Uber Unified ML Platform - Production MVP

Architecture Overview

This is a production-ready MVP implementation of Uber's Unified Enterprise ML Platform, designed to handle the full ML lifecycle from data ingestion to model deployment and monitoring.

Project Structure

```
uber-ml-platform/
├── infrastructure/        # Terraform & K8s manifests
├── data_platform/         # Kafka, Airflow, Spark pipelines
├── feature_store/         # Feast feature definitions & serving
├── training/              # Distributed training with Ray & MLflow
├── serving/               # FastAPI inference services
├── monitoring/            # Prometheus, Grafana, drift detection
├── governance/            # Model registry, audit trails, compliance
├── common/                # Shared utilities and configurations
└── tests/                 # Unit and integration tests
```

Key Components

1. Data Platform

- **Kafka**: Real-time event streaming (rider events, driver pings)
- **Airflow**: Orchestration of batch pipelines
- **Spark/dbt**: Data transformation at scale

- **Great Expectations**: Automated data quality validation

2. Feature Store (Feast)

- Centralized feature definitions
- Online/offline feature consistency
- Redis for low-latency online serving
- PostgreSQL for offline feature storage

3. Training Platform

- **Ray**: Distributed training orchestration
- **MLflow**: Experiment tracking & model registry
- Automated hyperparameter optimization
- Continuous retraining triggers

4. Serving Layer

- FastAPI-based inference service
- Kubernetes autoscaling (HPA)
- Blue-green & canary deployments
- Redis caching for performance

5. Monitoring & Governance

- Real-time model performance tracking
- Drift detection with Evidently AI
- Automated audit trails for compliance
- Role-based access control

Getting Started

Prerequisites

- Docker & Docker Compose
- Kubernetes cluster (EKS/GKE) or Minikube for local

- Python 3.9+
- Terraform (for infrastructure)

Quick Start

```
# Install dependencies
pip install -r requirements.txt

# Start local development environment
docker-compose up -d

# Initialize feature store
cd feature_store && feast apply

# Run a sample training job
python training/train_eta_model.py

# Start inference service
python serving/main.py
```

Environment Variables

Copy `.env.example` to `.env` and configure:

- Database credentials
- Kafka broker addresses
- Redis connection
- MLflow tracking URI
- AWS/GCP credentials

Deployment

```
# Deploy to Kubernetes
kubectl apply -f infrastructure/k8s/

# Apply Terraform infrastructure
cd infrastructure/terraform && terraform apply
```

Monitoring

Access dashboards:

- MLflow UI: http://localhost:5000
- Grafana: http://localhost:3000
- Airflow: http://localhost:8080