## Contributing to apollo

Apollo is released under the non-restrictive Apache 2.0 license, and follows a very standard Github development process, using Github tracker for issues and merging pull requests into master. If you want to contribute even something trivial please do not hesitate, but follow the guidelines below.

### Sign the Contributor License Agreement

Before we accept a non-trivial patch or pull request we will need you to sign the Contributor License Agreement. Signing the contributor’s agreement does not grant anyone commit rights to the main repository, but it does mean that we can accept your contributions, and you will get an author credit if we do. Active contributors might be asked to join the core team, and given the ability to merge pull requests.

### Code Conventions

Our code style is in line with [Google Java Style Guide](https://google.github.io/styleguide/javaguide.html).

We provide template files [intellij-java-google-style.xml](https://github.com/ctripcorp/apollo/blob/master/apollo-buildtools/style/intellij-java-google-style.xml) for IntelliJ IDEA and [eclipse-java-google-style.xml](https://github.com/ctripcorp/apollo/blob/master/apollo-buildtools/style/eclipse-java-google-style.xml) for Eclipse. If you use other IDEs, then you may config manually by referencing the template files.

\* Make sure all new .java files have a simple Javadoc class comment with at least an `@author` tag identifying you, and preferably at least a paragraph on what the class is for.

\* Add yourself as an @author to the .java files that you modify substantially (more than cosmetic changes).

\* Add some Javadocs and, if you change the namespace, some XSD doc elements.

\* A few unit tests should be added for a new feature or an important bug fix.

\* If no-one else is using your branch, please rebase it against the current master (or other target branch in the main project).

\* Normally, we would squash commits for one feature into one commit. There are 2 ways to do this:

1. To rebase and squash based on the remote branch

\* `git rebase -i <remote>/master`

\* merge commits via `fixup`, etc

2. Create a new branch and merge these commits into one

\* `git checkout -b <some-branch-name> <remote>/master`

\* `git merge --squash <current-feature-branch>`

\* When writing a commit message please follow these conventions: if you are fixing an existing issue, please add Fixes #XXX at the end of the commit message (where XXX is the issue number).