# Community Contribution Guidelines

As the creators and maintainers of this project, we want to ensure that the project lives and continues to grow. Not blocked by any singular person's computer time. One of the simplest ways of doing this is by encouraging a larger set of shallow contributors. Through this, we hope to mitigate the problems of a project that needs updates, but there is no-one who has the power to do so.

#### Development Process

We maintain two permanent, protected branches: `master` and `development`.

`master` is for the current release. It has already been distributed to jcenter and can't be modified at this point (except for docs that may be incorrect for the current release!).

`development` is where we stage work for the \*next\* release. Pull requests should be directed to this branch.

When working on a new feature or fix that may span multiple commits, please do so in a feature branch (ex: `feature/my\_cool\_thing`). Please clean up these feature branches once merged into `development`.

When a new version is ready to be released, please create a pull request to merge `development` into `master`, named something like "Release 10.0". Then we can have some final discussion before we merge it into `master` and push the release out to the public.

Since `development` is a \*shared\* branch, it is important not to ever rebase this branch onto `master`. If a bug fix is applied to `master` it can be merged into `development` using good old simple `git checkout development && git merge master`. Yes this will clutter the history a little bit, but it also provides important context to know how/when a patch was applied. Merge commits can be considered necessary historical data, not warts on an idealized history graph.

#### Testing

To run tests locally, just run `./gradlew test`. Tests should always be run before pushing anything to the repo.

#### Ownership

If you get a merged pull-request of substance (ie not just a typo fix), then you are eligible for push access to this repo. Simply request access via a new GitHub issue.

Offhand, it is easy to imagine that this would make code quality suffer, but in reality it offers fresh perspectives to the codebase and encourages ownership from people who are depending on the project. If you are building a project that relies on this codebase, then you probably have the skills to improve it and offer valuable feedback. At the end of the day, there isn't too much risk in this, as the `master` branch is still locked, and jcenter access is still being restricted.

Everyone comes in with their own perspective on what a project could/should look like, and encouraging discussion can help expose good ideas sooner.

#### Why do we give out push access?

It can be overwhelming to be offered the chance to wipe the source code for a project. Do not worry, we do not let you push to master. All code is peer-reviewed, and we have the convention that someone other than the submitter should merge non-trivial pull requests.

As a contributor, you can merge other people's pull requests, or other contributors can merge yours. You will not be assigned a pull request, but you are welcome to jump in and take a code review on topics that interest you.

This project is not continuously deployed, there is space for debate after review too. Offering everyone the chance to revert, or make an amending pull request. If it feels right, merge.

#### How can we help you get comfortable contributing?

It is normal for a first pull request to be a potential fix for a problem, and moving on from there to helping the project's direction can be difficult. We will try to help contributors cross that barrier by offering good first step issues. These issues can be fixed without feeling like you are stepping on toes. Ideally, these are non-critical issues that are well defined. They will be purposely avoided by mature contributors to the project, to make space for others.

We aim to keep all project discussion inside GitHub issues. This is to make sure valuable discussion is accessible via search. If you have questions about how to use the library, or how the project is running - GitHub issues are the goto tool for this project.

#### Our expectations on you as a contributor

To quote [@alloy](https://github.com/alloy) from [this issue](https://github.com/Moya/Moya/issues/135):

> Do not ever feel bad for not contributing to open source.

We want contributors to provide ideas, keep the ship shipping and to take some of the load from others. It is non-obligatory; we’re here to get things done in an enjoyable way. :trophy:

The fact that you will have push access will allow you to:

- Avoid having to fork the project if you want to submit other pull requests, as you will be able to create branches directly on the project.

- Help triage issues and merge pull requests.

- Pick up the project if other maintainers move their focus elsewhere.

It is up to you to use those superpowers or not though ?

If someone submits a pull request that is not perfect, and you are reviewing, it is better to think about the PR's motivation rather than the specific implementation. Having braces on the wrong line should not be a blocker. Though we do want to keep test coverage high, we will work with you to figure that out together.

#### What about if you have problems that cannot be discussed in a public issue?

[Eric Kuck](https://github.com/erickuck) has a contactable email on his GitHub profile, and is happy to talk about any problems.

#### This is a different way to handle open source! Where did it come from?

The original source of this document can be found at [https://github.com/moya/contributors](https://github.com/moya/contributors).