# Contributing

:raised\_hands::tada: First of all, thanks for taking the time to contribute! :tada::raised\_hands:

The following is a set of guidelines for contributing to @material-ui/pickers. The purpose of these

guidelines is to maintain a high quality of code \_and\_ traceability. Please respect these

guidelines.

## Setting up

Please make sure that we are using the `next` branch for active development. So your branches must be created from \_next\_ and not from ~~master~~. Here is a short step-by-step guide on how to get started:

1. Fork the material-ui-pickers repository on Github

2. Clone your fork to your local machine `git clone git@github.com:<yourname>/material-ui-pickers.git`

3. Checkout next `git checkout next`

4. Create a branch `git checkout -b my-topic-branch`

5. Make your changes, lint, run tests, then push to GitHub with `git push --set-upstream origin my-topic-branch`.

6. Visit GitHub and make your pull request.

If you have an existing local repository, please update it before you start, to minimise the chance of merge conflicts.

```sh

git remote add upstream git@github.com:mui-org/material-ui-pickers.git

git checkout next

git pull upstream next

git checkout -b my-topic-branch

```

## Dev environment

We are using yarn for development and linking, so you need to make

```sh

yarn

yarn docs:dev

```

to start the development environment and to be sure the `lib` folder is linked properly to the `docs` and your changes will be applied to the docs website.

## General

This repository uses tests and a linter as automatic tools to maintain the quality of the code.

These two tasks are run locally on your machine before every commit (as a pre-commit git hook). If any test fails or the linter gives an error the commit will not be created. They are also run on

a Travis CI machine when you create a pull request, and the PR will not be merged unless Travis

says all tests and the linting pass.

## Git Commit Messages

- Use the imperative mood ("Move pointer to..." not "Moves pointer to...")

- Think of it as you are \_commanding\_ what your commit is doing

- Git itself uses the imperative whenever it creates a commit on your behalf, so it makes sense

for you to use it too

- Use the body to explain \_what\_ and \_why\_

- If the commit is non-trivial, please provide more detailed information in the commit body

message

- \_How\_ you made the change is visible in the code and is therefore rarely necessary to include

in the commit body message, but \_why\_ you made the change is often harder to guess and is

therefore useful to include in the commit body message

[Here's a nice blog post on how to write great git messages.](http://chris.beams.io/posts/git-commit/)

## Pull Requests

- Follow the current code style

- Write tests for your changes

- Document your changes in the README if it's needed

- End files with a newline

- There's no need to create a new build for each pull request, we (the maintainers) do this when we

release a new version

## Issues

- Please be descriptive when you fill in the issue template, this will greatly help us maintainers

in helping you which will lead to your issue being resolved faster

- Feature requests are very welcomed, but not every feature that is requested can be guaranteed

to be implemented