# Contributing to JanusGraph

Thank you for your intention to contribute to the JanusGraph project. As an open-source community, we highly appreciate external contributions to our project.

To make the process smooth for the project \*committers\* (those who review and accept changes) and \*contributors\* (those who propose new changes via pull requests), there are a few rules to follow.

## Sign the CLA

Before you can contribute to JanusGraph, please sign the Contributor License

Agreement (CLA). This is not a copyright \*assignment\*, it simply gives the

JanusGraph project the permission and license to use and redistribute your

contributions as part of the project.

\* If you are an individual writing original source code and you're sure you own

the intellectual property, you will need to follow

[individual contributor instructions](https://github.com/swinslow/cla-tool-docs/blob/master/walkthroughs/1-Individual-Contributor.md).

\* If you work for a company or another organization that may have claim to

intellectual property you may produce, and the organization wants to allow you

to contribute your work, then an authorized representative of the organization

will need to follow [CLA manager instructions](https://github.com/swinslow/cla-tool-docs/blob/master/walkthroughs/4-CLA-Manager.md).

While the contributors should follow [corporate contributor instructions](https://github.com/swinslow/cla-tool-docs/blob/master/walkthroughs/3-Corporate-Contributor-first-for-company.md).

## Fork the repository on GitHub

Click on the [fork](https://github.com/JanusGraph/janusgraph/fork) button at the

upper-right of the repository to create a fork of the repository in your own

account.

## Clone your fork locally

To fork the repo locally, use one of the following methods:

\* if you have set up and are using [SSH keys](https://help.github.com/articles/generating-an-ssh-key/), run:

```bash

$ git clone git@github.com:$USER/janusgraph

```

\* otherwise, to use HTTPS transfer, run:

```bash

$ git clone https://github.com/$USER/janusgraph

```

Add the original repo to synchronize new changes:

```bash

$ git remote add upstream git@github.com:JanusGraph/janusgraph

```

Do not contribute to `master` from our own fork's `master` branch.

Also, do not push your working, in-progress, or PR branches to

`JanusGraph/janusgraph`; instead, use your own fork.

You can avoid pushing to upstream by accident by setting it as follows:

```bash

$ git remote set-url --push upstream do-not-push

```

## Configure your repo to match the CLA

Be sure to use the same name and email in your Git commits as in your signed CLA

to make sure that the automated verification process correctly matches it.

To configure them, use the following commands

```bash

$ cd janusgraph

$ git config user.name "My Name"

$ git config user.email "my-email@example.com"

```

> Note: if you forget to use the right email address the first time, and

> @janusgraph-bot adds the `[cla: no]` label to your PR, you can fix it in-place

> while reusing your existing PR as follows:

>

> \* set your name and email as per above

> \* run the following commands:

>

> ```bash

> $ git commit --amend -s --reset-author

> $ git push -f

> ```

## Create a new branch

First, you need to decide which release branch (e.g., `master`, `v0.2`) to

create the feature branch from. If you intend to add a new feature, then

`master` is the right branch. Bug fixes however should also be applied to

other releases, so you should create your feature branch from the release

branch with the lowest version number that is still active (e.g., `v0.2`).

When in doubt, ask on [janusgraph-dev](https://groups.google.com/forum/#!forum/janusgraph-dev).

Changes to all release branches will also be merged into `master`.

Do not develop on the release branch: feature branches are intended to be

light-weight and deleted after being merged to upstream, but you should not

delete your release branch, so don't use it for development.

Instead, update your release branch and create a new branch for development:

```bash

$ git checkout master

$ git pull --ff-only upstream master

$ git checkout -b my-new-feature

```

> NOTE: This listing assumes that you create the feature branch from `master`.

> Replace `master` by the name of the release branch (e.g., `v0.2`) if you want

> to create the branch from that release branch instead.

## Develop and test your changes

Make that feature or bugfix awesome!

## Configuration changes

If you add any configuration or change any configuration,

you have to update `docs/basics/janusgraph-cfg.md`.

To update config reference use following command:

```

mvn --quiet clean install -DskipTests=true -pl janusgraph-doc -am

```

Our release branches should always contain the correct configuration documentation,

that is enforced by our build pipeline.

## Commit changes and sign the Developer Certificate of Origin

The Linux Foundation requires that each contributor to its projects signs the

[Developer Certificate of Origin](https://developercertificate.org), also

available in this repo as [`DCO.txt`](DCO.txt). To sign it, just use the `commit

-s` command:

```bash

$ git add [...files...]

$ git commit -s

```

> Note: if you forget to sign your commit, you can amend it and fix the PR

> in-place as follows:

>

> ```bash

> $ git commit --amend -s

> $ git push -f

> ```

You can also create a convenient alias to automatically sign each commit in this

repository to avoid forgetting it:

```bash

$ git config alias.ci 'commit -s'

```

With this alias, from now on, you can just run:

```bash

$ git ci

```

to commit files, and it will automatically append the `-s` switch to ensure that

you've signed the contribution.

> Note: as it is an alias, you can still append flags to the command line, e.g.,

> `git ci -v` will get you a diff of your commit while writing your commit

> message.

> Note: If this is a non-code change, e.g. documentation, add `[doc only]` to the

> PR subject line. This is to save CPU time on Travis CI, which lets us get more

> build time for the other changes which actually change the code.

>

> The tests actually run twice for each PR:

>

> \* when the PR is submitted for review

> \* when the PR is merged to the base branch

>

> Having [doc only] in the commit skips the first one, but the merge commit also

> needs it, so having it in the title (first line of commit) helps it easily

> propagate to both places.

## Push your changes to your GitHub fork

```bash

$ git push

```

If this is the first time you are pushing this branch, depending on your

settings, `git` may tell you to re-run that command with additional flags and/or

parameters. If so, copy-paste and run that command.

After you do this once for this branch, you can use `git push` to add additional

changes.

## Open a Pull Request

Go to the [JanusGraph repository](https://github.com/JanusGraph/janusgraph) and

you should see that it will offer you a chance to compare your recently-pushed

branch to the current `master` of JanusGraph and submit a PR at the same time.

Review the [PR check list](.github/PULL\_REQUEST\_TEMPLATE.md) for criteria for acceptable contributions.

## Code review

If you have signed the CLA, you should pass `EasyCLA` status check.

If you see that CLA check isn't passed, there will be a comment from `EasyCLA` with the link

to sign a CLA. Make sure that your CLA and your name, email, and GitHub usernames all match,

for all of the commits in your PR.

If you are not sure what's wrong, reach out to

janusgraph-cla@googlegroups.com with further questions.

### Code Review

JanusGraph uses [Codacy](https://www.codacy.com) for static analysis pull requests to ensure code

quality. Codacy creates in-line comments for items which require attention, and can generate a large

number of potentially unwanted emails. Adding codacy-bot to your [blocked users](https://github.com/settings/blocked\_users) will stop the notifications.