

Fox ML Infrastructure – Release Policy

This document defines the versioning strategy, release cadence, and update policies for Fox ML Infrastructure.

1. Versioning Strategy

1.1 Semantic Versioning

Fox ML Infrastructure uses **semantic versioning** (SemVer): MAJOR.MINOR.PATCH

- **MAJOR** (v1.0.0 -> v2.0.0) – Breaking changes, incompatible API changes
- **MINOR** (v1.0.0 -> v1.1.0) – New features, backward-compatible additions
- **PATCH** (v1.0.0 -> v1.0.1) – Bug fixes, security patches, backward-compatible fixes

Examples: - v1.2.0 – Minor release with new features - v1.2.1 – Patch release with bug fixes - v1.2.2 – Patch release with security fixes - v2.0.0 – Major release with breaking changes

1.2 Version Tags

All releases are tagged in the enterprise base repository: - Tags follow the format: v1.2.0, v1.2.1, etc. - Tags are immutable – once released, a version tag is never modified - Release notes accompany each tag

2. Release Cadence

2.1 Patch Releases

Frequency: Weekly (as needed)

Includes: - Bug fixes - Security patches - Documentation updates - Minor configuration improvements

Backward Compatibility: Always backward-compatible within the same minor version

Examples: v1.2.0 -> v1.2.1, v1.2.2, v1.2.3

2.2 Minor Releases

Frequency: Monthly (approximately)

Includes: - New features - Performance improvements - New model types or strategies - Enhanced configuration options - Backward-compatible API additions

Backward Compatibility: Always backward-compatible within the same major version

Examples: v1.2.0 -> v1.3.0, v1.4.0

2.3 Major Releases

Frequency: As needed (typically 6-12 months)

Includes: - Breaking API changes - Architectural changes - Significant refactoring - Removal of deprecated features

Migration: Migration guides and documentation are provided

Examples: v1.9.0 -> v2.0.0

3. Deprecated API Handling

3.1 Deprecation Policy

Deprecation Timeline: 1. **Announcement** – Feature is marked as deprecated in release notes 2. **Warning Period** – Deprecated features remain functional for at least one minor version 3. **Removal** – Deprecated features are removed in the next major version

Example: - v1.2.0 – Feature X is deprecated (announcement) - v1.3.0 – Feature X still works but shows deprecation warnings - v2.0.0 – Feature X is removed

3.2 Migration Support

- **Migration guides** – Documentation provided for migrating from deprecated features
 - **Backward compatibility** – Deprecated features remain functional during the warning period
 - **Support** – Support is provided for migration questions during the deprecation period
-

4. Safe Migration Policy

4.1 Upgrade Path

Recommended Upgrade Strategy: 1. **Stay current** – Upgrade to the latest patch version within your minor version 2. **Test minor upgrades** – Test minor version upgrades in a development environment 3. **Plan major upgrades** – Major version upgrades require planning and migration

Example: - Current: v1.2.5 - Recommended: Upgrade to v1.2.6 (latest patch) - Next: Test v1.3.0 in development, then upgrade production - Future: Plan migration to v2.0.0 when ready

4.2 Version Support

Supported Versions: - **Full support** – Latest minor version and the previous minor version - **Security patches only** – Versions within two minor versions of latest - **No support** – Versions older than two minor versions

Example (if latest is v1.5.0): - v1.5.x – Full support - v1.4.x – Full support - v1.3.x – Security patches only - v1.2.x and older – No support

Recommendation: Stay within two minor versions of the latest release for full support.

5. Release Communication

5.1 Release Notes

Each release includes: - Version number and release date - Summary of changes - New features (minor/major releases) - Bug fixes (patch releases) - Security fixes (if applicable) - Deprecation notices (if applicable) - Migration notes (if applicable)

5.2 Enterprise Changelog

Enterprise customers receive: - CHANGelog_ENTERPRISE.md – Detailed changelog for commercial releases - Release notes in private repository - Email notifications for major releases and security patches (Enterprise/Premium support)

5.3 Security Advisories

Security patches are communicated: - **Immediately** – Enterprise and Premium support customers are notified immediately - **Public disclosure** – Security issues are disclosed after patches are available - **Patch releases** – Security fixes are released as patch versions

6. Client Repository Updates

6.1 Update Process

Client repositories track version tags from the enterprise base:

1. **Tag-based tracking** – Client repos track specific version tags (e.g., v1.2.0)
2. **Independent updates** – Each client repo can be updated independently
3. **Version bumping** – Clients upgrade by updating to a new version tag

6.2 Update Recommendations

Recommended Update Schedule: - **Patch releases** – Update within 1-2 weeks (especially security patches) - **Minor releases** – Update within 1-2 months (test in development first) - **Major releases** – Update when ready (plan migration, test thoroughly)

See `LEGAL/ENTERPRISE_DELIVERY.md` for details on repository structure and update workflow.

7. Breaking Changes

7.1 Major Version Policy

Breaking changes only occur in major versions:

- API changes that require code modifications
- Configuration format changes
- Removal of deprecated features
- Architectural changes that affect integration

7.2 Migration Support

For major version upgrades: - **Migration guides** – Detailed documentation provided - **Support** – Enterprise/Premium support includes migration assistance - **Timeline** – Deprecated features remain functional until the next major version

8. Release Quality Assurance

8.1 Testing

All releases undergo: - Unit testing - Integration testing - Regression testing - Documentation review

8.2 Release Criteria

Releases are made when: - All tests pass - Documentation is updated - Release notes are prepared - Security review is completed (for security patches)

9. Emergency Patches

9.1 Critical Security Issues

For critical security issues: - **Immediate patch** – Patch is released as soon as possible - **Notification** – Enterprise/Premium customers are notified immediately - **Documentation** – Security advisory is published

9.2 Critical Bug Fixes

For critical bugs: - **Priority handling** – Enterprise/Premium support customers receive priority - **Patch release** – Patch is released as soon as possible - **Communication** – Affected customers are notified

10. Summary

Key Release Principles:

1. **Semantic versioning** – Clear version numbering (MAJOR.MINOR.PATCH)
2. **Regular cadence** – Patch releases weekly, minor releases monthly
3. **Backward compatibility** – Breaking changes only in major versions
4. **Deprecation policy** – Clear timeline for deprecated features
5. **Safe migration** – Migration guides and support for upgrades
6. **Version support** – Support for current and previous minor versions
7. **Security priority** – Immediate patches and notifications for security issues

This policy ensures predictable, safe, and well-communicated releases for all Fox ML Infrastructure customers.

Contact

For questions about releases or upgrade planning:

Jennifer Lewis

Fox ML Infrastructure LLC

Email: jenn.lewis5789@gmail.com

Subject: *Release Policy Inquiry – Fox ML Infrastructure*

Related Documents

- `LEGAL/ENTERPRISE_DELIVERY.md` – Repository structure and update workflow
- `CHANGELOG_ENTERPRISE.md` – Detailed changelog for commercial releases
- `LEGAL/SUPPORT_POLICY.md` – Support tiers and response times
- `LEGAL/SERVICE_LEVEL_AGREEMENT.md` – SLA terms for Enterprise support