Fox ML Infrastructure — System Architecture Diagram

This document provides a visual and textual representation of the Fox ML Infrastructure system architecture.
This architecture diagram is essential for enterprise technical reviews and integration planning.

---

1. Executive Summary

Fox ML Infrastructure is a client-hosted, modular ML research infrastructure platform.

Key architectural principles: - **Client-hosted** — Software runs entirely on client infrastructure - **Modular design** — Clear separation of concerns across modules - **Configuration-driven** — All runtime parameters from centralized configs - **Pipeline-based** — Data flows through well-defined pipeline stages - **Extensible** — Easy to add new models, features, and strategies

---

2. High-Level Architecture

2.1 System Overview

```
+-----------------------------------------------------------------+
|                    CLIENT INFRASTRUCTURE                         |
|                    (Client-Controlled)                          |
+-----------------------------------------------------------------+
                              |
                              |
        +---------------------+---------------------+
        |                                             |
        v                                             v
+------------------+                      +------------------+
|  DATA ACQUISITION|                      |  DEPLOYMENT &    |
|  & INGESTION     |                      |  EXECUTION       |
|                  |                      |                  |
|  • Market Data   |                      |  • Model Serving |
|  • Data Sources  |                      |  • Trading       |
|  • Normalization |                      |  • Monitoring    |
+--------+---------+                      +--------+---------+
         |                                          |
         |                                          |
         v                                          v
+--------------------------------------------------------------+
|              FOX ML INFRASTRUCTURE PLATFORM                   |
|              (Client-Hosted Software)                          |
|--------------------------------------------------------------|
|                                                               |
|  +-----------------------------------------------------+   |
|  |         DATA PROCESSING LAYER                         |   |
|  |  +--------------+  +--------------+  +----------+  |   |
|  |  |   Features   |  |   Targets    |  | Pipeline |  |   |
|  |  |  Engineering |-> |  Generation  |-> | Workflow |  |   |
|  |  +--------------+  +--------------+  +----------+  |   |
|  +-----------------------------------------------------+   |
|                          |                                   |
|                          v                                   |
|  +-----------------------------------------------------+   |
|  |         MODEL TRAINING LAYER                         |   |
|  |  +--------------+  +--------------+  +----------+  |   |
|  |  |   Training   |  |  Validation  |  |  Model   |  |   |
|  |  |  Strategies  |-> |  (Walk-      |-> |  Storage |  |   |
|  |  |              |  |  Forward)    |  |          |  |   |
|  |  +--------------+  +--------------+  +----------+  |   |
|  +-----------------------------------------------------+   |
|                          |                                   |
|                          v                                   |
|  +-----------------------------------------------------+   |
|  |         CONFIGURATION LAYER                         |   |
|  |  +--------------+  +--------------+  +----------+  |   |
|  |  |   Model      |  |  Training    |  | Runtime  |  |   |
|  |  |   Configs    |  |  Configs     |  | Overrides|  |   |
|  |  +--------------+  +--------------+  +----------+  |   |
|  +-----------------------------------------------------+   |
|                                                               |
+---------------------------------------------------------------+
```

---

3. Component Architecture

3.1 Data Processing Layer

Purpose: Transform raw market data into ML-ready features and targets.

Components:

```
DATA_PROCESSING/
|-- features/
|   |-- SimpleFeatureComputer      # Basic technical indicators
|   |-- ComprehensiveFeatureBuilder # 200+ feature engineering
|   +-- StreamingFeatureBuilder    # Streaming/online features
|-- targets/
|   |-- BarrierTargetBuilder       # Barrier-based targets
|   |-- ExcessReturnsBuilder      # Excess return targets
|   +-- HFTForwardReturnsBuilder  # High-frequency targets
|-- pipeline/
|   |-- DataNormalization          # RTH alignment, grid correction
|   |-- FeaturePipeline           # End-to-end feature workflow
|   +-- TargetPipeline            # End-to-end target workflow
```

```
+-- utils/
    |-- MemoryManagement          # Efficient memory handling
    |-- Logging                   # Structured logging
    +-- Validation                # Data sanity checks
```

Data Flow:

```
Raw Market Data
    v
Normalization (RTH alignment, grid correction)
    v
Feature Engineering (200+ technical features)
    v
Target Generation (barrier, excess returns, HFT)
    v
Labeled Dataset (features + targets)
```

3.2 Model Training Layer

Purpose: Train and validate ML models using walk-forward validation.

Components:

```
TRAINING/
|-- model_fun/
|   |-- LightGBMTrainer           # Gradient boosting
|   |-- XGBoostTrainer            # Gradient boosting
|   |-- MLPTrainer                # Deep learning
|   |-- TransformerTrainer        # Transformer models
|   |-- EnsembleTrainer           # Model ensembles
|   |-- MultiTaskTrainer          # Multi-task learning
|   +-- [13+ additional trainers] # Probabilistic, advanced models
|-- strategies/
|   |-- SingleTaskStrategy        # One model per target
|   |-- MultiTaskStrategy         # Shared model for targets
|   +-- CascadeStrategy           # Sequential dependencies
|-- walkforward/
|   |-- WalkForwardValidator      # Time-series validation
|   +-- PurgedTimeSeriesSplit     # Leakage-safe splitting
+-- memory/
    |-- MemoryManager             # Memory optimization
    +-- BatchProcessing           # Efficient batch handling
```

Training Flow:

```
Labeled Dataset
    v
Walk-Forward Validation (time-series split)
    v
```

```
Model Training (17+ model types)
    v
Model Validation (performance metrics)
    v
Trained Models (serialized)
```

3.3 Configuration Layer

Purpose: Centralized, version-controlled configuration management.

Components:

```
CONFIG/
|-- model_config/
|   |-- lightgbm.yaml             # LightGBM configs (3 variants)
|   |-- xgboost.yaml              # XGBoost configs (3 variants)
|   |-- mlp.yaml                  # MLP configs (3 variants)
|   +-- [14+ additional configs]  # All model types
|-- training_config/
|   |-- walkforward.yaml          # Walk-forward parameters
|   |-- feature_selection.yaml    # Feature selection configs
|   +-- first_batch_specs.yaml    # Training specifications
+-- config_loader.py
    |-- load_model_config()       # Load model configs
    |-- list_available_configs()  # List available configs
    +-- apply_overrides()         # Runtime overrides
```

Configuration Flow:

```
Base Config (YAML)
    v
Variant Selection (conservative/balanced/aggressive)
    v
Runtime Overrides (environment variables, CLI args)
    v
Final Configuration (validated, applied)
```

3.4 Trading Integration Layer (Optional)

Purpose: Integration with trading brokers for paper and live trading.

Components:

```
IBKR_trading/                     # Interactive Brokers (Production)
|-- live_trading/
|   |-- MultiHorizonBlender       # 5m, 10m, 15m, 30m, 60m blending
|   |-- SafetyGuards              # Margin, short-sale, rate limiting
|   |-- CostAwareArbitration      # Cost-aware decision making
|   +-- C++InferenceEngine        # High-performance inference
+-- paper_trading/
    +-- PaperTradingSimulator     # Paper trading simulation

ALPACA_trading/                   # Alpaca (Paper Only)
|-- paper/
|   |-- RegimeAwareEnsemble      # Regime-aware strategies
|   |-- RiskManagement           # Risk management
|   +-- PerformanceTracking      # Performance tracking
+-- ml/
    +-- ModelIntegration          # ML model integration
```

Trading Flow:

```
Trained Models
    v
Model Inference (real-time predictions)
    v
Signal Generation (trading signals)
    v
Risk Management (safety guards, position sizing)
    v
Order Execution (broker integration)
    v
Performance Tracking (PnL, metrics)
```

---

4. Data Flow Architecture

4.1 End-to-End Pipeline

```
+--------------------------------------------------------------+
|                    STAGE 1: DATA INGESTION                   |
|                                                               |
|  Raw Market Data (OHLCV)                                      |
|  • yfinance, custom data sources                              |
|  • Multiple timeframes (1m, 5m, 15m, 1h, 1d)                 |
|  • Multiple symbols (SPY, QQQ, IWM, etc.)                     |
+----------------------+---------------------------------------+
                       |
                       v
+--------------------------------------------------------------+
|              STAGE 2: DATA PROCESSING                         |
|                                                               |
|  +--------------+    +--------------+    +--------------+  |
|  | Normalization| ->  |   Feature    | ->  |    Target    |  |
|  |              |    |  Engineering |    |  Generation  |  |
|  | • RTH align  |    |              |    |              |  |
|  | • Grid corr  |    | • 200+ feat  |    | • Barrier    |  |
|  | • Clean data |    | • Technical  |    | • Excess ret |  |
|  +--------------+    +--------------+    +--------------+  |
+----------------------+--------------------------------------+
                       |
                       v
+-------------------------------------------------------------+
|             STAGE 3: MODEL TRAINING                          |
|                                                               |
|  +--------------+    +--------------+    +--------------+  |
|  | Walk-Forward | ->  |    Model     | ->  |   Model      |  |
|  |  Validation  |    |   Training   |    |   Storage    |  |
|  |              |    |              |    |              |  |
|  | • Time-split |    | • 17+ models |    | • Serialized |  |
|  | • Purged     |    | • Strategies |    | • Versioned  |  |
|  +--------------+    +--------------+    +--------------+  |
+----------------------+--------------------------------------+
                       |
                       v
+-------------------------------------------------------------+
|             STAGE 4: DEPLOYMENT (Optional)                   |
|                                                               |
|  +--------------+    +--------------+    +--------------+  |
|  |   Model      | ->  |    Signal    | ->  |    Order     |  |
|  |  Inference   |    |  Generation  |    |  Execution   |  |
|  |              |    |              |    |              |  |
|  | • Real-time  |    | • Blending   |    | • IBKR/Alpaca|  |
|  | • C++ engine |    | • Risk mgmt  |    | • Monitoring |  |
|  +--------------+    +--------------+    +--------------+  |
+-------------------------------------------------------------+
```

5. Module Dependencies

5.1 Dependency Graph

```
CONFIG (Configuration Management)
    ^
    | (used by)
    |
DATA_PROCESSING (Feature Engineering)
    ^
    | (produces)
    |
TRAINING (Model Training)
    ^
    | (uses)
    |
CONFIG (Model Configs)
```

```
    ^
    | (optional)
    |
IBKR_trading / ALPACA_trading (Trading Integration)
```

5.2 External Dependencies

Core Dependencies: - **Python 3.9+** — Core runtime - **NumPy, Pandas** — Data processing - **Scikit-learn** — ML utilities - **LightGBM, XGBoost** — Gradient boosting models - **PyTorch** — Deep learning models (optional)

Trading Dependencies: - **ib_insync** — Interactive Brokers API - **alpaca-trade-api** — Alpaca API

Infrastructure: - **Git** — Version control - **GitHub** — Repository hosting - **YAML** — Configuration format

---

6. Deployment Architecture

6.1 Client-Hosted Deployment

```
+-------------------------------------------------------------+
|                    CLIENT INFRASTRUCTURE                     |
|                                                              |
|  +-------------------+   +-------------------+              |
|  |   Development     |   |    Production     |              |
|  |   Environment     |   |    Environment    |              |
|  |                   |   |                   |              |
|  | • Code repo       |   | • Model serving   |              |
|  | • Training        |   | • Trading      |              |
|  | • Testing         |   | • Monitoring    |              |
|  +-------------------+   +-------------------+              |
|                                                              |
|  +-------------------+   +-------------------+              |
|  |   Data Storage    |   |   Model Storage   |              |
|  |                   |   |                   |              |
|  | • Raw data        |   | • Trained models  |              |
|  | • Processed data  |   | • Configs         |              |
|  | • Features        |   | • Results         |              |
|  +-------------------+   +-------------------+              |
|                                                              |
|  +-------------------+   +-------------------+              |
|  |   Compute         |   |   Networking      |              |
|  |                   |   |                   |              |
|  • CPU/GPU           |   | • Market data     |              |
|  • Memory            |   | • Broker APIs     |              |
|  • Storage           |   | • Internal        |              |
|  +-------------------+   +-------------------+              |
+-------------------------------------------------------------+
```

6.2 No Vendor Infrastructure

Important: Fox ML Infrastructure does not operate: - [X] Vendor-hosted servers - [X] Vendor-hosted databases - [X] Vendor cloud services - [X] Vendor network infrastructure

All infrastructure is client-controlled and client-managed.

---

7. Security Architecture

7.1 Security Boundaries

```
+-------------------------------------------------------------+
|                    VENDOR BOUNDARY                            |
|                                                               |
|  • Code repositories (GitHub)                                |
|  • Support communications (Email)                            |
|  • Consulting data (Client-approved storage)                 |
+-------------------------------------------------------------+
                             |
                             | (Code Delivery)
                             |
                             v
+-------------------------------------------------------------+
|                    CLIENT BOUNDARY                            |
|                                                               |
|  • Software execution                                         |
|  • Data processing                                            |
|  • Model training                                             |
|  • Trading operations                                         |
|  • All client data                                            |
+-------------------------------------------------------------+
```

7.2 Security Controls

Vendor-side: - Repository access controls (2FA, SSH keys) - Code integrity (version control, signed commits) - Supply chain integrity (no telemetry, explicit dependencies)

Client-side: - Infrastructure security (client-managed) - Data security (client-managed) - Network security (client-managed) - Access control (client-managed)

---

8. Scalability Architecture

8.1 Horizontal Scalability

Scalable components: - **Data processing** — Parallel processing of multiple symbols/timeframes - **Model training** — Distributed training (if configured) - **Feature engineering** — Batch processing for large datasets - **Inference** — Parallel inference for multiple models

8.2 Vertical Scalability

Resource requirements: - **CPU** — Multi-core processors (8+ cores recommended) - **RAM** — 16GB+ (32GB+ for large datasets) - **GPU** — Optional but recommended for deep learning - **Storage** — Sufficient for datasets and models

---

9. Integration Points

9.1 Data Integration

Data sources: - **yfinance** — Equity market data (default) - **Custom adapters** — Client-specific data sources - **File-based** — CSV, Parquet, HDF5 files

9.2 Model Integration

Model deployment: - **Serialized models** — Pickle, joblib, PyTorch formats - **C++ inference** — High-performance C++ inference engine (IBKR) - **REST API** — Model serving via REST API (if configured)

9.3 Trading Integration

Broker integrations: - **Interactive Brokers** — Production trading (multi-horizon, C++ engine) - **Alpaca** — Paper trading (regime-aware ensemble)

---

10. Configuration Architecture

10.1 Configuration Hierarchy

```
Base Config (YAML)
    v
Variant Selection (conservative/balanced/aggressive)
    v
Environment Overrides (environment variables)
    v
Runtime Overrides (CLI arguments)
    v
Final Configuration (validated, applied)
```

10.2 Configuration Sources

Configuration sources: - **YAML files** — `CONFIG/model_config/*.yaml` - **Environment variables** — Runtime overrides - **CLI arguments** — Command-line overrides - **Code defaults** — Fallback defaults

---

11. Monitoring and Observability

11.1 Logging

Logging components: - **Structured logging** — JSON-structured logs - **Log levels** — DEBUG, INFO, WARNING, ERROR - **Contextual information** — Run IDs, symbols, fold numbers - **Client-managed** — Logs stored and managed by client

11.2 Metrics

Metrics tracked: - **Performance metrics** — Training time, inference latency - **Model metrics** — Accuracy, loss, validation scores - **Pipeline metrics** — Data processing throughput - **Client-managed** — Metrics collected and stored by client

---

12. Contact

For architecture questions or integration planning:

Jennifer Lewis
Fox ML Infrastructure LLC
Email: **jenn.lewis5789@gmail.com**
Subject: *Architecture Inquiry — Fox ML Infrastructure*

---

13. Related Documents

- `docs/00_executive/ARCHITECTURE_OVERVIEW.md` — Detailed architecture overview
- `docs/03_technical/design/ARCHITECTURE_DEEP_DIVE.md` — Deep technical dive
- `LEGAL/ENTERPRISE_DELIVERY.md` — Repository structure and delivery model
- `LEGAL/CLIENT_ONBOARDING.md` — Client onboarding and integration guide

---

14. Summary

Key Architectural Principles:

1. **Client-hosted** — Software runs entirely on client infrastructure
2. **Modular** — Clear separation of concerns across modules
3. **Configuration-driven** — All runtime parameters from configs
4. **Pipeline-based** — Data flows through well-defined stages
5. **Extensible** — Easy to add new models, features, strategies
6. **Scalable** — Horizontal and vertical scalability
7. **Secure** — Security boundaries and controls clearly defined

This architecture provides a comprehensive foundation for ML research and quantitative workflows.