

# Enterprise Delivery Model

This document describes how Fox ML Infrastructure is delivered to commercial licensees and consulting clients.

---

## Overview

Fox ML Infrastructure uses a **three-tier repository structure** that balances open-source transparency with commercial flexibility and client-specific customization:

1. **Public OSS Core** — Open-source foundation (AGPL-3.0)
2. **Enterprise Base Repository** — Private commercial-licensed foundation
3. **Per-Client Overlay Repositories** — Private client-specific customizations

This model enables: - **Marketing & validation** through the public OSS core - **Scalable bug fixes** that propagate to all clients - **Client isolation** for secrets, strategies, and custom code - **Clear IP boundaries** between platform and client work

---

## 1. Repository Structure

### 1.1 Public OSS Core Repository

**Location:** Public GitHub repository

**License:** AGPL-3.0

**Purpose:** Marketing, validation, and open-source contribution

**Contains:** - Core architecture and design - General improvements and bug fixes - Documentation demonstrating quality and seriousness - Open-source contributions from the community

**Benefits:** - Proves engineering rigor and architectural decisions - Attracts potential clients and contributors - Provides legal comfort (“clear open-source counterpart”) - Enables recruiting signal and community building

**Note:** Bugs fixed in the enterprise base are typically backported to the OSS core (when appropriate) to maintain quality and transparency.

---

### 1.2 Enterprise Base Repository

**Location:** Private repository (fox-v1-infra-enterprise)

**License:** Commercial License

**Purpose:** Foundation for all commercial deployments

**Contains:** - Cleaned-up core codebase - Enterprise-grade defaults and configurations - Hardened logging and security features - Generic commercial extras (scripts, tooling, deployment templates) - Version-tagged releases (v1.1.0, v1.1.1, etc.)

**Access:** - Available to all commercial licensees - Single source of truth for core platform - Tagged releases enable reproducible deployments

**Workflow:** - Bug fixes and improvements are made here first - Changes propagate to client repos via version tags - Prevents divergence into multiple slightly different cores

---

### 1.3 Per-Client Overlay Repositories

**Location:** Private repositories (e.g., `client-omega-fox-infra`, `client-delta-fox-infra`)

**License:** Commercial License + Client-Specific Terms

**Purpose:** Client-specific customizations and integrations

**Contains:** - Client-specific configurations - Custom modules and features built for the client - Glue code for client's existing infrastructure - Deployment recipes tuned to client's stack - Client secrets and proprietary strategies (if applicable)

**Dependency Management:** - Depends on enterprise base via: - Git submodule, or - Git subtree, or - Fork + upstream tracking - Tracks specific version tags from enterprise base - Can be updated to new base versions independently

**Benefits:** - Keeps client secrets and strategies isolated - Allows independent versioning per client - Enables client-specific features without affecting others - Maintains clear separation of concerns

---

## 2. Intellectual Property & Ownership Rules

### 2.1 Core Platform IP

**Ownership:** Fox ML Infrastructure (Consultant)

**Includes:** - Core platform code and architecture - Generic features and improvements - Enterprise base repository contents - Public OSS core repository contents

**Rights:** - Consultant may reuse core platform IP across all clients - Consultant may generalize client-specific work and roll into core (unless exclusivity is purchased) - Core platform improvements benefit all licensees

---

### 2.2 Client-Specific Work IP

**Ownership:** Determined by contract (SOW/MCA)

**Options:** 1. **Client-Owned** — Client receives full ownership of custom code, configs, and strategies 2. **Exclusive License** — Client receives exclusive license (requires premium pricing) 3.

**Non-Exclusive License** — Client receives internal-use license; Consultant may reuse generalized elements

**Default (unless specified otherwise):** - Client-specific logic, configs, and strategy code → Client-owned - Generic patterns and improvements → May be generalized into core platform - Custom work may be generalized and rolled into core product at Consultant's discretion (unless exclusivity premium is paid)

**Critical Clause:** > "Custom work may be generalized and rolled into the core product at Consultant's discretion, unless Client has purchased exclusivity rights."

This prevents future clients from claiming: > “We paid for that feature, you can’t give it to others.”

---

## 2.3 IP Boundaries

**Clear separation:** - **Core platform & generic features** → Consultant IP, reusable - **Client-specific logic/configs/strategy code** → Ownership per contract - **Client data and confidential algorithms** → Never reused, always client-owned

**Documentation:** - IP terms are defined in: - `legal/consulting/IP_TERMS_ADDENDUM.md` - Individual Statement of Work (SOW) documents - Master Consulting Agreement (MCA)

---

## 3. Git Workflow & Version Management

### 3.1 Version Tagging Strategy

**Enterprise Base:** - Semantic versioning: `v1.1.0`, `v1.1.1`, `v1.2.0`, etc. - Tags mark stable releases - Bug fixes and improvements are committed and tagged - All client repos track specific tags or branches

**Client Repos:** - Track specific version tags from enterprise base - Can be updated to new tags independently - Maintain client-specific branches for custom work

---

### 3.2 Update Workflow

**When a bug fix is made:**

1. **Fix in enterprise base** → Commit and tag new version
2. **Update client repos** → Bump to new tag in one pass
3. **Backport to OSS core** (when appropriate) → Maintain transparency

**Benefits:** - Single source of truth prevents divergence - Bug fixes propagate to all clients efficiently - Version tags enable reproducible deployments - Clear upgrade path for clients

---

### 3.3 Preventing Divergence

**Rules:** - **One source of truth** — Enterprise base is the canonical core - **No forking chaos** — Client repos track base versions, don’t diverge into separate cores - **Tag-based updates** — Clients upgrade by bumping version tags - **Centralized fixes** — All core improvements go through enterprise base first

---

## 4. Pricing & Delivery Tiers

The repository structure maps cleanly to pricing tiers:

## 4.1 License Only

**Includes:** - Access to enterprise base repository - Commercial license for internal use - No custom client repository

**Use case:** Organizations that can integrate the platform as-is

---

## 4.2 License + Light Integration

**Includes:** - Access to enterprise base repository - Small client repository with: - Client-specific configs - Minor tweaks and customizations - Basic deployment recipes

**Use case:** Organizations needing minimal customization

---

## 4.3 License + Heavy Customization

**Includes:** - Access to enterprise base repository - Large client repository with: - Extensive custom modules - Deep integrations with client infrastructure - Complex deployment recipes - Ongoing maintenance and updates

**Additional:** - Statement of Work (SOW) for custom development - Ongoing retainer for support and updates - Potentially exclusive IP rights (if premium is paid)

**Use case:** Organizations requiring significant customization and ongoing support

---

## 4.4 Pricing Alignment

Pricing tiers (see `legal/SUBSCRIPTIONS.md`) align with: - **Organization size** (employee count) - **Usage tier** (license only vs. customization level) - **IP exclusivity** (if premium exclusivity is purchased)

---

# 5. Benefits of This Model

## 5.1 For Consultant

- **Scalable maintenance** — Fix bugs once, propagate to all clients
  - **Clear IP boundaries** — Prevents ownership disputes
  - **Marketing value** — OSS core attracts clients and validates quality
  - **Efficient workflow** — One source of truth prevents chaos
  - **Flexible pricing** — Structure supports multiple pricing tiers
- 

## 5.2 For Clients

- **Isolated secrets** — Client-specific repos keep proprietary code separate
- **Clear ownership** — IP terms defined in contracts

- **Upgrade path** — Can update to new base versions independently
  - **Customization flexibility** — Can request client-specific features
  - **Legal comfort** — Clear open-source counterpart demonstrates transparency
- 

## 6. Legal & Contractual Considerations

### 6.1 Required Clauses

All contracts (SOW/MCA) should include:

- **IP ownership definitions** — What is core platform vs. client-specific
  - **Generalization rights** — Consultant's right to generalize client work (unless exclusivity purchased)
  - **Repository access terms** — Who has access to which repos
  - **Version update terms** — How and when clients receive updates
  - **Exclusivity terms** — If client purchases exclusive rights to features
- 

### 6.2 Documentation

IP and delivery terms are documented in: - `legal/consulting/IP_TERMS_ADDENDUM.md` — General IP framework - `legal/consulting/MASTER_CONSULTING_AGREEMENT.md` — Framework agreement  
- Individual SOW documents — Project-specific terms

---

## 7. Summary

**The Model:** - **Public OSS core** → Marketing, validation, transparency - **Enterprise base repo** → Single source of truth for commercial deployments - **Per-client repos** → Client-specific customizations and isolation

**Key Principles:** - One source of truth (enterprise base) - Clear IP boundaries (core vs. client-specific) - Scalable maintenance (fix once, propagate to all) - Flexible pricing (maps to delivery tiers)

**This is a legitimate, scalable infrastructure business model used by many successful infra vendors and consultancies.**

---

## Contact

For questions about the delivery model or custom repository setup:

**Jennifer Lewis**

Fox ML Infrastructure

Email: [jenn.lewis5789@gmail.com](mailto:jenn.lewis5789@gmail.com)

Subject: *Enterprise Delivery Model Inquiry*