MLOps – Best Practices and CI/CD

Karthikeyan Vaiyapuri

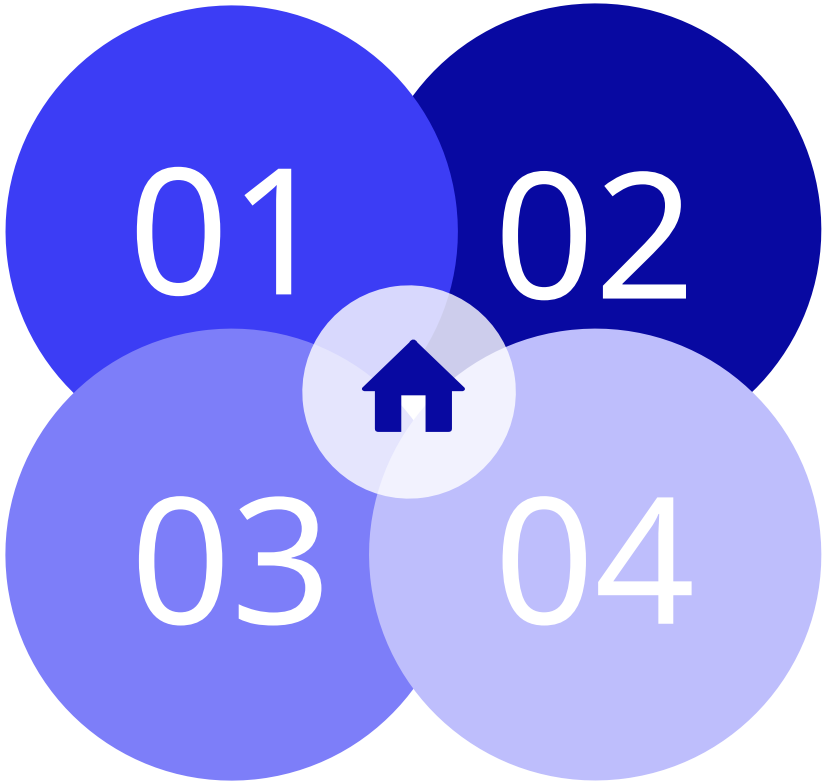
01

Overview

Agenda

MLOps Best Practices - Production-ready pipeline design

Model Lifecycle Management - From training to retirement



Simple CI/CD Pipelines - Automated testing and deployment

Mini Project - Complete end-to-end MLOps system

02

MLOps Best Practices

Core Principles

Automation First - Manual processes don't scale

Fail Fast, Recover Faster - Build resilient systems

Version Everything - Code, data, models, environments

Reproducibility - Same inputs = same outputs

Monitor Continuously - Performance, quality, costs

The MLOps Maturity Model

Level	Characteristics	Example
Level 0	Manual, script-driven	Jupyter notebooks
Level 1	ML pipeline automation	Automated training
Level 2	Ci/CD ML pipeline	Automated testing & deployment
Level 3	Full MLOps Automation	Self-healing systems

Success Metrics

Deployment Frequency - How often models are released

Lead Time - Idea to production deployment

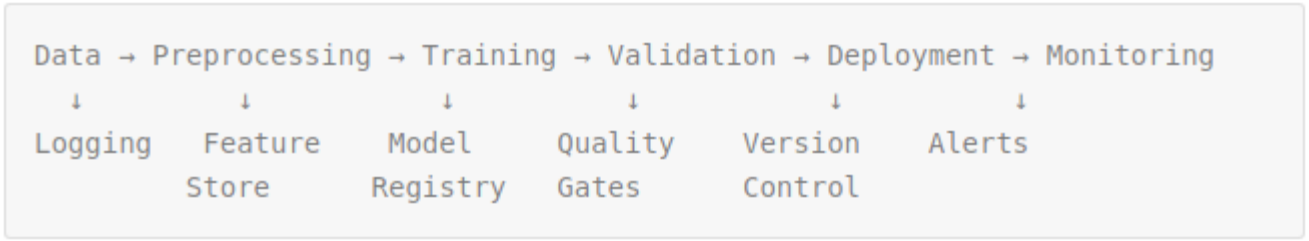
Mean Time to Recovery - Fix time for issues

Change Failure Rate - Percentage of failed deployments

03

Building Robust and Scalable MLOps Pipelines

Pipeline Architecture Components



Robustness Best Practices

01

Error Handling - Graceful failure and recovery

02

Data Validation - Schema and quality checks

03

Model Validation - Performance thresholds

04

Rollback Mechanisms - Quick reversion to previous versions

05

Health Checks - Continuous system monitoring

Scalability Patterns

01

Microservices Architecture - Independent, scalable components

02

Containerization - Docker for consistent environments

03

Load Balancing - Distribute traffic across instances

04

Horizontal Scaling - Add more instances as needed

05

Caching Strategies - Reduce computational overhead

04

Managing Model Lifecycles and Deployments

Model Lifecycle Stages

01

Development - Experimentation and training

02

Staging - Pre-production testing

03

Production - Live serving

04

Monitoring - Performance tracking

05

Retraining - Model updates

06

Retirement - End of lifecycle

Deployment Strategies

Strategy	Description	Use Case
Blue-Green	Two identical environments	Zero-downtime deployment
Canary	Gradual roolout to subset	Risk mitigation
A/B Testing	Compare model versions	Perforamce optimization
Shadow	Run alongside existing	Safe testing

Model Governance Framework

01

Model Registry - Centralized model storage

02

Approval Workflows - Controlled promotion process

03

Audit Trails - Complete change history

04

Compliance Tracking - Regulatory requirements

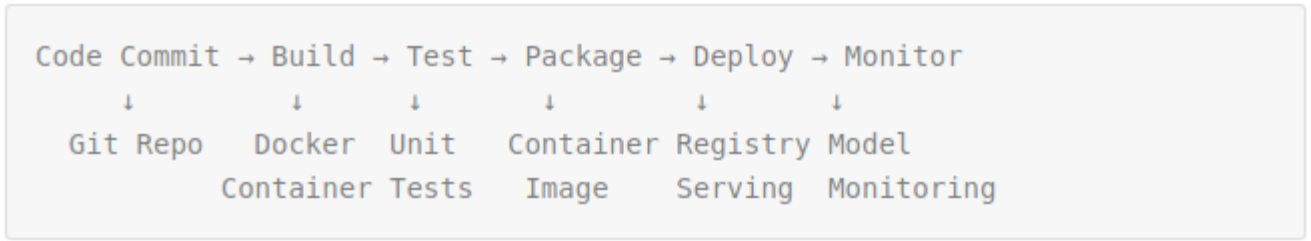
05

Performance SLAs - Service level agreements

05

Building Basic CI/CD Pipeline for ML Models

CI/CD Components for ML



Continuous Integration (CI)

Code Quality - Linting, formatting, complexity checks

Unit Tests - Individual component testing

Integration Tests - Component interaction testing

Data Tests - Schema and quality validation

Model Tests - Performance and bias testing

Continuous Deployment (CD)

01.

Environment Promotion - Dev → Staging → Production

02.

Automated Deployment - Infrastructure as Code

03.

Rolling Updates - Gradual deployment

04.

Monitoring Integration - Automatic health checks

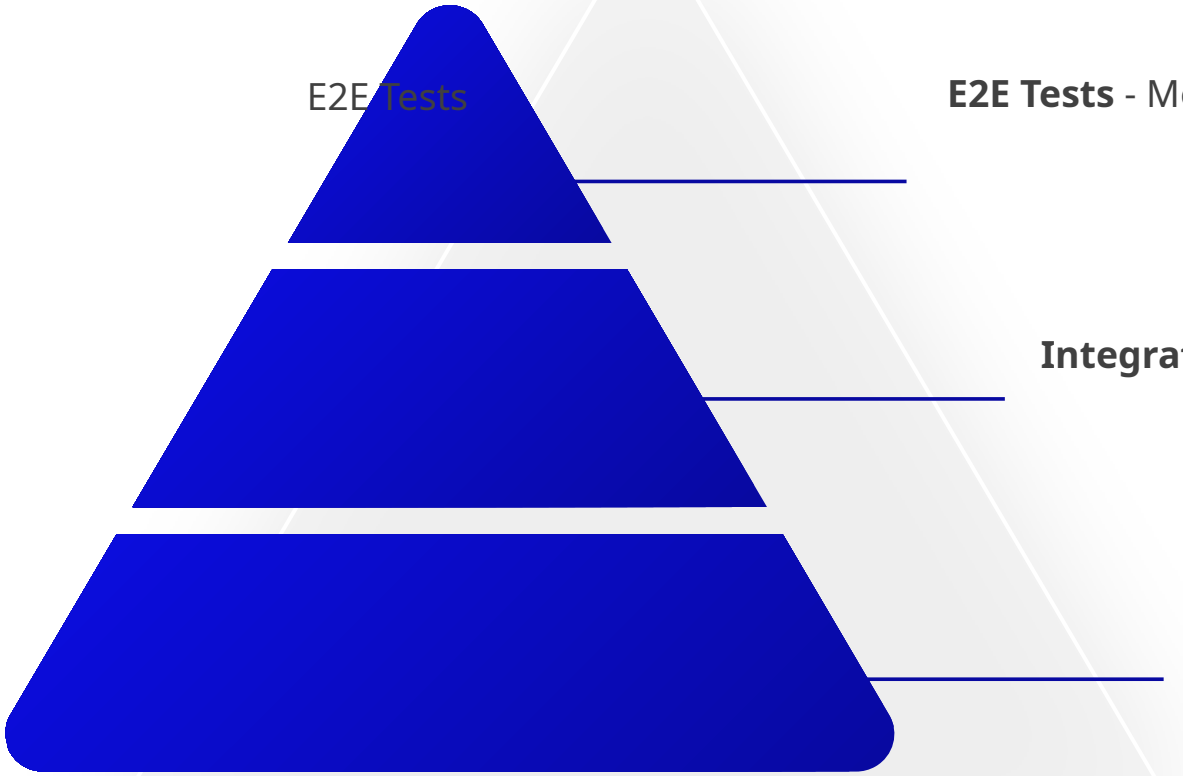
05.

Rollback Capabilities - Quick failure recovery

06

Automating Testing and Deployment

ML-Specific Testing Pyramid



E2E Tests - Model integration (Few, Slow)

Integration Tests - Data pipeline (Some, Medium)

Unit Tests - Individual functions (Many, Fast)

Automated Testing Categories

01 **Data Tests** - Schema, distribution, quality

02 **Model Tests** - Accuracy, bias, performance

03 **Infrastructure Tests** - Scaling, availability

04 **Security Tests** - Vulnerability scanning

05 **Performance Tests** - Latency, throughput

Deployment Automation Tools

01.

GitHub Actions - Code- triggered workflows

02.

Jenkins - Flexible automation server

03.

GitLab CI - Integrated DevOps platform

04.

Azure DevOps - Microsoft ecosystem

05.

Docker/Kubernetes - Container orchestration

Thanks

Karthikeyan Vaiyapuri