# Contributing to Carbon

## Issue Contributions

Please report any security issue using [Tidelift security contact](https://tidelift.com/security).

Tidelift will coordinate the fix and disclosure.

Please don't disclose security bugs publicly until they have been handled by us.

For any other bug or issue, please click this link and follow the template:

[Create new issue](https://github.com/briannesbitt/Carbon/issues/new)

You may think this template does not apply to your case but please think again. A long description will never be as

clear as a code chunk with the output you expect from it (for either bug report or new features).

## Code Contributions

### Where to begin

We use the label \*\*good first issue\*\* to tag issues that could be a good fit for new contributors, see if there are such issues now following this link:

https://github.com/briannesbitt/Carbon/issues?q=is%3Aissue+is%3Aopen+label%3A%22good+first+issue%22

Else, check the roadmap to see what we plan to do in next releases:

https://github.com/briannesbitt/Carbon/issues/1681

### Develop locally, then submit changes

Fork the [GitHub project](https://github.com/briannesbitt/Carbon) and download it locally:

```shell

git clone https://github.com/<username>/Carbon.git

cd Carbon

git remote add upstream https://github.com/briannesbitt/Carbon.git

```

Replace `<username>` with your GitHub username.

Then, you can work on the master or create a specific branch for your development:

```shell

git checkout -b my-feature-branch -t origin/master

```

You can now edit the "Carbon" directory contents.

Before committing, please set your name and your e-mail (use the same e-mail address as in your GitHub account):

```shell

git config --global user.name "Your Name"

git config --global user.email "your.email.address@example.com"

```

The ```--global``` argument will apply this setting for all your git repositories, remove it to set only your Carbon

fork with those settings.

Now you can commit your modifications as you usually do with git:

```shell

git add --all

git commit -m "The commit message log"

```

If your patch fixes an open issue, please insert ```#``` immediately followed by the issue number:

```shell

git commit -m "#21 Fix this or that"

```

Use git rebase (not git merge) to sync your work from time to time:

```shell

git fetch origin

git rebase origin/master

```

Please add some tests for bug fixes and features (so it will ensure next developments will not break your code),

then check all is right with phpunit:

Install PHP if you haven't yet, then install composer:

https://getcomposer.org/download/

Update dependencies:

```

./composer.phar update

```

Or if you installed composer globally:

```

composer update

```

Then call phpunit:

```

./vendor/bin/phpunit

```

Make sure all tests succeed before submitting your pull-request, else we will not be able to merge it.

Push your work on your remote GitHub fork with:

```

git push origin my-feature-branch

```

Go to https://github.com/yourusername/Carbon and select your feature branch. Click the 'Pull Request' button and fill

out the form.

We will review it within a few days. And we thank you in advance for your help.

## Versioning

### Note about Semantic Versioning and breaking changes

As a developer, you must understand every change is a breaking change. What is a bug for someone

is expected in someone else's workflow. The consequence of a change strongly depends on the usage.

[Semantic Versioning](https://semver.org/) relies to public API. In PHP, the public API of a class is its public

methods. However, if you extend a class, you can access protected methods, then if you use reflexion, you can

access private methods. So anything can become a public API if you force it to be. That doesn't mean we should handle

any possible usage, else we would have to publish a major release for each change and it would no longer make sense.

So before any complain about a breaking change, be warned, we do not guarantee a strict Semantic Versioning as you

may expect, we're following a pragmatic interpretation of Semantic Versioning that allows the software to evolve in a

reliable way with reasonable maintenance effort.

Concretely, we consider a change as breaking if it makes fail one of our unit test. We will do our best to avoid

incompatibilities with libraries that extends Carbon classes (such as Laravel that is continuously tested thanks to

Travis CI, [see the compatibility matrix](https://github.com/kylekatarnls/carbon-laravel/tree/master#carbon-1-dev-version-1next)).

If you're the owner of a library that strongly depends on Carbon, we recommend you to run unit tests daily requiring

`"nesbot/carbon": "dev-master"` (for `^2`) or `"nesbot/carbon": "dev-version-1.next"` (for `^1`), this way you can

detect incompatibilities earlier and report it to us before we tag a release. We'll pay attention and try to fix it to

make update to next minor releases as soft as possible.

We reserve the right to publish emergency patches within 24 hours after a release if a tag that does not respect

this pattern would have been released despite our vigilance. In this very rare and particular case, we would mark the

tag as broken on GitHub and backward compatibility would be based on previous stable tag.

Last, you must understand that Carbon extends PHP natives classes, that means Carbon can be impacted by any change

that occurs in the date/time API of PHP. We watch new PHP versions and handle those changes as quickly as possible

when detected, but as PHP does not follow the semantic versioning pattern, it basically means any releases (including

patches) can have unexpected consequences on Carbon methods results.

### Long term support

To benefit the better support, require Carbon using major version range (`^1` or `^2`). By requiring `1.26.\*`,

`~1.26.0` or limited range such as `>=1.20 <1.33`, you fall to low priority support (only security and critical issues

will be fixed), our prior support goes to next minor releases of each major version. It applies to bug fixes and

low-cost features. Other new features will only be added in the last stable release. At the opposite, we recommend you

to restrain to a major number, as there is no compatibility guarantee from a major version to the next. It means

requiring `>=2`, as it allows any newer version, will probably leads to errors on releasing our next major version.

Open milestones can be patched if a minor bug is detected while if you're on a closed milestone, we'll more likely

ask you to update first to an open one. See currently open milestones:

https://github.com/briannesbitt/Carbon/milestones