# Contributing to Vagrant

\*\*First:\*\* We like to encourage you to contribute to the repository. If you're unsure or afraid of \_anything\_, just ask or submit the issue or pull request anyways. You won't be yelled at for giving your best effort. The worst that can happen is that you'll be politely asked to change something. We appreciate any sort of contributions, and don't want a wall of rules to get in the way of that.

However, for those individuals who want a bit more guidance on the best way to contribute to the project, read on. This document will cover what we're looking for. By addressing all the points we're looking for, it raises the chances we can quickly merge or address your contributions.

Before opening a new issue or pull request, we do appreciate if you take some time to search for [possible duplicates](https://github.com/hashicorp/vagrant/issues?q=sort%3Aupdated-desc), or similar discussions on [HashiCorp Discuss](https://discuss.hashicorp.com/c/vagrant/24). On GitHub, you can scope searches by labels to narrow things down.

## Issues

### Reporting an Issue

\*\*Tip:\*\* We have provided a [GitHub issue template](https://github.com/hashicorp/vagrant/blob/master/.github/ISSUE\_TEMPLATE/bug-report.md). By respecting the proposed format and filling all the relevant sections, you'll strongly help the Vagrant collaborators to handle your request the best possible way.

### Issue Lifecycle

1. The issue is reported.

2. The issue is verified and categorized by Vagrant collaborator(s). Categorization is done via GitHub tags for different dimensions (like issue type, affected components, pending actions, etc.)

3. Unless it is critical, the issue is left for a period of time, giving outside contributors a chance to address the issue. Later, the issue may be assigned to a Vagrant collaborator and planned for a specific release [milestone](https://github.com/hashicorp/vagrant/milestones)

4. The issue is addressed in a pull request or commit. The issue will be referenced in the commit message so that the code that fixes it is clearly linked.

5. The issue is closed. Sometimes, valid issues will be closed to keep the issue tracker clean. The issue is still indexed and available for future viewers, or can be re-opened if necessary.

## Pull Requests

Thank you for contributing! Here you'll find information on what to include in your Pull Request (“PR” for short) to ensure it is reviewed quickly, and possibly accepted.

Before starting work on a new feature or anything besides a minor bug fix, it is highly recommended to first initiate a discussion with the Vagrant community (either via a GitHub issue or [HashiCorp Discuss](https://discuss.hashicorp.com/c/vagrant/24)). This will save you from wasting days implementing a feature that could be rejected in the end.

No pull request template is provided on GitHub. The expected changes are often already described and validated in an existing issue, that obviously should be referenced. The Pull Request thread should be mainly used for the code review.

\*\*Tip:\*\* Make it small! A focused PR gives you the best chance of having it accepted. Then, repeat if you have more to propose!

### How to prepare

Once you're confident that your upcoming changes will be accepted:

\* In your forked repository, create a topic branch for your upcoming patch.

\* Usually this is based on the master branch.

\* Checkout a new branch based on master; `git checkout -b my-contrib master`

Please avoid working directly on the `master` branch.

\* Make focused commits of logical units and describe them properly.

\* Avoid re-formatting of the existing code

\* Check for unnecessary whitespace with `git diff --check` before committing.

\* If possible, submit tests along with your topic branch. It will help a lot to get your your patch / new feature accepted, and should prevent unwanted breaking changes to silently happen in future developments.

\* Assure nothing is broken by running manual tests, and all the automated tests.

### Submit Changes

\* Push your changes to a topic branch in your fork of the repository.

\* Open a PR to the original repository and choose the right original branch you want to patch (master for most cases).

\* If not done in commit messages (which you really should do) please reference and update your issue with the code changes.

\* Even if you have write access to the repository, do not directly push or merge your own pull requests. Let another team member review your PR and approve.

### Pull Request Lifecycle

1. You are welcome to submit your PR for commentary or review before it is fully completed. Please prefix the title of your PR with "[WIP]" to indicate this. It's also a good idea to include specific questions or items you'd like feedback on.

2. Sign the [HashiCorp CLA](#hashicorp-cla). If you haven't signed the CLA yet, a bot will ask you to do so. You only need to sign the CLA once. If you've already signed the CLA, the CLA status will be green automatically.

3. The PR is categorized by Vagrant collaborator(s), applying GitHub tags similarly to issues triage.

4. Once you believe your PR is ready to be merged, you can remove any

"[WIP]" prefix from the title and a Vagrant collaborator will review.

5. One of the Vagrant collaborators will look over your contribution and either provide comments letting you know if there is anything left to do. We do our best to provide feedback in a timely manner, but it may take some time for us to respond.

6. Once all outstanding comments have been addressed, your contribution will be merged! Merged PRs will be included in the next Vagrant release. The Vagrant contributors will take care of updating the CHANGELOG as they merge.

7. We might decide that a PR should be closed. We'll make sure to provide clear reasoning when this happens.

## HashiCorp CLA

We require all contributors to sign the [HashiCorp CLA](https://www.hashicorp.com/cla).

In simple terms, the CLA affirms that the work you're contributing is original, that you grant HashiCorp permission to use that work (including license to any patents necessary), and that HashiCorp may relicense your work for our commercial products if necessary. Note that this description is a summary and the specific legal terms should be read directly in the [CLA](https://www.hashicorp.com/cla).

The CLA does not change the terms of the standard open source license used by our software such as MPL2 or MIT. You are still free to use our projects within your own projects or businesses, republish modified source, and more. Please reference the appropriate license of this project to learn more.

To sign the CLA, open a pull request as usual. If you haven't signed the CLA yet, a bot will respond with a link asking you to sign the CLA. We cannot merge any pull request until the CLA is signed. You only need to sign the CLA once. If you've signed the CLA before, the bot will not respond to your PR and your PR will be allowed to merge.

# Additional Resources

\* [HashiCorp Community Guidelines](https://www.hashicorp.com/community-guidelines)

\* [General GitHub documentation](https://help.github.com/)

\* [GitHub pull request documentation](https://help.github.com/send-pull-requests/)