# How to Contribute

First of all, thank you for your interest in ndb!

We'd love to accept your patches and contributions!

## Contributor License Agreement

Contributions to this project must be accompanied by a Contributor License

Agreement. You (or your employer) retain the copyright to your contribution,

this simply gives us permission to use and redistribute your contributions as

part of the project. Head over to <https://cla.developers.google.com/> to see

your current agreements on file or to sign a new one.

You generally only need to submit a CLA once, so if you've already submitted one

(even if it was for a different project), you probably don't need to do it

again.

## Getting setup

1. Clone this repository

```bash

git clone https://github.com/GoogleChromeLabs/ndb

cd ndb

```

2. Install dependencies

```bash

npm install

```

## Code reviews

All submissions, including submissions by project members, require review. We

use GitHub pull requests for this purpose. Consult

[GitHub Help](https://help.github.com/articles/about-pull-requests/) for more

information on using pull requests.

## Code Style

- Coding style is fully defined in [.eslintrc](https://github.com/GoogleChrome/puppeteer/blob/master/.eslintrc.js)

- Code should be annotated with [closure annotations](https://github.com/google/closure-compiler/wiki/Annotating-JavaScript-for-the-Closure-Compiler).

- Comments should be generally avoided. If the code would not be understood without comments, consider re-writing the code to make it self-explanatory.

To run code linter, use:

```bash

npm run lint

```

## Commit Messages

Commit messages should follow the Semantic Commit Messages format:

```

label(namespace): title

description

footer

```

1. \*label\* is one of the following:

- `fix` - ndb bug fixes.

- `feat` - ndb features.

- `docs` - changes to docs, e.g. `docs(api.md): ..` to change documentation.

- `test` - changes to ndb tests infrastructure.

- `style` - ndb code style: spaces/alignment/wrapping etc.

- `chore` - build-related work.

2. \*namespace\* is put in parenthesis after label and is \*\*optional\*\*.

3. \*title\* is a brief summary of changes.

4. \*description\* is \*\*optional\*\*, new-line separated from title and is in present tense.

5. \*footer\* is \*\*optional\*\*, new-line separated from \*description\* and contains "fixes" / "references" attribution to github issues.

Example:

```

fix(NddService): fix NddService.attach method

This patch fixes NddService.attach so that it works with Node 12.

Fixes #123, Fixes #234

```

## Adding New Dependencies

For all dependencies (both installation and development):

- \*\*Do not add\*\* a dependency if the desired functionality is easily implementable.

- If adding a dependency, it should be well-maintained and trustworthy.

A barrier for introducing new installation dependencies is especially high:

- \*\*Do not add\*\* installation dependency unless it's critical to project success.

## Writing Tests

- Every ndb service feature should be accompanied by a test.

- Tests should be \*hermetic\*. Tests should not depend on external services.

- Tests should work on all three platforms: Mac, Linux and Win.

ndb tests are located in [test/test.js](https://github.com/GoogleChromeLabs/ndb/blob/master/test/test.js)

and are written with a [mocha](https://mochajs.org/) framework.

- To run all tests:

```bash

npm run unit

```

- To run a specific test, substitute the `it` with `fit` (mnemonic rule: '\*focus it\*'):

```js

...

// Using "fit" to run specific test

fit('should work', async function({service}) {

const response = await service.method();

expect(response).toBe(true);

})

```

- To disable a specific test, substitute the `it` with `xit` (mnemonic rule: '\*cross it\*'):

```js

...

// Using "xit" to skip specific test

xit('should work', async function({service}) {

const response = await service.method();

expect(response).toBe(true);

})

```

## Developing ndb hints

- Environment variable NDB\_DEBUG\_FRONTEND=1 forces ndb to fetch

frontend from front\_end folder and chrome-devtools-frontend

package.

```bash

NDB\_DEBUG\_FRONTEND=1 ndb .

```

- To debug ndb by itself or any ndb service you can use ndb.

```bash

NDB\_DEBUG\_FRONTEND=1 ndb ndb index.js

```

- To debug running Chrome DevTools frontend you can open DevTools,

use Ctrl+Shift+I on Linux or View > Developer > Developer Tools menu

item on Mac OS.

## [For Project Maintainers] Releasing to NPM

Releasing to NPM consists of 3 phases:

1. Source Code: mark a release.

1. Bump `package.json` version following the SEMVER rules and send a PR titled `'chore: mark version vXXX.YYY.ZZZ'`.

2. Make sure the PR passes \*\*all checks\*\*.

3. Merge the PR.

4. Once merged, publish release notes using the "create new tag" option.

- \*\*NOTE\*\*: tag names are prefixed with `'v'`, e.g. for version `1.4.0` tag is `v1.4.0`.

2. Publish to NPM.

1. On your local machine, pull from [upstream](https://github.com/GoogleChromeLabs/ndb) and make sure the last commit is the one just merged.

2. Run `git status` and make sure there are no untracked files.

- \*\*WHY\*\*: this is to avoid bundling unnecessary files to NPM package

3. Run [`pkgfiles`](https://www.npmjs.com/package/pkgfiles) to make sure you don't publish anything unnecessary.

4. Run `npm publish`.

3. Source Code: mark post-release.

1. Bump `package.json` version to `-post` version and send a PR titled `'chore: bump version to vXXX.YYY.ZZZ-post'`

- \*\*NOTE\*\*: no other commits should be landed in-between release commit and bump commit.