

# Fox ML Infrastructure — Release Notes & Tagging Standard

This document defines the standard format for release notes, version tagging, and changelog entries. This standard ensures consistency and clarity for enterprise clients tracking updates and changes.

---

## 1. Semantic Versioning

### 1.1 Version Format

All releases use 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 Pre-Release Versions

Pre-release versions (if applicable):

- **Alpha:** v1.2.0-alpha.1
- **Beta:** v1.2.0-beta.1
- **Release Candidate:** v1.2.0-rc.1

Pre-release versions are not recommended for production use.

---

## 2. Git Tagging

### 2.1 Tag Format

All releases are tagged in the enterprise base repository:

- **Format:** v1.2.0, v1.2.1, v1.2.2, etc.
- **Prefix:** Always use v prefix
- **Immutable:** Tags are immutable — once released, a version tag is never modified
- **Annotated tags:** Use annotated tags (not lightweight tags) for better metadata

### 2.2 Tagging Commands

Example tagging commands:

```
# Create annotated tag
git tag -a v1.2.0 -m "Release v1.2.0: New features and improvements"

# Push tag to remote
git push origin v1.2.0
```

```
# List tags
git tag -l "v*"
```

## 2.3 Tag Naming Rules

Tag naming rules:

- **Lowercase:** Use lowercase v prefix
  - **No spaces:** No spaces in version numbers
  - **Consistent format:** Always use vMAJOR.MINOR.PATCH format
  - **No special characters:** No special characters except dots and hyphens (for pre-releases)
- 

## 3. Release Notes Format

### 3.1 Standard Structure

Release notes follow this structure:

```
# Release v1.2.0 - [Release Date]

## Summary
Brief one-paragraph summary of the release.

## New Features
- Feature 1: Description
- Feature 2: Description

## Improvements
- Improvement 1: Description
- Improvement 2: Description

## Bug Fixes
- Fix 1: Description (Issue #123)
- Fix 2: Description (Issue #124)

## Security Fixes
- Security fix 1: Description (CVE-YYYY-XXXXX, if applicable)

## Deprecations
- Deprecated feature: Description (removal planned in v2.0.0)

## Migration Notes
- Migration step 1: Description
- Migration step 2: Description

## Breaking Changes
- Breaking change 1: Description
- Breaking change 2: Description
```

## ## Contributors

- Contributor 1
- Contributor 2

## 3.2 Release Notes Sections

Required sections (if applicable):

- **Summary** — Always include
- **New Features** — Include if new features added
- **Improvements** — Include if improvements made
- **Bug Fixes** — Include if bugs fixed
- **Security Fixes** — Include if security fixes applied
- **Deprecations** — Include if features deprecated
- **Migration Notes** — Include for major releases or significant changes
- **Breaking Changes** — Include for major releases

Optional sections:

- **Contributors** — Include if external contributors
  - **Performance** — Include if performance improvements
  - **Documentation** — Include if significant documentation updates
- 

## 4. Changelog Format

### 4.1 Changelog Structure

Changelog entries follow this format:

## [v1.2.0] - 2026-01-15

#### ### Added

- New feature 1
- New feature 2

#### ### Changed

- Changed behavior 1
- Changed behavior 2

#### ### Fixed

- Bug fix 1 (Issue #123)
- Bug fix 2 (Issue #124)

#### ### Security

- Security fix 1 (CVE-YYYY-XXXXX)

#### ### Deprecated

- Deprecated feature 1 (removal in v2.0.0)

### Removed

- Removed feature 1 (was deprecated in v1.1.0)

## 4.2 Changelog Categories

Standard changelog categories:

- **Added** — New features
- **Changed** — Changes to existing functionality
- **Deprecated** — Features that will be removed
- **Removed** — Removed features
- **Fixed** — Bug fixes
- **Security** — Security fixes

Additional categories (if applicable):

- **Performance** — Performance improvements
  - **Documentation** — Documentation updates
  - **Refactoring** — Code refactoring (internal)
- 

## 5. Release Notes Content Guidelines

### 5.1 Clarity and Conciseness

Release notes should be:

- **Clear** — Use clear, concise language
- **Specific** — Be specific about what changed
- **Actionable** — Include actionable information (migration steps, etc.)
- **Scannable** — Use bullet points and headings for easy scanning

### 5.2 Technical Detail Level

Technical detail level:

- **User-facing** — Focus on user-visible changes
- **Technical context** — Include technical context when relevant
- **Examples** — Include examples for complex changes
- **Links** — Link to detailed documentation when available

### 5.3 Breaking Changes

Breaking changes must include:

- **Clear identification** — Clearly marked as “Breaking Changes”
  - **Impact description** — Description of impact on users
  - **Migration guide** — Step-by-step migration instructions
  - **Timeline** — Timeline for migration (if applicable)
-

## 6. Enterprise Changelog

### 6.1 Enterprise Changelog Format

Enterprise changelog (CHANGELOG\_ENTERPRISE.md) follows this format:

```
# Fox ML Infrastructure - Enterprise Changelog
```

```
## Version History
```

```
### v1.2.0 - 2026-01-15
```

```
**Release Type:** Minor Release
```

```
**New Features:**
```

- Feature 1: Description
- Feature 2: Description

```
**Improvements:**
```

- Improvement 1: Description
- Improvement 2: Description

```
**Bug Fixes:**
```

- Fix 1: Description
- Fix 2: Description

```
**Security Fixes:**
```

- Security fix 1: Description

```
**Migration Notes:**
```

- Migration step 1: Description

```
---
```

```
### v1.1.0 - 2025-12-10
```

```
[Previous release notes...]
```

### 6.2 Enterprise Changelog Updates

Enterprise changelog is updated:

- **With each release** — Updated with each new release
  - **Chronological order** — Most recent releases first
  - **Complete history** — Maintains complete release history
  - **Version links** — Links to version tags in repository
-

## 7. Tagging Workflow

### 7.1 Pre-Release Checklist

Before creating a release tag:

- ☐ All tests pass
- ☐ Documentation is updated
- ☐ Release notes are prepared
- ☐ Changelog is updated
- ☐ Version numbers are updated (if applicable)
- ☐ Security review completed (for security patches)

### 7.2 Tag Creation Process

Tag creation process:

1. **Prepare release** — Complete pre-release checklist
2. **Create tag** — Create annotated tag with release notes
3. **Push tag** — Push tag to remote repository
4. **Update changelog** — Update enterprise changelog
5. **Notify clients** — Notify enterprise clients (if applicable)

### 7.3 Post-Release

After creating a release tag:

- **Verify tag** — Verify tag is accessible in repository
  - **Update documentation** — Update any documentation referencing versions
  - **Client notification** — Notify clients of new release (Enterprise/Premium support)
- 

## 8. Version Numbering Guidelines

### 8.1 When to Increment MAJOR

Increment MAJOR version when:

- **Breaking API changes** — API changes that break backward compatibility
- **Removal of features** — Removal of previously available features
- **Architectural changes** — Significant architectural changes
- **Configuration changes** — Breaking configuration format changes

### 8.2 When to Increment MINOR

Increment MINOR version when:

- **New features** — New features added (backward-compatible)
- **Performance improvements** — Significant performance improvements
- **New model types** — New model types or strategies added
- **Enhanced functionality** — Enhanced functionality (backward-compatible)

## 8.3 When to Increment PATCH

Increment PATCH version when:

- **Bug fixes** — Bug fixes (backward-compatible)
  - **Security patches** — Security patches and fixes
  - **Documentation updates** — Documentation corrections
  - **Minor improvements** — Minor improvements and optimizations
- 

## 9. Examples

### 9.1 Example Release Notes

Example: v1.2.0 Release Notes

# Release v1.2.0 - January 15, 2026

## Summary

This release adds new model types, improves performance, and fixes several bugs.

## New Features

- Added Transformer model support for time-series forecasting
- Added multi-task learning capabilities
- Added new feature engineering functions

## Improvements

- Improved training performance by 30% for large datasets
- Enhanced configuration validation with better error messages
- Updated documentation with new examples

## Bug Fixes

- Fixed memory leak in feature engineering pipeline (Issue #123)
- Fixed incorrect validation split calculation (Issue #124)
- Fixed configuration loading error for nested configs (Issue #125)

## Security Fixes

- Updated dependency with security vulnerability (CVE-2025-12345)

## Migration Notes

- Configuration format unchanged - no migration required
- New optional parameters available - see documentation for details

### 9.2 Example Changelog Entry

Example: Changelog Entry

## [v1.2.0] - 2026-01-15

### Added

- Transformer model support for time-series forecasting
- Multi-task learning capabilities
- New feature engineering functions

### ### Changed

- Improved training performance by 30% for large datasets
- Enhanced configuration validation with better error messages

### ### Fixed

- Memory leak in feature engineering pipeline (Issue #123)
- Incorrect validation split calculation (Issue #124)
- Configuration loading error for nested configs (Issue #125)

### ### Security

- Updated dependency with security vulnerability (CVE-2025-12345)
- 

## 10. Contact

For questions about release notes or tagging:

**Jennifer Lewis**

Fox ML Infrastructure LLC

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

Subject: *Release Notes Inquiry — Fox ML Infrastructure*

---

## 11. Related Documents

- `legal/RELEASE_POLICY.md` — Release policy (versioning strategy and cadence)
  - `legal/CHANGELOG_ENTERPRISE.md` — Enterprise changelog
  - `docs/` — Technical documentation
- 

## 12. Summary

**Key Release Notes & Tagging Principles:**

1. **Semantic versioning** — Use MAJOR.MINOR.PATCH format consistently
2. **Annotated tags** — Use annotated tags with release notes
3. **Standard format** — Follow standard release notes and changelog format
4. **Clear communication** — Clearly communicate changes, especially breaking changes
5. **Complete history** — Maintain complete release history in changelog
6. **Client notification** — Notify clients of releases (Enterprise/Premium support)

**This standard ensures consistency and clarity for enterprise clients tracking updates.**