# Contributing to Compose

Compose is a part of the Docker project, and follows the same rules and

principles. Take a read of [Docker's contributing guidelines](https://github.com/docker/docker/blob/master/CONTRIBUTING.md)

to get an overview.

## TL;DR

Pull requests will need:

- Tests

- Documentation

- [To be signed off](https://github.com/docker/docker/blob/master/CONTRIBUTING.md#sign-your-work)

- A logical series of [well written commits](https://github.com/alphagov/styleguides/blob/master/git.md)

## Development environment

If you're looking contribute to Compose

but you're new to the project or maybe even to Python, here are the steps

that should get you started.

1. Fork [https://github.com/docker/compose](https://github.com/docker/compose)

to your username.

2. Clone your forked repository locally `git clone git@github.com:yourusername/compose.git`.

3. You must [configure a remote](https://help.github.com/articles/configuring-a-remote-for-a-fork/) for your fork so that you can [sync changes you make](https://help.github.com/articles/syncing-a-fork/) with the original repository.

4. Enter the local directory `cd compose`.

5. Set up a development environment by running `python setup.py develop`. This

will install the dependencies and set up a symlink from your `docker-compose`

executable to the checkout of the repository. When you now run

`docker-compose` from anywhere on your machine, it will run your development

version of Compose.

## Install pre-commit hooks

This step is optional, but recommended. Pre-commit hooks will run style checks

and in some cases fix style issues for you, when you commit code.

Install the git pre-commit hooks using [tox](https://tox.readthedocs.io) by

running `tox -e pre-commit` or by following the

[pre-commit install guide](http://pre-commit.com/#install).

To run the style checks at any time run `tox -e pre-commit`.

## Submitting a pull request

See Docker's [basic contribution workflow](https://docs.docker.com/v17.06/opensource/code/#code-contribution-workflow) for a guide on how to submit a pull request for code.

## Documentation changes

Issues and pull requests to update the documentation should be submitted to the [docs repo](https://github.com/docker/docker.github.io). You can learn more about contributing to the documentation [here](https://docs.docker.com/opensource/#how-to-contribute-to-the-docs).

## Running the test suite

Use the test script to run linting checks and then the full test suite against

different Python interpreters:

$ script/test/default

Tests are run against a Docker daemon inside a container, so that we can test

against multiple Docker versions. By default they'll run against only the latest

Docker version - set the `DOCKER\_VERSIONS` environment variable to "all" to run

against all supported versions:

$ DOCKER\_VERSIONS=all script/test/default

Arguments to `script/test/default` are passed through to the `tox` executable, so

you can specify a test directory, file, module, class or method:

$ script/test/default tests/unit

$ script/test/default tests/unit/cli\_test.py

$ script/test/default tests/unit/config/config\_test.py::ConfigTest

$ script/test/default tests/unit/config/config\_test.py::ConfigTest::test\_load

## Finding things to work on

[Issues marked with the `exp/beginner` label](https://github.com/docker/compose/issues?q=is%3Aopen+is%3Aissue+label%3Aexp%2Fbeginner) are a good starting point for people looking to make their first contribution to the project.