# Contributing to Milvus

First of all, thanks for taking the time to contribute to Milvus! It's people like you that help Milvus come to fruition. :tada:

The following are a set of guidelines for contributing to Milvus. Following these guidelines helps contributing to this project easy and transparent. These are mostly guideline, not rules. Use your best judgment, and feel free to propose changes to this document in a pull request.

As for everything else in the project, the contributions to Milvus are governed by our [Code of Conduct](CODE\_OF\_CONDUCT.md).

## Contribution Checklist

Before you make any contributions, make sure you follow this list.

- Read [Contributing to Milvus](CONTRIBUTING.md).

- Check if the changes are consistent with the [coding style](CONTRIBUTING.md#coding-style), and format your code accordingly.

- Run [unit tests](CONTRIBUTING.md#run-unit-test-with-code-coverage) and check your code coverage rate.

## What contributions can I make?

Contributions to Milvus fall into the following categories.

1. To report a bug or a problem with documentation, please file an [issue](https://github.com/milvus-io/milvus/issues/new/choose) providing the details of the problem. If you believe the issue needs priority attention, please comment on the issue to notify the team.

2. To propose a new feature, please file a new feature request [issue](https://github.com/milvus-io/milvus/issues/new/choose). Describe the intended feature and discuss the design and implementation with the team and community. Once the team agrees that the plan looks good, go ahead and implement it, following the [Contributing code](CONTRIBUTING.md#contributing-code).

3. To implement a feature or bug-fix for an existing outstanding issue, follow the [Contributing code](CONTRIBUTING.md#contributing-code). If you need more context on a particular issue, comment on the issue to let people know.

## How can I contribute?

### Contributing code

If you have improvements to Milvus, send us your pull requests! For those just getting started, see [GitHub workflow](#github-workflow). Make sure to refer to the related issue in your pull request's comment and update [CHANGELOG.md](CHANGELOG.md).

The Milvus team members will review your pull requests, and once it is accepted, the status of the projects to which it is associated will be changed to \*\*Reviewer approved\*\*. This means we are working on submitting your pull request to the internal repository. After the change has been submitted internally, your pull request will be merged automatically on GitHub.

### GitHub workflow

Please create a new branch from an up-to-date master on your fork.

1. Fork the repository on GitHub.

2. Clone your fork to your local machine with `git clone git@github.com:<yourname>/milvus-io/milvus.git`.

3. Create a branch with `git checkout -b my-topic-branch`.

4. Make your changes, commit, then push to to GitHub with `git push --set-upstream origin my-topic-branch`. You must record your changes in [CHANGELOG.md](CHANGELOG.md) with issue numbers and descriptions.

5. Visit GitHub and make your pull request.

If you have an existing local repository, please update it before you start, to minimize the chance of merge conflicts.

```shell

git remote add upstream git@github.com:milvus-io/milvus.git

git checkout master

git pull upstream master

git checkout -b my-topic-branch

```

### General guidelines

Before sending your pull requests for review, make sure your changes are consistent with the guidelines and follow the Milvus coding style.

- Include unit tests when you contribute new features, as they help to prove that your code works correctly, and also guard against future breaking changes to lower the maintenance cost.

- Bug fixes also require unit tests, because the presence of bugs usually indicates insufficient test coverage.

- Keep API compatibility in mind when you change code in Milvus. Reviewers of your pull request will comment on any API compatibility issues.

- When you contribute a new feature to Milvus, the maintenance burden is (by default) transferred to the Milvus team. This means that the benefit of the contribution must be compared against the cost of maintaining the feature.

### Developer Certificate of Origin (DCO)

All contributions to this project must be accompanied by acknowledgment of, and agreement to, the [Developer Certificate of Origin](https://developercertificate.org/). Acknowledgment of and agreement to the Developer Certificate of Origin \_must\_ be included in the comment section of each contribution and \_must\_ take the form of `Signed-off-by: {{Full Name}} <{{email address}}>` (without the `{}`). Contributions without this acknowledgment will be required to add it before being accepted. If contributors are unable or unwilling to agree to the Developer Certificate of Origin, their contribution will not be included.

Contributors sign-off that they adhere to DCO by adding the following Signed-off-by line to commit messages:

```text

This is my commit message

Signed-off-by: Random J Developer <random@developer.example.org>

```

Git also has a `-s` command line option to append this automatically to your commit message:

```shell

$ git commit -s -m 'This is my commit message'

```

## Coding Style

The coding style used in Milvus generally follow [Google C++ Style Guide](https://google.github.io/styleguide/cppguide.html).

And we made the following changes based on the guide:

- 4 spaces for indentation

- Adopt .cpp file extension instead of .cc extension

- 120-character line length

- Camel-Cased file names

### Format code

Install clang-format

```shell

$ sudo apt-get install clang-format

$ rm cmake\_build/CMakeCache.txt

```

Check code style

```shell

$ ./build.sh -l

```

To format the code

```shell

$ cd cmake\_build

$ make clang-format

```

## Run unit test with code coverage

Before submitting your PR, make sure you have run unit test, and your code coverage rate is >= 90%.

Install lcov

```shell

$ sudo apt-get install lcov

```

Run unit test and generate code for code coverage check

```shell

$ ./build.sh -u -c

```

Run MySQL docker

```shell

docker pull mysql:latest

docker run -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=123456 -d mysql:latest

```

Run code coverage

```shell

$ ./coverage.sh -u root -p 123456 -t 127.0.0.1

```

Or start your own MySQL server, and then run code coverage

```shell

$ ./coverage.sh -u ${MYSQL\_USERNAME} -p ${MYSQL\_PASSWORD} -t ${MYSQL\_SERVER\_IP}

```