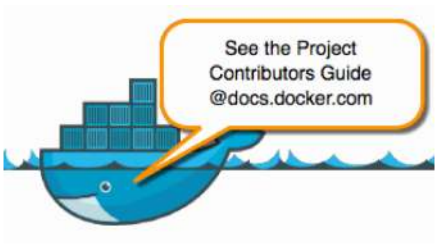
# Contributing to Docker

Want to hack on Docker? Awesome! We have a contributor's guide that explains setting up a Docker development environment and the contribution process.



This page contains information about reporting issues as well as some tips and guidelines useful to experienced open source contributors. Finally, make sure you read our community guidelines before you start participating.

## Topics

## Reporting security issues

The Docker maintainers take security seriously. If you discover a security issue, please bring it to their attention right away!

Please **DO NOT** file a public issue, instead send your report privately to security@docker.com,

Security reports are greatly appreciated and we will publicly thank you for it. We also like to send gifts—if you're into Docker schwag, make sure to let us know. We currently do not offer a paid security bounty program, but are not ruling it out in the future.

## Reporting other issues

A great way to contribute to the project is to send a detailed report when you encounter an issue. We always appreciate a well-written, thorough bug report, and will thank you for it!

Check that our issue database doesn't already include that problem or suggestion before submitting an issue. If you find a match, add a quick "+1" or "I have this problem too." Doing this helps prioritize the most common problems and requests.

When reporting issues, please include your host OS (Ubuntu 12.04, Fedora 19, etc). Please include:

- The output of `uname -a` .
- The output of `docker version` .
- The output of `docker -D info` .

Please also include the steps required to reproduce the problem if possible and applicable. This information will help us review and fix your issue faster.

**Issue Report Template**:

```
Description of problem:


`docker version`:


`docker info`:


`uname -a`:


Environment details (AWS, VirtualBox, physical, etc.):


How reproducible:


Steps to Reproduce:
1.
2.
3.


Actual Results:


Expected Results:


Additional info:
```

## Quick contribution tips and guidelines

This section gives the experienced contributor some tips and guidelines.

### Pull requests are always welcome

Not sure if that typo is worth a pull request? Found a bug and know how to fix it? Do it! We will appreciate it. Any significant improvement should be documented as a GitHub issue before anybody starts working on it.

We are always thrilled to receive pull requests. We do our best to process them quickly. If your pull request is not accepted on the first try, don't get discouraged! Our contributor's guide explains the review process we use for simple changes.

### Design and cleanup proposals

You can propose new designs for existing Docker features. You can also design entirely new features. We really appreciate contributors who want to refactor or otherwise cleanup our project. For information on making these types of contributions, see the advanced contribution section in the contributors guide.

We try hard to keep Docker lean and focused. Docker can't do everything for everybody. This means that we might decide against incorporating a new feature. However, there might be a way to implement that feature *on top of* Docker.

### Talking to other Docker users and contributors

| | |
|---|---|
| Internet Relay Chat (IRC) | IRC a direct line to our most knowledgeable Docker users; we have both the `#docker` and `#docker-dev` group on **irc.freenode.net**. IRC is a rich chat protocol but it can overwhelm new users. You can search our chat archives.<br><br>Read our IRC quickstart guide for an easy way to get started. |
| | |

| | |
|---|---|
| Google Groups | There are two groups. Docker-user is for people using Docker containers. The docker-dev group is for contributors and other people contributing to the Docker project. |
| Twitter | You can follow Docker's Twitter feed to get updates on our products. You can also tweet us questions or just share blogs or stories. |
| Stack Overflow | Stack Overflow has over 7000K Docker questions listed. We regularly monitor Docker questions and so do many other knowledgeable Docker users. |

## Conventions

Fork the repository and make changes on your fork in a feature branch:

- If it's a bug fix branch, name it XXXX-something where XXXX is the number of the issue.
- If it's a feature branch, create an enhancement issue to announce your intentions, and name it XXXX-something where XXXX is the number of the issue.

Submit unit tests for your changes. Go has a great test framework built in; use it! Take a look at existing tests for inspiration. Run the full test suite on your branch before submitting a pull request.

Update the documentation when creating or modifying features. Test your documentation changes for clarity, concision, and correctness, as well as a clean documentation build. See our contributors guide for our style guide and instructions on building the documentation.

Write clean code. Universally formatted code promotes ease of writing, reading, and maintenance. Always run `gofmt -s -w file.go` on each changed file before committing your changes. Most editors have plug-ins that do this automatically.

Pull request descriptions should be as clear as possible and include a reference to all the issues that they address.

Commit messages must start with a capitalized and short summary (max. 50 chars) written in the imperative, followed by an optional, more detailed explanatory text which is separated from the summary by an empty line.

Code review comments may be added to your pull request. Discuss, then make the suggested modifications and push additional commits to your feature branch. Post a comment after pushing. New commits show up in the pull request automatically, but the reviewers are notified only when you comment.

Pull requests must be cleanly rebased on top of master without multiple branches mixed into the PR.

**Git tip**: If your PR no longer merges cleanly, use `rebase master` in your feature branch to update your pull request rather than `merge master` .

Before you make a pull request, squash your commits into logical units of work using `git rebase -i` and `git push -f` . A logical unit of work is a consistent set of patches that should be reviewed together: for example, upgrading the version of a vendored dependency and taking advantage of its now available new feature constitute two separate units of work. Implementing a new function and calling it in another file constitute a single logical unit of work. The very high majority of submissions should have a single commit, so if in doubt: squash down to one.

After every commit, make sure the test suite passes. Include documentation changes in the same pull request so that a revert would remove all traces of the feature or fix.

Include an issue reference like `Closes #XXXX` or `Fixes #XXXX` in commits that close an issue. Including references automatically closes the issue on a merge.

Please do not add yourself to the `AUTHORS` file, as it is regenerated regularly from the Git history.

## Merge approval

Docker maintainers use LGTM (Looks Good To Me) in comments on the code review to indicate acceptance.

A change requires LGTMs from an absolute majority of the maintainers of each component affected. For example, if a change affects `docs/` and `registry/` , it needs an absolute majority from the maintainers of `docs/` AND, separately, an absolute majority of the maintainers of `registry/` .

For more details, see the MAINTAINERS page.

## Sign your work

The sign-off is a simple line at the end of the explanation for the patch. Your signature certifies that you wrote the patch or otherwise have the right to pass it on as an open-source patch. The rules are pretty simple: if you can certify the below (from developercertificate.org):

```
Developer Certificate of Origin
Version 1.1

Copyright (C) 2004, 2006 The Linux Foundation and its contributors.
660 York Street, Suite 102,
San Francisco, CA 94110 USA

Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.

Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I
    have the right to submit it under the open source license
    indicated in the file; or

(b) The contribution is based upon previous work that, to the best
    of my knowledge, is covered under an appropriate open source
    license and I have the right under that license to submit that
    work with modifications, whether created in whole or in part
    by me, under the same open source license (unless I am
    permitted to submit under a different license), as indicated
    in the file; or

(c) The contribution was provided directly to me by some other
    person who certified (a), (b) or (c) and I have not modified
    it.

(d) I understand and agree that this project and the contribution
    are public and that a record of the contribution (including all
    personal information I submit with it, including my sign-off) is
    maintained indefinitely and may be redistributed consistent with
    this project or the open source license(s) involved.
```

Then you just add a line to every git commit message:

```
Signed-off-by: Joe Smith <joe.smith@email.com>
```

Use your real name (sorry, no pseudonyms or anonymous contributions.)

If you set your `user.name` and `user.email` git configs, you can sign your commit automatically with `git commit -s` .

Note that the old-style `Docker-DCO-1.1-Signed-off-by: ...` format is still accepted, so there is no need to update outstanding pull requests to the new format right away, but please do adjust your processes for future contributions.

## How can I become a maintainer?

- Step 1: Learn the component inside out
- Step 2: Make yourself useful by contributing code, bug fixes, support etc.
- Step 3: Volunteer on the IRC channel (#docker at Freenode)
- Step 4: Propose yourself at a scheduled docker meeting in #docker-dev

Don't forget: being a maintainer is a time investment. Make sure you will have time to make yourself available. You don't have to be a maintainer to make a difference on the project!

## IRC meetings

There are two monthly meetings taking place on #docker-dev IRC to accommodate all timezones. Anybody can propose a topic for discussion prior to the meeting.

If you feel the conversation is going off-topic, feel free to point it out.

For the exact dates and times, have a look at the irc-minutes repo. The minutes also contain all the notes from previous meetings.

## Docker community guidelines

We want to keep the Docker community awesome, growing and collaborative. We need your help to keep it that way. To help with this we've come up with some general guidelines for the community as a whole:

- Be nice: Be courteous, respectful and polite to fellow community members: no regional, racial, gender, or other abuse will be tolerated. We like nice people way better than mean ones!
- Encourage diversity and participation: Make everyone in our community feel welcome, regardless of their background and the extent of their contributions, and do everything possible to encourage participation in our community.
- Keep it legal: Basically, don't get us in trouble. Share only content that you own, do not share private or sensitive information, and don't break the law.
- Stay on topic: Make sure that you are posting to the correct channel and avoid off-topic discussions. Remember when you update an issue or respond to an email you are potentially sending to a large number of people. Please consider this before you update. Also remember that nobody likes spam.

### Guideline violations — 3 strikes method

The point of this section is not to find opportunities to punish people, but we do need a fair way to deal with people who are making our community suck.

1. First occurrence: We'll give you a friendly, but public reminder that the behavior is inappropriate according to our guidelines.
2. Second occurrence: We will send you a private message with a warning that any additional violations will result in removal from the community.
3. Third occurrence: Depending on the violation, we may need to delete or ban your account.

**Notes:**

- Obvious spammers are banned on first occurrence. If we don't do this, we'll have spam all over the place.
- Violations are forgiven after 6 months of good behavior, and we won't hold a grudge.
- People who commit minor infractions will get some education, rather than hammering them in the 3 strikes process.
- The rules apply equally to everyone in the community, no matter how much you've contributed.
- Extreme violations of a threatening, abusive, destructive or illegal nature will be addressed immediately and are not subject to 3 strikes or forgiveness.
- Contact abuse@docker.com to report abuse or appeal violations. In the case of appeals, we know that mistakes happen, and we'll work with you to come up with a fair solution if there has been a misunderstanding.