# Contribute Code

You are welcome to contribute to project PaddlePaddle. To contribute to PaddlePaddle, you have to agree with the

[PaddlePaddle Contributor License Agreement](https://gist.github.com/XiaoguangHu01/75018ad8e11af13df97070dd18ae6808).

We sincerely appreciate your contribution. This document explains our workflow and work style.

## Workflow

PaddlePaddle uses this [Git branching model](http://nvie.com/posts/a-successful-git-branching-model/). The following steps guide usual contributions.

1. Fork

Our development community has been growing fastly; it doesn't make sense for everyone to write into the official repo. So, please file Pull Requests from your fork. To make a fork, just head over to the GitHub page and click the ["Fork" button](https://help.github.com/articles/fork-a-repo/).

1. Clone

To make a copy of your fork to your local computers, please run

```bash

git clone https://github.com/your-github-account/paddle

cd paddle

```

1. Create the local feature branch

For daily works like adding a new feature or fixing a bug, please open your feature branch before coding:

```bash

git checkout -b my-cool-stuff

```

1. Commit

Before issuing your first `git commit` command, please install [`pre-commit`](http://pre-commit.com/) by running the following commands:

```bash

pip install pre-commit

pre-commit install

```

Our pre-commit configuration requires clang-format 3.8 for auto-formating C/C++ code and yapf for Python.

Once installed, `pre-commit` checks the style of code and documentation in every commit. We will see something like the following when you run `git commit`:

```

? git commit

CRLF end-lines remover...............................(no files to check)Skipped

yapf.................................................(no files to check)Skipped

Check for added large files..............................................Passed

Check for merge conflicts................................................Passed

Check for broken symlinks................................................Passed

Detect Private Key...................................(no files to check)Skipped

Fix End of Files.....................................(no files to check)Skipped

clang-formater.......................................(no files to check)Skipped

[my-cool-stuff c703c041] add test file

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 233

```

NOTE: The `yapf` installed by `pip install pre-commit` and `conda install -c conda-forge pre-commit` is slightly different. Paddle developers use `pip install pre-commit`.

1. Build and test

Users can build PaddlePaddle natively on Linux and Mac OS X. But to unify the building environment and to make it easy for debugging, the recommended way is [using Docker](https://github.com/PaddlePaddle/Paddle/blob/develop/doc/howto/dev/build\_en.md).

1. Keep pulling

An experienced Git user pulls from the official repo often -- daily or even hourly, so they notice conflicts with others work early, and it's easier to resolve smaller conflicts.

```bash

git remote add upstream https://github.com/PaddlePaddle/Paddle

git pull upstream develop

```

1. Push and file a pull request

You can "push" your local work into your forked repo:

```bash

git push origin my-cool-stuff

```

The push allows you to create a pull request, requesting owners of this [official repo](https://github.com/PaddlePaddle/Paddle) to pull your change into the official one.

To create a pull request, please follow [these steps](https://help.github.com/articles/creating-a-pull-request/).

If your change is for fixing an issue, please write ["Fixes <issue-URL>"](https://help.github.com/articles/closing-issues-using-keywords/) in the description section of your pull request. Github would close the issue when the owners merge your pull request.

Please remember to specify some reviewers for your pull request. If you don't know who are the right ones, please follow Github's recommendation.

1. Delete local and remote branches

To keep your local workspace and your fork clean, you might want to remove merged branches:

```bash

git push origin :my-cool-stuff

git checkout develop

git pull upstream develop

git branch -d my-cool-stuff

```

### Code Review

- Please feel free to ping your reviewers by sending them the URL of your pull request via IM or email. Please do this after your pull request passes the CI.

- Please answer reviewers' every comment. If you are to follow the comment, please write "Done"; please give a reason otherwise.

- If you don't want your reviewers to get overwhelmed by email notifications, you might reply their comments by [in a batch](https://help.github.com/articles/reviewing-proposed-changes-in-a-pull-request/).

- Reduce the unnecessary commits. Some developers commit often. It is recommended to append a sequence of small changes into one commit by running `git commit --amend` instead of `git commit`.

## Coding Standard

### Code Style

Our C/C++ code follows the [Google style guide](http://google.github.io/styleguide/cppguide.html).

Our Python code follows the [PEP8 style guide](https://www.python.org/dev/peps/pep-0008/).

Our build process helps to check the code style. In [`build.sh`](https://github.com/PaddlePaddle/Paddle/blob/b84e8226514b8bb4405c3c28e54aa5077193d179/paddle/scripts/docker/build.sh#L42), the entry point of our [builder Docker image](https://github.com/PaddlePaddle/Paddle/blob/b84e8226514b8bb4405c3c28e54aa5077193d179/Dockerfile#L88), the CMake argument `WITH\_STYLE\_CHECK` is set to `ON` by default. This flag is on

Please install pre-commit, which automatically reformat the changes to C/C++ and Python code whenever we run `git commit`. To check the whole codebase, we can run the command `pre-commit run -a`, as in the [`check\_style.sh` file](https://github.com/PaddlePaddle/Paddle/blob/b84e8226514b8bb4405c3c28e54aa5077193d179/paddle/scripts/travis/check\_style.sh#L30), which is invoked by [our Travis CI configuration](https://github.com/PaddlePaddle/Paddle/blob/b84e8226514b8bb4405c3c28e54aa5077193d179/.travis.yml#L43).

### Unit Tests

Please remember to add related unit tests.

- For C/C++ code, please follow [`google-test` Primer](https://github.com/google/googletest/blob/master/googletest/docs/primer.md) .

- For Python code, please use [Python's standard `unittest` package](http://pythontesting.net/framework/unittest/unittest-introduction/).

### Writing Logs

We use [glog](https://github.com/google/glog) for logging in our C/C++ code.

For general information, please use `LOG`. For debug information, please use [`VLOG`](http://htmlpreview.github.io/?https://github.com/google/glog/blob/master/doc/glog.html#verbose). The reason is at [here](https://groups.google.com/a/chromium.org/d/msg/chromium-dev/3NDNd1KzXeY/AZKMMx37fdQJ).

`VLOG` requires a \*verbose level\* parameter. For example:

```c++

VLOG(3) << "Operator FC is taking " << num\_inputs << "inputs."

```

When we run a PaddlePaddle application or test, we can specify a verbose threshold. For example:

```bash

GLOG\_vmodule=buddy\_allocator=2 \

GLOG\_v=10 \

python \

../python/paddle/v2/framework/tests/test\_recurrent\_op.py

```

This will enable VLOG messages generated by `buddy\_allocator.{h,cc}` and in the verbose range of 0 to 3, so you will see above example VLOG message, which is in level 3. This suggests that we output overall messages in lower verbose levels, so they display with higher probability. When coding C++, please follow the verbose level convention as follows:

- verbose level 1: [framework](https://github.com/PaddlePaddle/Paddle/tree/develop/paddle/fluid/framework)

- verbose level 3: [operators](https://github.com/PaddlePaddle/Paddle/tree/develop/paddle/fluid/operators)

- verbose level 5: [memory](https://github.com/PaddlePaddle/Paddle/tree/develop/paddle/fluid/memory), [platform](https://github.com/PaddlePaddle/Paddle/tree/develop/paddle/fluid/platform)

- verbose level 7: [math](https://github.com/PaddlePaddle/Paddle/tree/develop/paddle/fluid/operators/math/)