# Contributor Guidelines

(Please talk to people on the mailing list before you change this page, see our section on [how to get in touch](https://github.com/ether/etherpad-lite#get-in-touch))

## Pull requests

\* the commit series in the PR should be \_linear\_ (it \*\*should not contain merge commits\*\*). This is necessary because we want to be able to [bisect](https://en.wikipedia.org/wiki/Bisection\_(software\_engineering)) bugs easily. Rewrite history/perform a rebase if necessary

\* PRs should be issued against the \*\*develop\*\* branch: we never pull directly into \*\*master\*\*

\* PRs \*\*should not have conflicts\*\* with develop. If there are, please resolve them rebasing and force-pushing

\* when preparing your PR, please make sure that you have included the relevant \*\*changes to the documentation\*\* (preferably with usage examples)

\* contain meaningful and detailed \*\*commit messages\*\* in the form:

```

submodule: description

longer description of the change you have made, eventually mentioning the

number of the issue that is being fixed, in the form: Fixes #someIssueNumber

```

\* if the PR is a \*\*bug fix\*\*:

\* the first commit in the series must be a test that shows the failure

\* subsequent commits will fix the bug and make the test pass

\* the final commit message should include the text `Fixes: #xxx` to link it to its bug report

\* think about stability: code has to be backwards compatible as much as possible. Always \*\*assume your code will be run with an older version of the DB/config file\*\*

\* if you want to remove a feature, \*\*deprecate it instead\*\*:

\* write an issue with your deprecation plan

\* output a `WARN` in the log informing that the feature is going to be removed

\* remove the feature in the next version

\* if you want to add a new feature, put it under a \*\*feature flag\*\*:

\* once the new feature has reached a minimal level of stability, do a PR for it, so it can be integrated early

\* expose a mechanism for enabling/disabling the feature

\* the new feature should be \*\*disabled\*\* by default. With the feature disabled, the code path should be exactly the same as before your contribution. This is a \_\_necessary condition\_\_ for early integration

\* think of the PR not as something that \_\_you wrote\_\_, but as something that \_\_someone else is going to read\_\_. The commit series in the PR should tell a novice developer the story of your thoughts when developing it

## How to write a bug report

\* Please be polite, we all are humans and problems can occur.

\* Please add as much information as possible, for example

\* client os(s) and version(s)

\* browser(s) and version(s), is the problem reproducible on different clients

\* special environments like firewalls or antivirus

\* host os and version

\* npm and nodejs version

\* Logfiles if available

\* steps to reproduce

\* what you expected to happen

\* what actually happened

\* Please format logfiles and code examples with markdown see github Markdown help below the issue textarea for more information.

If you send logfiles, please set the loglevel switch DEBUG in your settings.json file:

```

/\* The log level we are using, can be: DEBUG, INFO, WARN, ERROR \*/

"loglevel": "DEBUG",

```

The logfile location is defined in startup script or the log is directly shown in the commandline after you have started etherpad.

## General goals of Etherpad

To make sure everybody is going in the same direction:

\* easy to install for admins and easy to use for people

\* easy to integrate into other apps, but also usable as standalone

\* lightweight and scalable

\* extensible, as much functionality should be extendable with plugins so changes don't have to be done in core.

Also, keep it maintainable. We don't wanna end up as the monster Etherpad was!

## How to work with git?

\* Don't work in your master branch.

\* Make a new branch for every feature you're working on. (This ensures that you can work you can do lots of small, independent pull requests instead of one big one with complete different features)

\* Don't use the online edit function of github (this only creates ugly and not working commits!)

\* Try to make clean commits that are easy readable (including descriptive commit messages!)

\* Test before you push. Sounds easy, it isn't!

\* Don't check in stuff that gets generated during build or runtime

\* Make small pull requests that are easy to review but make sure they do add value by themselves / individually

## Coding style

\* Do write comments. (You don't have to comment every line, but if you come up with something that's a bit complex/weird, just leave a comment. Bear in mind that you will probably leave the project at some point and that other people will read your code. Undocumented huge amounts of code are worthless!)

\* Never ever use tabs

\* Indentation: JS/CSS: 2 spaces; HTML: 4 spaces

\* Don't overengineer. Don't try to solve any possible problem in one step, but try to solve problems as easy as possible and improve the solution over time!

\* Do generalize sooner or later! (if an old solution, quickly hacked together, poses more problems than it solves today, refactor it!)

\* Keep it compatible. Do not introduce changes to the public API, db schema or configurations too lightly. Don't make incompatible changes without good reasons!

\* If you do make changes, document them! (see below)

\* Use protocol independent urls "//"

## Branching model / git workflow

see git flow http://nvie.com/posts/a-successful-git-branching-model/

### `master` branch

\* the stable

\* This is the branch everyone should use for production stuff

### `develop`branch

\* everything that is READY to go into master at some point in time

\* This stuff is tested and ready to go out

### release branches

\* stuff that should go into master very soon

\* only bugfixes go into these (see http://nvie.com/posts/a-successful-git-branching-model/ for why)

\* we should not be blocking new features to develop, just because we feel that we should be releasing it to master soon. This is the situation that release branches solve/handle.

### hotfix branches

\* fixes for bugs in master

### feature branches (in your own repos)

\* these are the branches where you develop your features in

\* If it's ready to go out, it will be merged into develop

Over the time we pull features from feature branches into the develop branch. Every month we pull from develop into master. Bugs in master get fixed in hotfix branches. These branches will get merged into master AND develop. There should never be commits in master that aren't in develop

## Documentation

The docs are in the `doc/` folder in the git repository, so people can easily find the suitable docs for the current git revision.

Documentation should be kept up-to-date. This means, whenever you add a new API method, add a new hook or change the database model, pack the relevant changes to the docs in the same pull request.

You can build the docs e.g. produce html, using `make docs`. At some point in the future we will provide an online documentation. The current documentation in the github wiki should always reflect the state of `master` (!), since there are no docs in master, yet.

## Testing

Front-end tests are found in the `tests/frontend/` folder in the repository. Run them by pointing your browser to `<yourdomainhere>/tests/frontend`.

Back-end tests can be run from the `src` directory, via `npm test`.

## Things you can help with

Etherpad is much more than software. So if you aren't a developer then worry not, there is still a LOT you can do! A big part of what we do is community engagement. You can help in the following ways

\* Triage bugs (applying labels) and confirming their existence

\* Testing fixes (simply applying them and seeing if it fixes your issue or not) - Some git experience required

\* Notifying large site admins of new releases

\* Writing Changelogs for releases

\* Creating Windows packages

\* Creating releases

\* Bumping dependencies periodically and checking they don't break anything

\* Write proposals for grants

\* Co-Author and Publish CVEs

\* Work with SFC to maintain legal side of project

\* Maintain TODO page - https://github.com/ether/etherpad-lite/wiki/TODO#IMPORTANT\_TODOS