skills: Software engineering, APIs, Docker, Kubernetes, model serving frameworks (e.g., TensorFlow Serving, FastAPI) ...

Role in MLOps:

- Automate data pipelines, model training, and deployment.
- Optimize model inference for low latency and scalability.

Example Task: Build a pipeline that automates retraining of the recommendation system when new data arrives.

Key Roles in MLOps – 3/3

DevOps (or MLOps Engineers)

Focus: Infrastructure, CI/CD, and monitoring systems.

Skills: Cloud platforms (AWS, GCP, Azure), CI/CD tools (Jenkins, GitHub Actions), logging/monitoring tools (Prometheus, Grafana).

Role in MLOps:

- Set up infrastructure for model deployment and scaling.
- Automate workflows for continuous integration and delivery.
- Monitor production systems for failures or performance degradation.

Example Task: Configure a Kubernetes cluster to host the recommendation system and ensure it scales during peak traffic.

Tools and Technologies in MLOps

Category	Tools	Purpose	Example
Data Management	Apache Airflow, Prefect, Pachyderm, Kedro	Automates data pipelines, ensures data quality, and manages data lineage.	Use Apache Airflow to schedule daily data ingestion for a recommendation system.
Model Development	Jupyter, TensorFlow, PyTorch, Scikit-learn, Metaflow	Model training and experimentation.	Use PyTorch to train a computer vision model for object detection.
Version Control	Git (code), DVC (Data Version Control), MLflow	Track changes in code, datasets, and models.	Use DVC to version datasets and MLflow to track experiments.
Model Serving	TensorFlow Serving, TorchServe, FastAPI, Flask	Expose trained models via APIs for real-time or batch predictions.	Use FastAPI to serve the recommendation model as a REST API.
Containerization and Orchestration	Docker, Kubernetes, Kubeflow	Package and deploy models in scalable environments.	Use Docker to containerize a model and deploy it on a Kubernetes cluster.
CI/CD for ML	Jenkins, GitHub Actions, GitLab CI/CD, ArgoCD	Automate integration, testing, and deployment.	Use GitHub Actions to trigger model retraining when new code is pushed.
Monitoring and Logging	Prometheus, Grafana, ELK Stack, Seldon Core	Monitor model performance, detect drift, log predictions.	Use Grafana to visualize model latency and alert if it exceeds a threshold.
Cloud Platforms	AWS SageMaker, GCP Vertex AI, Azure ML	End-to-end MLOps solutions for model training, deployment, and monitoring.	Use AWS SageMaker to train and deploy an NLP model for sentiment analysis.

Amazon SageMaker · gradient by Paperspace · FLOYD · DOMINO DATA LAB

"All-in-one"

Data

- **Versioning**: Pachyderm, DVC, LIQUIDATA
- **Labeling**: figure eight, scale, Aquarium
- **Processing**: Spark, DAGSTER
- **Exploration**: pandas, RAPIDS, dbt
- **Data Lake / Warehouse**: snowflake, databricks
- **Sources**: S3, LOG, Parquet

Training/Evaluation

- **Frameworks & Distributed Training**: fast.ai, PYTORCH, RAY, K, HOROVOD
- **Hyperparameter Tuning**: SIGOPT, Determined AI, Weights & Biases, tune
- **Experiment Management**: Weights & Biases, comet, TensorBoard, Neptune, mlflow
- **Resource Management**: Determined AI, PY, slurm
- **Software Engineering**: jupyter, Streamlit, git
- **Compute**: CoreWeave, Lambda

Deployment

- **Feature Store**: tecton
- **Monitoring**: fiddler
- **Edge**: ML, NVIDIA TensorRT, TensorFlow Lite, ONNX
- **Web**: ALGORITHMIA, SELDON
- **CI / Testing**: Buildkite, great_expectations
- or · or