## How to Contribute to Apollo

### Contributor License Agreements

You are welcome to contribute to project Apollo. To contribute to apollo, you have to agree with the [Apollo individual contributor license agreement](

https://gist.githubusercontent.com/startcode/f5ccf8887bfc7727a0ae05bf0d601e30/raw/029a11300e987e34a29a9d247ac30caa7f6741a7/Apollo\_Individual\_Contributor\_License\_Agreement).

### How do I start contributing

\* You can follow the standard [Github approach](https://help.github.com/articles/using-pull-requests/) to contribute code.

\* There are [issues with label "help wanted"](https://github.com/ApolloAuto/apollo/issues?utf8=%E2%9C%93&q=label%3A%22Type%3A+Help+wanted%22+) that are best to help you get started.

\* If you are currently working on an issue, leave a message to let people know that you are working on it.

\* Before sending in your pull request for

[review](https://github.com/ApolloAuto/apollo/pulls),

make sure your changes follow the guidelines mentioned below, namely: license, testing and coding style guidelines.

#### License

For each new file, please include a license at the top of the file.

\* C++ code License example [util.h](modules/common/util/util.h);

\* Python code License example [process.py](modules/tools/vehicle\_calibration/process.py);

\* Bash code License example [apollo\_base.sh](scripts/apollo\_base.sh);

#### Testing

Please include unit tests for the contributed code to prove that your code works correctly,

and make sure that your code does not break existing tests. Test files are always named to end with `\_test.cc`, and the test target names in the BUILD file always end with `test`.

Here is an example test file [file\_test.cc](cyber/common/file\_test.cc).

You can use command `bash apollo.sh test` to run all unit tests.

#### Coding style

\* \*\*C/C++ coding style\*\*: Apollo adopted the [Google C++ Style Guide](https://google.github.io/styleguide/cppguide.html). Make sure your code conforms to this style guide. You can use command `bash apollo.sh lint` to check if your code has any style problem.

\* \*\*Python coding style\*\*: Apollo adopted the [Google Python Style Guide](https://google.github.io/styleguide/pyguide.html). You can use the [yapf](https://github.com/google/yapf) command `yapf -i --style='{based\_on\_style: google}' foo.py` to format a file foo.py.

\* \*\*Apollo best coding practice\*\*: Please also refer to [Apollo Best Coding Practice](docs/technical\_tutorial/best\_coding\_practice.md) for more coding practice disciplines.

\* \*\*BUILD coding style\*\* : you can use command `bash apollo.sh buildify` to format your BUILD files before you submit.

\* \*\*Best coding practice\*\*: Follow [this guide](docs/technical\_tutorial/best\_coding\_practice.md)

#### Documentation

If your code is not straightforward for other contributors to understand, it is recommended to implement the code in a clear and efficient way, and provide sufficient comments and documentation.

Apollo uses doxygen to help generate formatted API Document with command `bash apollo.sh doc`.

Document your code following this guide [How to document code](docs/howto/how\_to\_document\_code.md).

#### Commit Message

The first line of commit message should be a one-line summary of the change.

A paragraph can be added following the summary to clearly explain the details of the change.

If your code fixed an issue, add the issue number to your commit message.

An example of a good commit message is:

> Replaced algorithm A with algorithm B in apollo/modules/control.

>

> Algorithm B is faster than A because it uses binary search. The runtime is reduced from O(N) to O(log(N)).

>

> Fixes #1234

### Before Creating Pull Request

After you finish your code and ready to create a Pull Request, please make sure your

change don't break build/test/lint by running `bash apollo.sh check`, which is

equivalent to a combination of `bash apollo.sh build`, `bash apollo.sh test` and

`bash apollo.sh lint`.