AEGIS IT

MLOPS BLUEPRINT

ML Pipeline Architecture

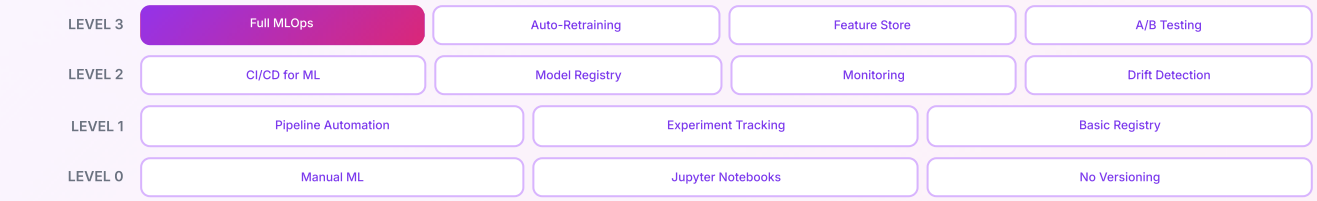
A comprehensive guide to building production-grade machine learning systems with MLOps best practices.

VERSION	PAGES	UPDATED
2.0	10	February 2025

01

MLOps Maturity Model

Understand where you are and where you need to go on your MLOps journey.



Level 0: Manual ML

Jupyter notebooks, manual deployment, no versioning. Suitable for experimentation only. High risk for production.

Level 1: Pipeline Automation

Automated training pipelines, experiment tracking, basic model registry. Production-ready for stable models.

Level 2: CI/CD for ML

Automated testing, continuous training, model monitoring. Enterprise-grade ML operations.

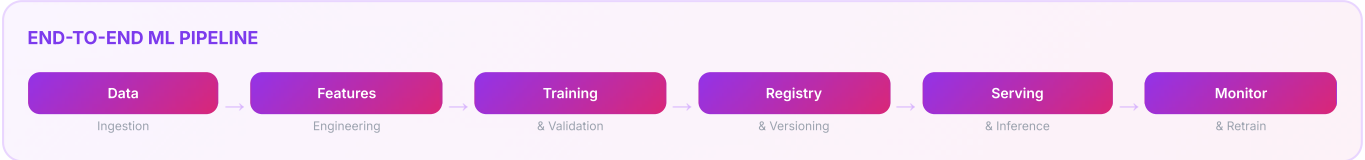
Level 3: Full MLOps

Feature stores, drift detection, auto-retraining, A/B testing. AI-native organization.

02

ML Pipeline Architecture

The core components that make up a production ML system.



Essential Pipeline Components

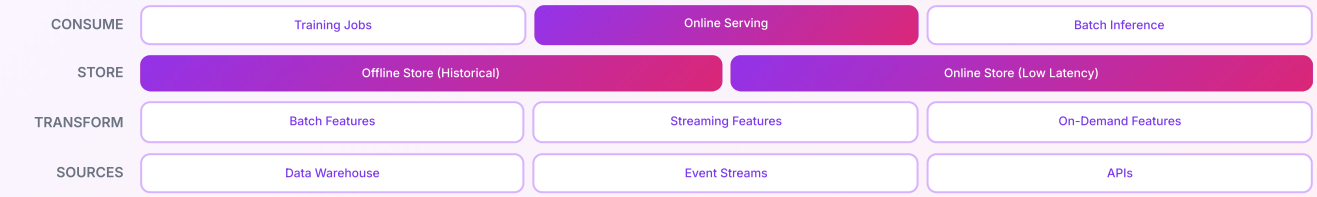
Component	Purpose	Tools
Feature Store	Centralized feature repository for training and serving	Feast, Tecton, Databricks FS
Experiment Tracking	Log parameters, metrics, artifacts for reproducibility	MLflow, W&B, Neptune
Model Registry	Version control for models with stage transitions	MLflow, SageMaker, Vertex AI
Model Serving	Low-latency inference APIs at scale	Seldon, KServe, Triton
Model Monitoring	Drift detection and performance tracking	Evidently, WhyLabs, Arize

```
project/
├── features/           # Feature definitions & pipelines
│   ├── feature_store.py
│   └── transformations.py
├── training/           # Model training code
│   ├── train.py
│   ├── evaluate.py
│   └── hyperparameter_tuning.py
├── serving/            # Inference endpoints
│   ├── model_server.py
│   └── preprocessing.py
├── monitoring/         # Drift & performance
│   └── drift_detection.py
└── pipelines/          # Orchestration DAGs
    └── training_pipeline.py
```

03

Feature Store Architecture

The feature store is the heart of MLOps—ensuring consistency between training and serving.



Feature Store Benefits

Training-Serving Consistency

Same feature transformations in training and production. Eliminates training-serving skew.

Feature Reuse

Share features across teams and models. Reduce duplicate feature engineering effort.

Point-in-Time Correctness

Retrieve features as of a specific timestamp. Prevent data leakage in training.

Feature Monitoring

Track feature distributions and detect drift. Alert on anomalies before they affect models.

Key Insight

Companies report 3x faster model development after implementing a feature store, with 50% reduction in feature engineering time across teams.

04

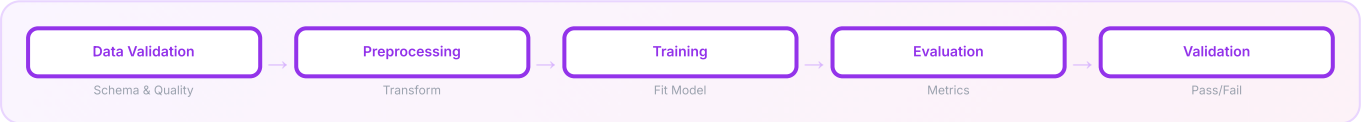
Model Training & Validation

Best practices for reproducible, scalable model training.

Experiment Tracking Requirements

What to Track	Why It Matters	Example
Parameters	Reproduce any experiment exactly	learning_rate=0.001, epochs=100
Metrics	Compare model performance	accuracy=0.95, f1=0.92
Artifacts	Store models and outputs	model.pkl, confusion_matrix.png
Code Version	Know exactly what code was run	git commit sha, branch
Data Version	Know what data was used	dataset hash, date range
Environment	Reproduce compute environment	requirements.txt, Docker image

Training Pipeline Stages



Model Validation Gates

- [ ] Performance meets minimum threshold (e.g., accuracy > 0.90)
- [ ] Performance improves over current production model
- [ ] No significant performance degradation on any segment
- [ ] Inference latency within SLA (e.g., p99 < 100ms)
- [ ] Model size within deployment constraints
- [ ] Bias and fairness metrics within acceptable range

05

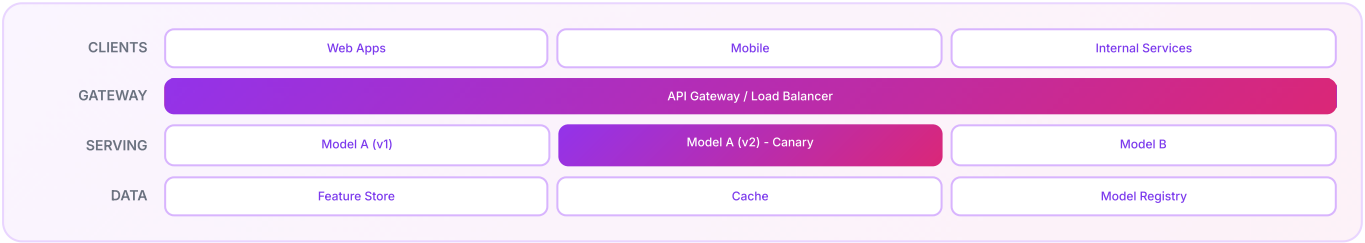
Model Serving Patterns

Deploy models for real-time and batch inference at scale.

Serving Patterns Comparison

Pattern	Latency	Use Case	Infrastructure
Real-time API	< 100ms	User-facing predictions	K8s, Serverless
Batch Inference	Minutes-Hours	Bulk scoring, reports	Spark, Airflow
Streaming	< 1s	Event-driven predictions	Kafka, Flink
Edge/Embedded	< 10ms	IoT, mobile devices	TFLite, ONNX

Model Serving Architecture



< 50ms
p50 Latency Target

< 100ms
p99 Latency Target

99.9%
Availability SLA

10K+
Requests/Second

06

Model Monitoring & Drift Detection

Production models degrade over time. Monitoring catches issues before they impact users.

Types of Drift

Data Drift

Input feature distributions change over time. Detected by comparing production data to training data statistics.

Concept Drift

Relationship between features and target changes. Model predictions become less accurate even with same features.

Prediction Drift

Model output distribution changes. May indicate data or concept drift affecting predictions.

Label Drift

Ground truth distribution changes. Common in dynamic environments like fraud detection.

Key Monitoring Metrics

Category	Metrics	Alert Threshold
Model Performance	Accuracy, Precision, Recall, AUC	> 5% degradation from baseline
Data Quality	Missing values, outliers, schema violations	> 1% anomalies
Feature Drift	PSI, KL divergence, JS distance	PSI > 0.2
Prediction Distribution	Mean, std, percentiles of outputs	> 2 std from baseline
Operational	Latency, throughput, error rate	p99 > SLA, errors > 0.1%

Retraining Triggers

Automated retraining should be triggered by: (1) Scheduled intervals, (2) Performance degradation, (3) Significant data drift, or (4) New labeled data availability. Most mature MLOps teams use a combination of all four.

07

Technology Stack

Recommended tools for each component of the ML platform.

Component	Open Source	AWS	Azure	GCP
Experiment Tracking	MLflow, W&B	SageMaker	Azure ML	Vertex AI
Feature Store	Feast, Hopsworks	SageMaker FS	Azure ML FS	Vertex FS
Model Registry	MLflow	SageMaker	Azure ML	Vertex AI
Training	Kubeflow	SageMaker	Azure ML	Vertex AI
Serving	Seldon, KServe	SageMaker	Azure ML	Vertex AI
Monitoring	Evidently, WhyLabs	SageMaker MM	Azure ML	Vertex AI
Orchestration	Airflow, Kubeflow	Step Functions	Data Factory	Composer

Recommended Stack by Stage

Starting Out

MLflow + Airflow
Simple, proven, flexible. Works with any cloud.

Scaling Up

Databricks MLflow + Feast + Seldon
Unified platform with enterprise features.

Enterprise

Cloud-native (SageMaker/Vertex) + Tecton
Managed services, minimal ops overhead.

Regulated

Self-hosted + Fiddler/Arize
Full control, audit trails, explainability.

08

Implementation Checklist

Use this checklist to track your MLOps implementation progress.

Level 1: Foundation

- [ ] Version control for all ML code (Git)
- [ ] Experiment tracking system deployed (MLflow/W&B)
- [ ] Reproducible training environment (Docker/Conda)
- [ ] Basic model registry with versioning
- [ ] Automated training pipeline (Airflow/Kubeflow)

Level 2: Production Ready

- [ ] CI/CD pipeline for model training and deployment
- [ ] Automated model validation gates
- [ ] Model serving infrastructure with autoscaling
- [ ] Basic monitoring (latency, throughput, errors)
- [ ] Rollback capabilities tested

Level 3: Enterprise MLOps

- [ ] Feature store with online/offline serving
- [ ] Data and model drift monitoring
- [ ] Automated retraining triggers
- [ ] A/B testing infrastructure
- [ ] Model explainability and fairness monitoring
- [ ] Complete audit trail for compliance

Ready to Build Your ML Platform?

Our ML engineering experts can help you design and implement production-grade MLOps infrastructure. Schedule a free consultation to discuss your AI initiatives.

aegisit.ai/contact

(404) 490-0234 | info@aegisit.ai