<!--

---

title: "Contributing"

custom\_edit\_url: https://github.com/netdata/netdata/edit/master/CONTRIBUTING.md

---

-->

# Contributing

Thank you for considering contributing to Netdata.

We love to receive contributions. Maintaining a platform for monitoring everything imaginable requires a broad understanding of a plethora of technologies, systems and applications. We rely on community contributions and user feedback to continue providing the best monitoring solution out there.

There are many ways to contribute, with varying requirements of skills, explained in detail in the following sections.

Specific GitHub issues we need help with can be seen [here](https://github.com/netdata/netdata/issues?q=is%3Aissue+is%3Aopen+sort%3Aupdated-desc+label%3A%22help+wanted%22). Some of them are also labeled as "good first issue".

## All Netdata Users

### Give Netdata a GitHub star

This is the minimum open-source users should contribute back to the projects they use. Github stars help the project gain visibility, stand out. So, if you use Netdata, consider pressing that button. \*\*It really matters\*\*.

### Spread the word

Community growth allows the project to attract new talent willing to contribute. This talent is then developing new features and improves the project. These new features and improvements attract more users and so on. It is a loop. So, post about Netdata, present it to local meetups you attend, let your online social network or twitter, facebook, reddit, etc. know you are using it. \*\*The more people involved, the faster the project evolves\*\*.

### Provide feedback

Is there anything that bothers you about Netdata? Did you experience an issue while installing it or using it? Would you like to see it evolve to you need? Let us know. [Open a github issue](https://github.com/netdata/netdata/issues) to discuss it. Feedback is very important for open-source projects. We can't commit we will do everything, but your feedback influences our road-map significantly. \*\*We rely on your feedback to make Netdata better\*\*.

### Translate some documentation

The [Netdata localization project](https://github.com/netdata/localization) contains instructions on how to provide translations for parts of our documentation. Translating the entire documentation is a daunting task, but you can contribute as much as you like, even a single file. The Chinese translation effort has already begun and we are looking forward to more contributions.

### Sponsor a part of Netdata

Netdata is a complex system, with many integrations for the various collectors, backends and notification endpoints. As a result, we rely on help from "sponsors", a concept similar to "power users" or "product owners". To become a sponsor, just let us know in any Github issue and we will record your GitHub username in a "CONTRIBUTORS.md" in the appropriate directory.

#### Sponsor a collector

Netdata is all about simplicity and meaningful presentation. A "sponsor" for a collector does the following:

- Assists the devs with feedback on the charts.

- Specifies the alarms that would make sense for each metric.

- When the implementation passes QA, tests the implementation in production.

- Uses the charts and alarms in his/her day to day work and provides additional feedback.

- Requests additional improvements as things change (e.g. new versions of an API are available).

#### Sponsor a backend

We already support various [backends](backends) and we intend to support more. A "sponsor" for a backend:

- Suggests ways in which the information in Netdata could best be exposed to the particular backend, to facilitate meaningful presentation.

- When the implementation passes QA, tests the implementation in production.

- Uses the backend in his/her day to day work and provides additional feedback, after the backend is delivered.

- Requests additional improvements as things change (e.g. new versions of the backend API are available).

#### Sponsor a notification method

Netdata delivers alarms via various [notification methods](health/notifications). A "sponsor" for a notification method:

- Points the devs to the documentation for the API and identifies any unusual features of interest (e.g. the ability in Slack to send a notification either to a channel or to a user).

- Uses the notification method in production and provides feedback.

- Requests additional improvements as things change (e.g. new versions of the API are available).

## Experienced Users

### Help other users

As the project grows, an increasing share of our time is spent on supporting this community of users in terms of answering questions, of helping users understand how Netdata works and find their way with it. Helping other users is crucial. It allows the developers and maintainers of the project to focus on improving it.

### Improve documentation

Our documentation is in need of constant improvement and expansion. As Netdata's features grow, we need to clearly explain how each feature works and document all the possible configurations. And as Netdata's community grows, we need to improve existing documentation to make it more accessible to people of all skill levels.

We also need to produce beginner-level tutorials on using Netdata to monitor common applications, web servers, and more.

Start with the [guide for contributing to documentation](/docs/contributing/contributing-documentation.md), and then review the [documentation style guide](/docs/contributing/style-guide.md) for specifics on how we write our documentation.

Don't be afraid to submit a pull request with your corrections or additions! We need a lot of help and are willing to guide new contributors through the process.

## Developers

We expect most contributions to be for new data collection plugins. You can read about how external plugins work [here](collectors/plugins.d/). Additional instructions are available for [Node.js plugins](collectors/node.d.plugin) and [Python plugins](collectors/python.d.plugin).

Of course we appreciate contributions for any other part of the Netdata agent, including the [daemon](daemon), [backends for long term archiving](backends/), innovative ways of using the [REST API](web/api) to create cool [Custom Dashboards](web/gui/custom/) or to include Netdata charts in other applications, similarly to what can be done with [Confluence](web/gui/confluence/).

If you are working on the C source code please be aware that we have a standard build configuration that we use. This

is meant to keep the source tree clean and free of warnings. When you are preparing to work on the code:

```

CFLAGS="-O1 -ggdb -Wall -Wextra -Wformat-signedness -fstack-protector-all -DNETDATA\_INTERNAL\_CHECKS=1 -D\_FORTIFY\_SOURCE=2 -DNETDATA\_VERIFY\_LOCKS=1" ./netdata-installer.sh --disable-lto --dont-wait

```

Typically we will enable LTO during production builds. The reasons for configuring it this way are:

| CFLAG / argument | Reasoning |

| ---------------- | --------- |

| `-O1` | This makes the debugger easier to use as it disables optimisations that break the relationship between the source and the state of the debugger |

| `-ggdb` | Enable debugging symbols in gdb format (this also works with clang / llbdb) |

| `-Wall -Wextra -Wformat-signedness` | Really, definitely, absolutely all the warnings |

| `-DNETDATA\_INTERNAL\_CHECKS=1` | This enables the debug.log and turns on the macro that outputs to it |

| `-D\_FORTIFY\_SOURCE=2` | Enable buffer-overflow checks on string-processing functions |

| `-DNETDATA\_VERIFY\_LOCKS=1` | Enable extra checks and debug |

| `--disable-lto ` | We enable LTO for production builds, but disable it during development are it can hide errors about missing symbols that have been pruned. |

Before submitting a PR we run through this checklist:

\* Compilation warnings

\* valgrind

\* ./netdata-installer.sh

\* make dist

\* `packaging/makeself/build-x86\_64-static.sh`

\* `clang-format -style=file`

Please be aware that the linting pass at the end is currently messy as we are transitioning between code styles

across most of our code-base, but we prefer new contributions that match the linting style.

### Contributions Ground Rules

#### Code of Conduct and CLA

We expect all contributors to abide by the [Contributor Covenant Code of Conduct](CODE\_OF\_CONDUCT.md). For a pull request to be accepted, you will also need to accept the [Netdata contributors license agreement](CONTRIBUTORS.md), as part of the PR process.

#### Performance and efficiency

Everything on Netdata is about efficiency. We need Netdata to always be the most lightweight monitoring solution available. We will reject to merge PRs that are not optimal in resource utilization and efficiency.

Of course there are cases that such technical excellence is either not reasonable or not feasible. In these cases, we may require the feature or code submitted to be by disabled by default.

#### Meaningful metrics

Unlike other monitoring solutions, Netdata requires all metrics collected to have some structure attached to them. So, Netdata metrics have a name, units, belong to a chart that has a title, a family, a context, belong to an application, etc.

This structure is what makes Netdata different. Most other monitoring solution collect bulk metrics in terms of name-value pairs and then expect their users to give meaning to these metrics during visualization. This does not work. It is neither practical nor reasonable to give to someone 2000 metrics and let him/her visualize them in a meaningful way.

So, Netdata requires all metrics to have a meaning at the time they are collected. We will reject to merge PRs that loosely collect just a "bunch of metrics", but we are very keen to help you fix this.

#### Automated Testing

Netdata is a very large application to have automated testing end-to-end. But automated testing is required for crucial functions of it.

Generally, all pull requests should be coupled with automated testing scenarios. However since we do not currently have a framework in place for testing everything little bit of it, we currently require automated tests for parts of Netdata that seem too risky to be changed without automated testing.

Of course, manual testing is always required.

#### Netdata is a distributed application

Netdata is a distributed monitoring application. A few basic features can become quite complicated for such applications. We may reject features that alter or influence the nature of Netdata, though we usually discuss the requirements with contributors and help them adapt their code to be better suited for Netdata.

#### Operating systems supported

Netdata should be running everywhere, on every production system out there.

Although we focus on \*\*supported operating systems\*\*, we still want Netdata to run even on non-supported systems. This, of course, may require some more manual work from the users (to prepare their environment, or enable certain flags, etc).

If your contributions limit the number of operating systems supported we will request from you to improve it.

#### Documentation

Your contributions should be bundled with related documentation to help users understand how to use the features you introduce.

#### Maintenance

When you contribute code to Netdata, you are automatically accepting that you will be responsible for maintaining that code in the future. So, if users need help, or report bugs, we will invite you to the related github issues to help them or fix the issues or bugs of your contributions.

#### Code Style

The single most important rule when writing code is this: \*check the surrounding code and try to imitate it\*. [Reference](https://developer.gnome.org/programming-guidelines/stable/c-coding-style.html.en)

We use several different languages and have had contributions from several people with different styles. When in doubt, you can check similar existing code.

For C contributions in particular, we try to respect the [Linux kernel style](https://www.kernel.org/doc/html/v4.10/process/coding-style.html), with the following exceptions:

- Use 4 space indentation instead of 8

- We occasionally have multiple statements on a single line (e.g. `if (a) b;`)

- Allow max line length of 120 chars

- Allow opening brace at the end of a function declaration: `function() {`.

- Allow trailing comments

### Your first pull request

There are several guides for pull requests, such as the following:

- <https://thenewstack.io/getting-legit-with-git-and-github-your-first-pull-request/>

- <https://github.com/firstcontributions/first-contributions#first-contributions>

However, it's not always that simple. Our [PR approval process](#pr-approval-process) and the several merges we do every day may cause your fork to get behind the Netdata master. If you worked on something that has changed in the meantime, you will be required to do a git rebase, to bring your fork to the correct state. A very easy to follow guide on how to do it without learning all the intricacies of GitHub can be found [here](https://medium.com/@ruthmpardee/git-fork-workflow-using-rebase-587a144be470)

One thing you will need to do only for your first pull request in Netdata is to accept the CLA. Until you do, the automated check for the CLA acceptance will be showing as failed.

#### PR Guidelines

PR Titles:

- Must follow the [Imperative Mood](https://en.wikipedia.org/wiki/Imperative\_mood)

- Must be no more than ~50 characters (\_longer description in the PR\_)

PR Descriptions:

- Must clearly contain sufficient information regarding the content of the PR, including area/component, test plan, etc.

- Must reference an existing issue.

Some PR title examples:

- Fix bug in Netdata installer for FreeBSD 11.2

- Update docs for other installation methods

- Add new collector for Prometheus endpoints

- Add 4.19 Kernel variant for eBPF

- Fix typo in README

- Refactor code for better maintainability

- etc

The key idea here is to start with a "verb" of what you are doing in the PR.

For good examples have a look at other projects like:

- https://github.com/facebook/react/commits/master

- https://github.com/tensorflow/tensorflow/commits/master

- https://github.com/vuejs/vue/commits/dev

- https://github.com/microsoft/vscode/commits/master

- Also see the Linux Kernel and Git projects as well as good examples.

#### Commit messages when PRs are merged

When a PR gets squashed and merged into master, the title of the commit message (first line) must be the PR title

followed by the PR number.

The body of the commit message should be a short description of the work, preferably taken from the connected issue.

### PR approval process

Each PR automatically [requires a review](https://help.github.com/articles/about-required-reviews-for-pull-requests/) from the code owners specified in `.github/CODEOWNERS`. Depending on the files contained in your PR, several people may be need to approve it.

We also have a series of automated checks running, such as linters to check code quality and QA tests. If you get an error or warning in any of those checks, you will need to click on the link included in the check to identify the root cause, so you can fix it.

One special type of automated check is the "WIP" check. You may add "[WIP]" to the title of the PR, to tell us that the particular request is "Work In Progress" and should not be merged. You're still not done with it, you created it to get some feedback. When you're ready to get the final approvals and get it merged, just remove the "[WIP]" string from the title of your PR and the "WIP" check will pass.

[![analytics](https://www.google-analytics.com/collect?v=1&aip=1&t=pageview&\_s=1&ds=github&dr=https%3A%2F%2Fgithub.com%2Fnetdata%2Fnetdata&dl=https%3A%2F%2Fmy-netdata.io%2Fgithub%2FCONTRIBUTING&\_u=MAC~&cid=5792dfd7-8dc4-476b-af31-da2fdb9f93d2&tid=UA-64295674-3)](<>)