# Contributing to Nacos

Welcome to Nacos! This document is a guideline about how to contribute to Nacos.

If you find something incorrect or missing, please leave comments / suggestions.

## Before you get started

### Code of Conduct

Please make sure to read and observe our [Code of Conduct](./CODE\_OF\_CONDUCT.md).

## Contributing

Nacos welcome new participants of any role, including user, contributor, committer and PMC.

![](http://acm-public.oss-cn-hangzhou.aliyuncs.com/contributor\_definition.png)

We encourage new comers actively join in Nacos projects and involving from user role to committer role, and even PMC role. In order to accomplish this, new comers needs to actively contribute in Nacos project. The following paragraph introduce how to contribute in Nacos way.

#### Open / pickup an issue for preparation

If you find a typo in document, find a bug in code, or want new features, or want to give suggestions, you can [open an issue on GitHub](https://github.com/alibaba/Nacos/issues/new) to report it.

If you just want to contribute directly you can choose the issue below.

- [Contribution Welcome](https://github.com/alibaba/nacos/labels/contribution%20welcome): Heavily needed issue, but currently short of hand.

- [good first issue](https://github.com/alibaba/nacos/labels/good%20first%20issue): Good for newcomers, new comer can pickup one for warm-up.

We strongly value documentation and integration with other projects such as Spring Cloud, Kubernetes, Dubbo, etc. We are very glad to work on any issue for these aspects.

Please note that any PR must be associated with a valid issue. Otherwise the PR will be rejected.

#### Begin your contribute

Now if you want to contribute, please create a new pull request.

We use the `develop` branch as the development branch, which indicates that this is a unstable branch.

Further more, our branching model complies to [https://nvie.com/posts/a-successful-git-branching-model/](https://nvie.com/posts/a-successful-git-branching-model/). We strongly suggest new comers walk through the above article before creating PR.

Now, if you are ready to create PR, here is the workflow for contributors:

1. Fork to your own

2. Clone fork to local repository

3. Create a new branch and work on it

4. Keep your branch in sync

5. Commit your changes (make sure your commit message concise)

6. Push your commits to your forked repository

7. Create a pull request to \*\*develop\*\* branch.

When creating pull request:

1. Please follow [the pull request template](./.github/PULL\_REQUEST\_TEMPLATE.md).

2. Please create the request to \*\*develop\*\* branch.

3. Please make sure the PR has a corresponding issue.

4. If your PR contains large changes, e.g. component refactor or new components, please write detailed documents about its design and usage.

5. Note that a single PR should not be too large. If heavy changes are required, it's better to separate the changes to a few individual PRs.

6. After creating a PR, one or more reviewers will be assigned to the pull request.

7. Before merging a PR, squash any fix review feedback, typo, merged, and rebased sorts of commits. The final commit message should be clear and concise.

If your PR contains large changes, e.g. component refactor or new components, please write detailed documents about its design and usage.

### Code review guidance

Committers will rotate reviewing the code to make sure all the PR will be reviewed timely and by at least one committer before merge. If we aren't doing our job (sometimes we drop things). And as always, we welcome volunteers for code review.

Some principles:

- Readability - Important code should be well-documented. API should have Javadoc. Code style should be complied with the existing one.

- Elegance: New functions, classes or components should be well designed.

- Testability - 80% of the new code should be covered by unit test cases.

- Maintainability - Comply with our [PMD spec](style/codeStyle.md), and 3-month-frequency update should be maintained at least.

### Now how about try become a committer?

Generally speaking, contribute 8 non-trivial patches and get at least three different people to review them (you'll need three people to support you). Then ask someone to nominate you. You're demonstrating your:

- at least 8 PR and the associated issues to the project,

- ability to collaborate with the team,

- understanding of the projects' code base and coding style, and

- ability to write good code (last but certainly not least)

A current committer nominates you by slacking the team on the Nacos issue with label "nomination"

- your first and last name

- a link to your Git profile

- an explanation of why you should be a committer,

- Elaborate the top 3 PR and the associated issues the nominator has worked with you that can demonstrate your ability.

Two other committer need to second your nomination. If no one objects in 5 working days (China), you're a committer. If anyone objects or wants more information, the committers discuss and usually come to a consensus (within the 5 working days). If issues cannot be resolved, there's a vote among current committers.

![](http://acm-public.oss-cn-hangzhou.aliyuncs.com/nomination\_process.png)

In the worst case, this can drag out for two weeks. Keep contributing! Even in the rare cases where a nomination fails, the objection is usually something easy to address like "more patches" or "not enough people are familiar with this person's work."