

ML Ops: Transforming Machine Learning Workflows

Let's explore ML Ops, its significance, and how it's revolutionizing machine learning.

We'll cover development methods, key frameworks, deployment challenges, and security. Discover how MLOps transforms traditional workflows.

By: Doug Ortiz Technologist



Doug Ortiz

About Me

- . LinkedIn Top Voice in Technology
- Co-Author and Contributor of renowned open-source PostgreSQL projects
- · Bring a powerful fusion of Cloud, Data, Databases, DevOps, AI, and ML expertise to drive digital transformation.
- My ability to deliver compelling presentations and facilitate clear communication across diverse audiences sets me apart in the industry.

Bio



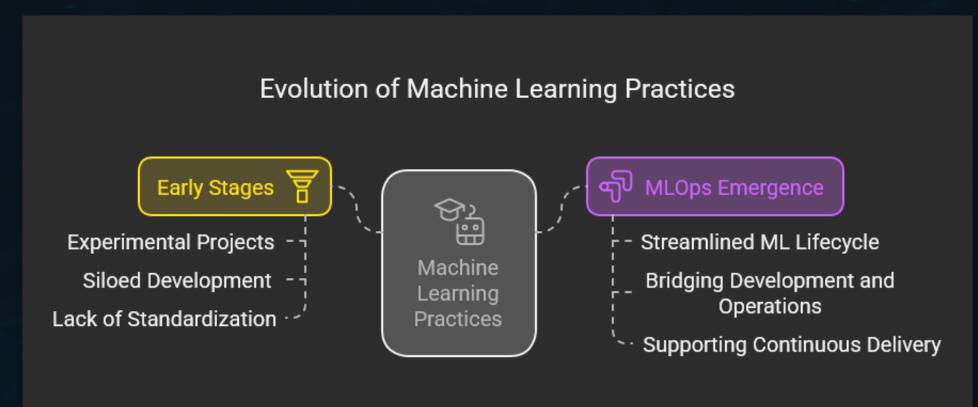
The Evolution of Machine Learning Development

Early Stages

Initial ML projects were often experimental and siloed. Development lacked standardized processes. Scaling was difficult.

Emergence of MLOps

MLOps emerged to streamline the ML lifecycle. It bridges the gap between development and operations. This supports continuous delivery.



Phases of Machine Learning Maturity

Phases of Machine Learning Maturity

Ad Hoc Scalable Optimized **Emerging** Operational Characterized by Includes end-to-end Features Involves basic Focuses on real-time manual workflows standardized automation and pipelines and initial and ML-driven and exploratory pipelines and CI/CD advanced version control capabilities integration monitoring projects

Traditional vs. MLOps-Driven Workflows

1

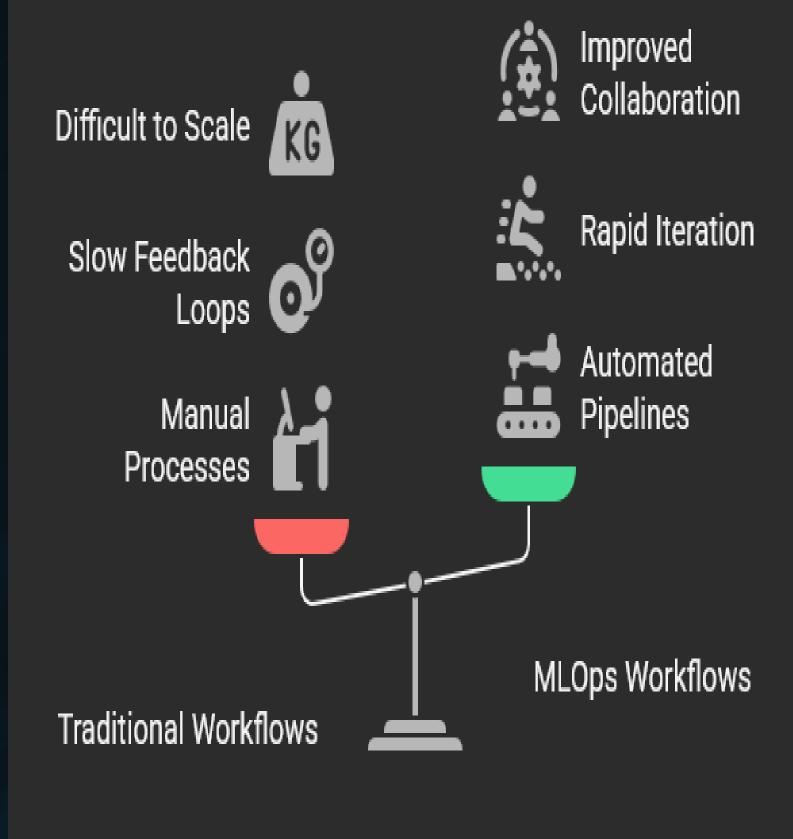
Traditional

Manual processes. Limited automation. Slow feedback loops. Difficult to maintain and scale models.

2

MLOps

Automated pipelines. Continuous integration/continuous delivery (CI/CD). Rapid iteration. Improved collaboration.



MLOps Enhances Workflow Efficiency



Leading ML Frameworks and Ecosystem

Machine Learning Frameworks



TensorFlow

An end-to-end opensource platform for machine learning.



PyTorch

Known for its flexibility, ideal for research and rapid prototyping.



Scikit-learn

Simple and efficient tools for data analysis and machine learning.



MLflow

A tool for experiment tracking in machine learning.

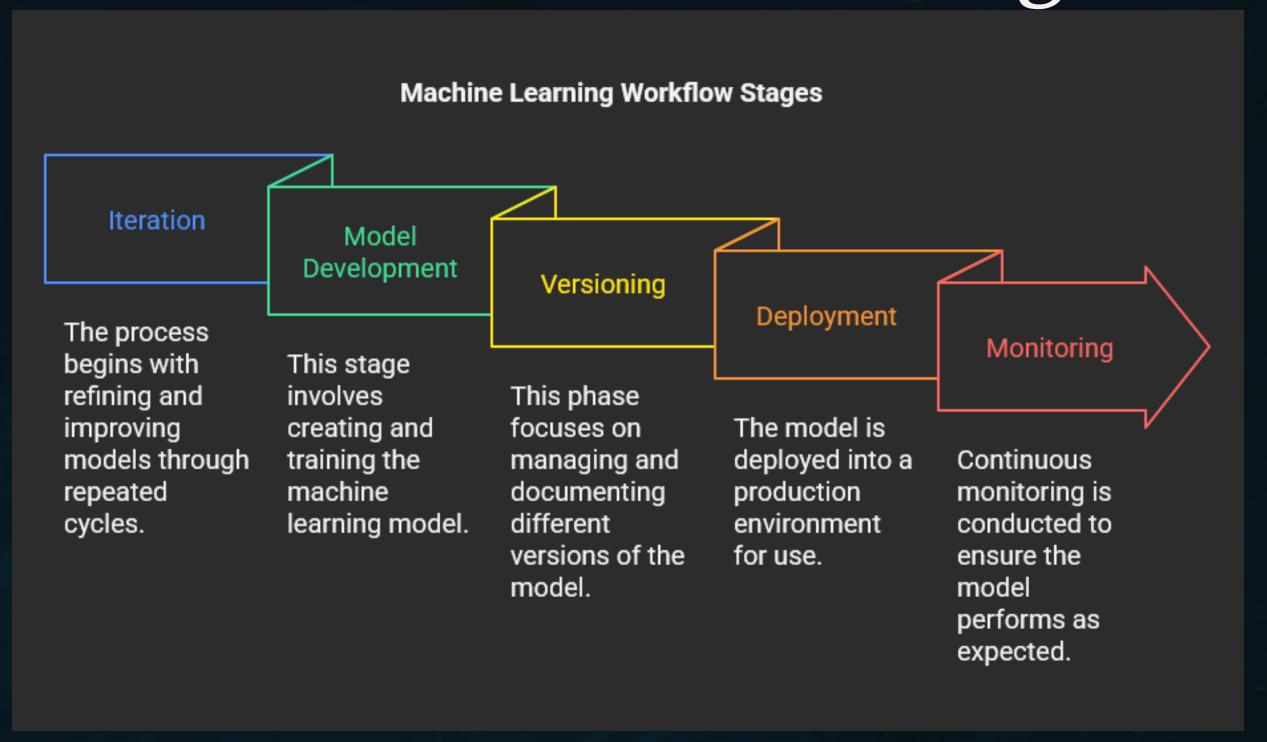


Kubeflow Pipelines

A platform for orchestration in machine learning workflows.



Jfrog Artifactory: Universal Artifact Repository for Model Versioning





Challenges in Model Deployment and Management

1

Model Versioning

Tracking changes and managing model iterations.

2

Reproducibility

Ensuring consistent results across environments.

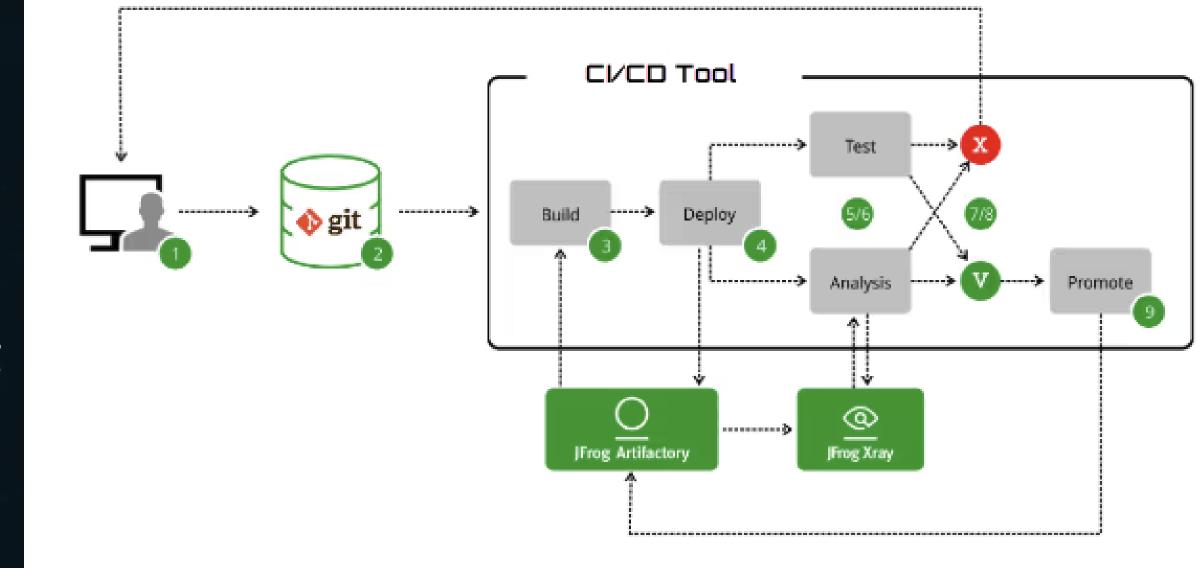
Monitoring Drift

Detecting changes in model performance over time.



Deployment Challenges

Model Versioning



Monitoring Drift

```
import pandas as pd
from evidently.dashboard import Dashboard
from evidently.tabs import DataDriftTab
reference_data = pd.DataFrame({
    'feature_1': [1, 2, 3, 4, 5],
    'feature_2': [10, 20, 30, 40, 50]
})
current_data = pd.DataFrame({
    'feature_1': [1, 2, 3, 6, 7], # Slight drift in feature_1
    'feature_2': [10, 20, 30, 40, 50]
3)
# Create a DataDrift dashboard
data_drift_dashboard = Dashboard(tabs=[DataDriftTab()])
data_drift_dashboard.calculate(reference_data, current_data, column_mapping=None)
# Save the dashboard as an HTML report
data_drift_dashboard.save("data_drift_report.html")
```

Why ML Breaks in Production

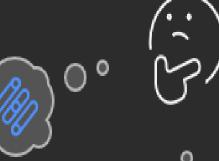
- Dependency Hell conda vs pip conflicts
- Monitoring Blindspots Undetected data drift
- Toolchain Complexity

Overwhelm from 10+ fragmented tools \rightarrow Standardize on platforms like MLFlow + Artifactory

How to address ML production challenges?



Use conda or pip to manage package dependencies effectively.







Detect Data Drift

Implement monitoring tools like Evidently.ai to identify data drift.



Standardize Toolchain

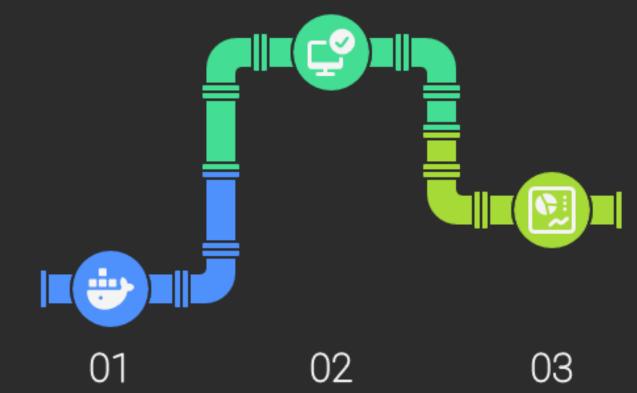
Consolidate tools using platforms like MLflow and Artifactory.

Ensuring Model Reproducibility and Monitoring

- 1 Containerization
 Use Docker to package
 models. Maintain
 consistent
 environments.
- Automated Testing
 Implement unit and
 integration tests.
 Confirm model accuracy.

Monitoring Tools

Employ tools to track model performance metrics. Detect data drift.



Containerization

Use Docker to package models and maintain consistent environments. Automated Testing

Implement unit and integration tests to confirm model accuracy. Monitoring Tools

Employ tools to track model performance metrics and detect data drift.



Addressing Security Risks in Machine Learning

Dependency Scanning

Identify vulnerabilities in open-source libraries.

Access Control

Limit access to sensitive data and models. Enforce strong authentication.

Regular Audits

Conduct security audits. Evaluate model integrity. **Include HIPAA.**



End-to-End Security



Native Support for All OSS Repositories



Source Code Repositories



Native Support for All CI/CD Tools





Data Sources



JFrog CURATION

Curate OSS packages and ML models entering your organization.



Resolve OSS security risks and compliance issues.



Advanced scanning of code, binaries and containers.



Fast discovery and remediation code to production.

JFrog ARTIFACTORY

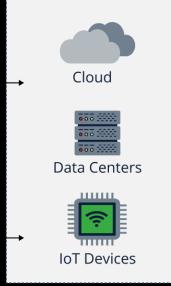
DevOps system of record, storing and managing all binary software packages, ML models & container images.



Accelerate releases with fast, trusted software distribution.



Manage connected IoT devices at scale.







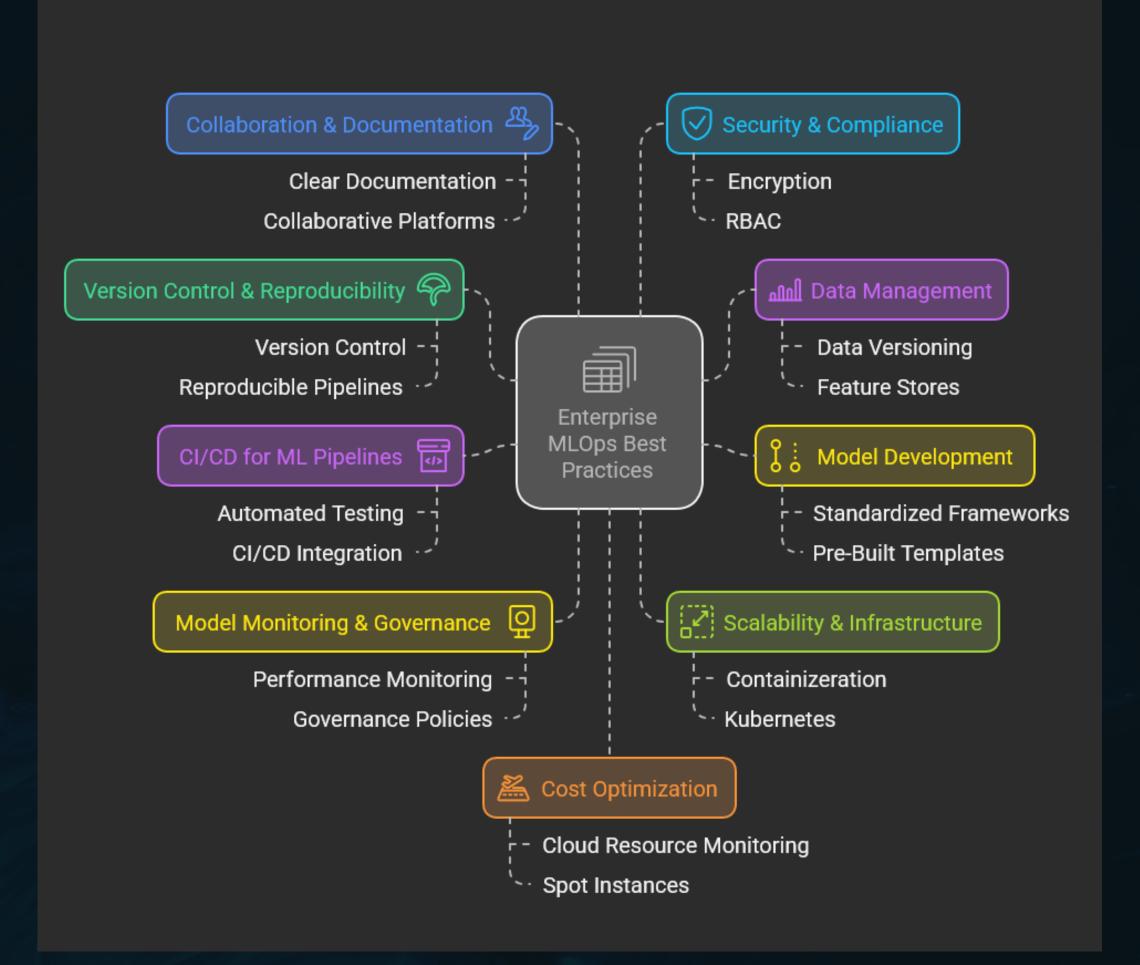
Build, secure, manage, monitor and deploy AI/ML models at scale to extend orchestration of your complete software supply chain to AI development.

Future of MLOps: Key Strategies and Best Practices



Embrace automation. Foster collaboration. Prioritize continuous monitoring. Implement robust governance. These strategies enhance MLOps.

Future Strategies: Enterprise MLOps Best Practices Checklist





Takeaways

Key Strategies to Implement Today

- 1-Automate your workflow
- Toolkit: MLFlow + GitHub Actions (CI/CD) + JFrog
 Artifactory (Model Registry)
- 2-Secure the ML Supply Chain
- Scan PyTorch/Tensorflow dependencies with JFrog Xray
- Generate SBOMs for compliance (GCPR/HIPAA)
- 3-Standardize Environments
- Use Docker/Kubernetes to eliminate "works on my machine" issues

Free Resources to Start Clone Our Tutorial!

MLOps Tutorial

Building a CI/CD Pipeline for MNIST Digit Classification





This repository provides a hands-on tutorial for building a CI/CD pipeline for MNIST digit classification using GitHub Actions, JFrog Artifactory, and JFrog Xray.

The pipeline includes:

- Training a PyTorch model.
- Storing the model in JFrog Artifactory.
- Generating a Software Bill of Materials (SBOM).
- Scanning for vulnerabilities using JFrog Xray.

Thank You



GitHub Repository





<u>LinkedIn</u>





YouTube Channel





Bio

