# Contribution Guide

Cobar is a community driven open source project and we welcome any contributor.

This document outlines some conventions about development workflow, commit message formatting, contact points and other resources to make it easier to get your contribution accepted.

## Pre submit pull request/issue flight checks

Before you move on, please make sure what your issue and/or pull request is, a simple bug fix or an architecture change.

In order to save reviewers' time, each issue should be filed with template and should be sanity-checkable in under 5 minutes.

### Is this a simple bug fix?

Bug fixes usually come with tests. With the help of continuous integration test, patches can be easy to review. Please update the unit tests so that they catch the bug!

### Is this an architecture improvement?

Some examples of "Architecture" improvements:

- Converting structs to interfaces.

- Improving test coverage.

- Decoupling logic or creation of new utilities.

- Making code more resilient (sleeps, backoffs, reducing flakiness, etc).

If you are improving the quality of code, then justify/state exactly what you are 'cleaning up' in your Pull Request so as to save reviewers' time.

If you're making code more resilient, test it locally to demonstrate how exactly your patch changes

things.

## Workflow

### Step 1: Fork in the cloud

1. Visit https://github.com/alibaba/cobar

2. Click `Fork` button (top right) to establish a cloud-based fork.

### Step 2: Clone fork to local storage and develop

### Step 3: Create a pull request

1. Visit your fork at https://github.com/$user/cobar (replace `$user` obviously).

2. Click the `Compare & pull request` button next to your branch.

### Step 4: get a code review

Once your pull request has been opened, it will be assigned to reviewers,

Those reviewers will do a thorough code review, looking for

correctness, bugs, opportunities for improvement, documentation and comments,

and style.

Commit changes made in response to review comments to the same branch on your

fork.

Very small PRs are easy to review. Very large PRs are very difficult to

review.

## Code style

The IntelliJ IDEA default java code style is Recommended?in every project has `.editorconfig` file, for more information please visit http://editorconfig.org/

## Commit message style

Please follow this style to make Cobar easy to review, maintain and develop.

```

<subsystem>: <what changed>

<BLANK LINE>

<why this change was made>

<BLANK LINE>

<footer>(optional)

```

The first line is the subject and should be no longer than 70 characters, the

second line is always blank, and other lines should be wrapped at 80 characters.

This allows the message to be easier to read on GitHub as well as in various

git tools.

If the change affects more than one subsystem, you can use comma to separate them like `util/codec,util/types:`.

If the change affects many subsystems, you can use ```\*``` instead, like ```\*:```.

For the why part, if no specific reason for the change,

you can use one of some generic reasons like "Improve documentation.",

"Improve performance.", "Improve robustness.", "Improve test coverage."