# Contributing to Logstash

All contributions are welcome: ideas, patches, documentation, bug reports,

complaints, etc!

Programming is not a required skill, and there are many ways to help out!

It is more important to us that you are able to contribute.

That said, some basic guidelines, which you are free to ignore :)

## Want to learn?

Want to lurk about and see what others are doing with Logstash?

\* The #logstash channel on Elastic Stack Community slack (https://elasticstack.slack.com/channels/logstash) is a good place to start.

\* The [forum](https://discuss.elastic.co/c/logstash) is also

great for learning from others.

## Got Questions?

Have a problem you want Logstash to solve for you?

\* You can ask a question in the [forum](https://discuss.elastic.co/c/logstash)

\* You are welcome to join Elastic Stack Community slack (https://elasticstack.slack.com) and ask for help on the #logstash channel.

## Have an Idea or Feature Request?

\* File a ticket on [GitHub](https://github.com/elastic/logstash/issues). Please remember that GitHub is used only for issues and feature requests. If you have a general question, the [forum](https://discuss.elastic.co/c/logstash) or Elastic Stack Community slack (https://elasticstack.slack.com) is the best place to ask.

## Something Not Working? Found a Bug?

If you think you found a bug, it probably is a bug.

\* If it is a general Logstash or a pipeline issue, file it in [Logstash GitHub](https://github.com/elasticsearch/logstash/issues)

\* If it is specific to a plugin, please file it in the respective repository under [logstash-plugins](https://github.com/logstash-plugins)

\* or ask the [forum](https://discuss.elastic.co/c/logstash).

## Issue Prioritization

The Logstash team takes time to digest, consider solutions, and weigh applicability of issues to both the broad

Logstash user base and our own goals for the project. Through this process, we triage and update issues as we get to them. Please provide context in your issues instead of just adding a +1 comment. If you like a certain idea or enhancement, and have nothing more to add, please just use GitHub :+1: emoji.

## Found a Security Issue?

If you've found a security issue, before submitting anything via a PR, please

get in touch with our security team [here](https://www.elastic.co/community/security).

# Contributing Documentation and Code Changes

If you have a bugfix or new feature that you would like to contribute to Logstash, and you think it will take

more than a few minutes to produce the fix (ie; write code), it is worth discussing the change with the Logstash

users and developers first. You can reach us via [GitHub](https://github.com/elastic/logstash/issues), the [forum](https://discuss.elastic.co/c/logstash), or Elastic Stack Community slack (https://elasticstack.slack.com).

Please note that Pull Requests without tests and documentation may not be merged. If you would like to contribute but do not have

experience with writing tests, please ping us on the forum or create a PR and ask for our help.

If you would like to contribute to Logstash, but don't know where to start, you can use the GitHub labels "adoptme",

"low hanging fruit" and "good first issue". Issues marked with these labels are relatively easy, and provide a good starting

point to contribute to Logstash.

See the following links:

\* https://github.com/elastic/logstash/labels/adoptme

\* https://github.com/elastic/logstash/labels/low%20hanging%20fruit

\* https://github.com/elastic/logstash/labels/good%20first%20issue

Or go directly here for an exhaustive list: https://github.com/elastic/logstash/contribute

Using IntelliJ? See a detailed getting started guide [here](https://docs.google.com/document/d/1kqunARvYMrlfTEOgMpYHig0U-ZqCcMJfhvTtGt09iZg/pub).

## Contributing to plugins

Check our [documentation](https://www.elastic.co/guide/en/logstash/current/contributing-to-logstash.html) on how to contribute to plugins or write your own!

### Logstash Plugin Changelog Guidelines

This document provides guidelines on editing a logstash plugin's CHANGELOG file.

### What's a CHANGELOG file?

According to [keepachangelog.com](https://keepachangelog.com/en/1.0.0/):

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

Each logstash plugin contains a CHANGELOG.md markdown file. Here's an example: https://github.com/logstash-plugins/logstash-input-file/blob/master/CHANGELOG.md

#### How is the CHANGELOG.md populated?

Updates are done manually, according to the following rules:

#### When should I add an entry to the CHANGELOG.md?

Update the changelog before you push a new release to rubygems.org.

To prevent merged pull requests from sitting in the master branch until someone decides to cut a release,

we strive to publish a new version for each pull request. (This is not mandatory. See below.)

You are therefore encouraged to bump the gemspec version and add a changelog entry in the same pull request as the change. See [logstash-input-snmp#8](https://github.com/logstash-plugins/logstash-input-snmp/pull/8/files) as an example.

If have no intentions to release the change immediately after merging the pull request

then you should still create an entry in the changelog, using the word `Unreleased` as the version. Example:

```markdown

## Unreleased

- Removed `sleep(0.0001)` from inner loop that was making things a bit slower [#133](http://example.org)

## 3.3.3

- Fix when no delimiter is found in a chunk, the chunk is reread - no forward progress

is made in the file [#185](http://example.org)

```

#### What is the format of the Changelog entries?

Most code in logstash-plugins comes from self-contained changes in the form of pull requests, so each entry should:

1. Be a summary of the Pull Request and contain a markdown link to the PR.

2. Begin your entry with an introductory label if appropriate. Labels include:

`[DOC]`, `[BREAKING]`, `[SECURITY]`.

NOTE: Your PR may not need a label.

3. After the label, start with one of the following keywords:

`Added`, `Changed`, `Deprecated`, `Removed`, `Fixed`.

4. Keep multiple entries with the same keyword in the same changelog revision together.

Labels (optional):

- \*\*`[BREAKING]`\*\* for a breaking change.

Note that breaking changes should warrant a major version bump.

- \*\*`[DOC]`\*\* for a change that affects only documentation, and not code.

- \*\*`[SECURITY]`\*\* for a security fix. If you're working on a security issue,

please make sure to follow the process outlined by our security team.

If you haven't done so already, please [get in touch with them](https://www.elastic.co/community/security) first.

Keywords (copied from [keepachangelog.com](https://keepachangelog.com/en/1.0.0/#how)):

- \*\*`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.

- \*\*Important reminder\*\*: If you're working on a security issue, please make sure

you're following the process outlined by our security team. If you haven't

done so already, please get in touch with them

[here](https://www.elastic.co/community/security) first.

Example:

```

## 4.0.0

- Changed default value of `number\_of\_threads` from 2 to 1 [#101](http://example.org)

- Changed default value of `execution\_bugs` from 30 to 0 [#104](http://example.org)

- [BREAKING] Removed obsolete option `enable\_telnet` option [#100](http://example.org)

## 3.3.3

- [DOC] Fixed incorrect formatting of code sample [#85](http://example.org)

## 3.3.2

- Fixed incorrect serialization of input data when encoding was `Emacs-Mule` [#84](http://example.org)

## 3.3.1

- Fixed memory leak by removing calls to `leak\_lots\_of\_memory` [#86](http://example.org)

```

## Contribution Steps

1. Test your changes! [Run](https://github.com/elastic/logstash#testing) the test suite

2. Please make sure you have signed our [Contributor License

Agreement](https://www.elastic.co/contributor-agreement/). We are not

asking you to assign copyright to us, but to give us the right to distribute

your code without restriction. We ask this of all contributors in order to

assure our users of the origin and continuing existence of the code. You

need to sign the CLA only once. If your contribution involves changes to

third-party dependencies in Logstash core or the default plugins, you may

wish to review the documentation for our [dependency audit process](https://github.com/elastic/logstash/blob/master/tools/dependencies-report/README.md).

3. Send a pull request! Push your changes to your fork of the repository and

[submit a pull

request](https://help.github.com/articles/using-pull-requests). In the pull

request, describe what your changes do and mention any bugs/issues related

to the pull request.

# Pull Request Guidelines

The following exists as a way to set expectations for yourself and for the review process. We \*want\* to merge fixes and features, so let's describe how we can achieve this:

## Goals

\* To constantly make forward progress on PRs

\* To have constructive discussions on PRs

## Overarching Guiding Principles

Keep these in mind as both authors and reviewers of PRs:

\* Have empathy in both directions (reviewer <--> reviewee/author)

\* Progress over perfection and personal preferences

\* Authors and reviewers should proactively address questions of pacing in order to reach an acceptable balance between meeting the author's expected timeline for merging the PR and the reviewer's ability to keep up with revisions to the PR.

## As a reviewee (i.e. author) of a PR:

\* I must put up atomic PRs. This helps the reviewer of the PR do a high quality review fast. "Atomic" here means two things:

- The PR must contain related changes and leave out unrelated changes (e.g. refactorings, etc. that could be their own PR instead).

- If the PR could be broken up into two or more PRs either "vertically" (by separating concerns logically) or horizontally (by sharding the PR into a series of PRs --- usually works well with mass refactoring or cleanup type PRs), it should. A set of such related PRs can be tracked and given context in a meta issue.

\* I must strive to please the reviewer(s). In other words, bias towards taking the reviewers suggestions rather than getting into a protracted argument. This helps move the PR forward. A convenient "escape hatch" to use might be to file a new issue for a follow up discussion/PR. If you find yourself getting into a drawn out argument, ask yourself: is this a good use of our time?

## As a reviewer of a PR:

\* I must first focus on whether the PR works functionally -- i.e. does it solve the problem (bug, feature, etc.) it sets out to solve.

\* Then I should ask myself: can I understand what the code in this PR is doing and, more importantly, why its doing whatever its doing, within 1 or 2 passes over the PR?

\* If yes, LGTM the PR!

\* If no, ask for clarifications on the PR. This will usually lead to changes in the code such as renaming of variables/functions or extracting of functions or simply adding "why" inline comments. But first ask the author for clarifications before assuming any intent on their part.

\* I must not focus on personal preferences or nitpicks. If I understand the code in the PR but simply would've implemented the same solution a different way that's great but its not feedback that belongs in the PR. Such feedback only serves to slow down progress for little to no gain.