Languages (28): 1.1.0 English

keep a changelog

Don't let your friends dump git logs into changelogs.

Version **1.1.0**

```
# Changelog

All notable changes to this project will be documented in this file.

The format is based on [Keep a Changelog](https://keepachangelog.com/en/1
and this project adheres to [Semantic Versioning](https://semver.org/spec

## [Unreleased]

### Added

- v1.1 Brazilian Portuguese translation.
- v1.1 German Translation
- v1.1 Spanish translation.
- v1.1 Italian translation.
```

What is a changelog?

A changelog is a file which contains a curated, chronologically ordered list of notable changes for each version of a project.

Why keep a changelog?

To make it easier for users and contributors to see precisely what notable changes have been made between each release (or version) of the project.

Who needs a changelog?

People do. Whether consumers or developers, the end users of software are human beings who care about what's in the software. When the software changes, people want to know why and how.

How do I make a good changelog?

Guiding Principles

- Changelogs are *for humans*, not machines.
- There should be an entry for every single version.
- The same types of changes should be grouped.
- Versions and sections should be linkable.
- The latest version comes first.
- The release date of each version is displayed.
- Mention whether you follow Semantic Versioning.

Types of changes

- `Added` for new features.
- `Changed` for changes in existing functionality.
- `Deprecated` for soon-to-be removed features.
- `Removed` for now removed features.

- `Fixed` for any bug fixes.
- `Security` in case of vulnerabilities.

How can I reduce the effort required to maintain a changelog?

Keep an `Unreleased` section at the top to track upcoming changes.

This serves two purposes:

- People can see what changes they might expect in upcoming releases
- At release time, you can move the `Unreleased` section changes into a new release version section.

Can changelogs be bad?

Yes. Here are a few ways they can be less than useful.

Commit log diffs

Using commit log diffs as changelogs is a bad idea: they're full of noise. Things like merge commits, commits with obscure titles, documentation changes, etc.

The purpose of a commit is to document a step in the evolution of the source code. Some projects clean up commits, some don't.

The purpose of a changelog entry is to document the noteworthy difference, often across multiple commits, to communicate them clearly to end users.

Ignoring Deprecations

When people upgrade from one version to another, it should be painfully clear when something will break. It should be possible to upgrade to a version that lists deprecations, remove what's deprecated, then upgrade to the version where the deprecations become removals.

If you do nothing else, list deprecations, removals, and any breaking changes in your changelog.

Confusing Dates

Regional date formats vary throughout the world and it's often difficult to find a human-friendly date format that feels intuitive to everyone. The advantage of dates formatted like `2017-07-17` is that they follow the order of largest to smallest units: year, month, and day. This format also doesn't overlap in ambiguous ways with other date formats, unlike some regional formats that switch the position of month and day numbers. These reasons, and the fact this date format is an ISO standard, are why it is the recommended date format for changelog entries.

Inconsistent Changes

A changelog which only mentions some of the changes can be as dangerous as not having a changelog. While many of the changes may not be relevant - for instance, removing a single whitespace may not need to be recorded in all instances - any important changes should be mentioned in the changelog. By inconsistently applying changes, your users may mistakenly think that the changelog is the single source of truth. It ought to be. With great power comes great responsibility - having a good changelog means having a consistently updated changelog.

There's more. Help me collect these antipatterns by opening an issue or a pull request.

Frequently Asked Questions

Is there a standard changelog format?

Not really. There's the GNU changelog style guide, or the two-paragraph-long GNU NEWS file "guideline". Both are inadequate or insufficient.

This project aims to be a better changelog convention. It comes from observing good practices in the open source community and gathering them.

Healthy criticism, discussion and suggestions for improvements are welcome.

What should the changelog file be named?

Call it `CHANGELOG.md`. Some projects use `HISTORY`, `NEWS` or `RELEASES`.

While it's easy to think that the name of your changelog file doesn't matter that much, why make it harder for your end users to consistently find notable changes?

What about GitHub Releases?

It's a great initiative. Releases can be used to turn simple git tags (for example a tag named `v1.0.0`) into rich release notes by manually adding release notes or it can pull annotated git tag messages and turn them into notes.

GitHub Releases create a non-portable changelog that can only be displayed to users within the context of GitHub. It's possible to make them look very much like the Keep a Changelog format, but it tends to be a bit more involved.

The current version of GitHub releases is also arguably not very discoverable by end-users, unlike the typical uppercase files (`README`, `CONTRIBUTING`, etc.). Another minor issue is that the interface doesn't currently offer links to commit logs between each release.

Can changelogs be automatically parsed?

It’s difficult, because people follow wildly different formats and file names.

Vandamme is a Ruby gem created by the Gemnasium team and which parses many (but not all) open source project changelogs.

What about yanked releases?

Yanked releases are versions that had to be pulled because of a serious bug or security issue. Often these versions don't even appear in change logs. They should. This is how you should display them:

```
## [0.0.5] - 2014-12-13 [YANKED]
```

The `[YANKED]` tag is loud for a reason. It's important for people to notice it. Since it's surrounded by brackets it's also easier to parse programmatically.

Should you ever rewrite a changelog?

Sure. There are always good reasons to improve a changelog. I regularly open pull requests to add missing releases to open source projects with unmaintained changelogs.

It's also possible you may discover that you forgot to address a breaking change in the notes for a version. It's obviously important for you to update your changelog in this case.

How can I contribute?

This document is not the **truth**; it’s my carefully considered opinion, along with information and examples I gathered.

This is because I want our community to reach a consensus. I believe the discussion is as important as the end result.

So please **pitch in**.

Conversations

I went on The Changelog podcast to talk about why maintainers and contributors should care about changelogs, and also about the motivations behind this project.