# Contributing Guidelines

We are always looking to promote good contributors to be maintainers and provide them a front-row seat to serverless innovation.

If you would like to be a maintainer for the [Serverless Framework](https://github.com/serverless/serverless) or any of our plugins, please get started with making code contributions and engaging with open issues/PRs. Also, please reach out to any of [Serverless organization](https://github.com/serverless) members to express your interest.

We'd love to collaborate closely with amazing developers as we drive the development of this open technology into the future.

Welcome, and thanks in advance for your help!

# How to contribute to Serverless

## Setup

Once you've cloned forked repository, all is needed is to run `npm install` at its root folder

## Anatomy of a Framework

Check [docs/anatomy.md](./docs/anatomy.md)

## When you propose a new feature or bug fix

Please make sure there is an open issue discussing your contribution before jumping into a Pull Request!

There are just few situations (listed below) in which it is fine to submit PR without a corresponding issue:

- Documentation update

- Obvious bug fix

- Maintanance improvement

In all other cases please check if there's an open an issue discussing given proposal, if there is not, create an issue respecting all its template remarks.

In non trivial cases please propose and let us review an implementation spec (in corresponding issue) before jumping into implementation.

Do not submit draft PR's. Submit only finalized work which is ready for merge. If you have any doubts related to implementation work please discuss in corresponding issue.

Once PR was reviewed and some changes were suggested, please ensure to re-request review after all new changes were pushed. It's the best, noiseless way to inform maintainers that your work is ready to be checked again

## When you want to work on an existing issue

\*\*Note:\*\* Please write a quick comment in the corresponding issue and ask if the feature is still relevant and that you want to jump into the implementation.

Check out our [help wanted](https://github.com/serverless/serverless/labels/help%20wanted) or [good first issue](https://github.com/serverless/serverless/labels/good%20first%20issue) labels to find issues we want to move forward on with your help.

We will do our best to respond/review/merge your PR according to priority. We hope that you stay engaged with us during this period to insure QA. Please note that the PR will be closed if there hasn't been any activity for a long time (~ 30 days) to keep us focused and keep the repo clean.

## Reviewing Pull Requests

Another really useful way to contribute to Serverless is to review other peoples Pull Requests. Having feedback from multiple people is really helpful and reduces the overall time to make a final decision about the Pull Request.

## Writing / improving documentation

Our documentation lives on GitHub in the [docs](docs) directory. Do you see a typo or other ways to improve it? Feel free to edit it and submit a Pull Request!

## Providing support

The easiest thing you can do to help us move forward and make an impact on our progress is to simply provide support to other people having difficulties with their Serverless projects.

You can do that by replying to [issues on Github](https://github.com/serverless/serverless/issues), chatting with other community members in [our Chat](http://chat.serverless.com) or helping with questions in [our Forum](http://forum.serverless.com).

---

# Code Style

We aim for clean, consistent code style. We're using [Prettier](https://prettier.io/) to confirm on one code formatting style and [ESlint](https://eslint.org/) helps us to stay away from obvious issues that can be picked via static analisys.

Ideally you should have Prettier and ESlint integrated into your code editor, that will help you not think about specific rules and be sure you submit the code that follows guidelines.

## Verifying prettier formatting

```

npm run prettier-check

```

## Verifying linting style

```

npm run lint

```

## Other guideliness

- Minimize [lodash](https://lodash.com/) usage - resort to it, only if given part of logic cannot be expressed easily with native language constructs

- Do not rely on custom [Bluebird](http://bluebirdjs.com) functions methods (aside of `Bluebird.try`) - we're looking forward to drop this dependency with next major.

### Configuring deprecations

Ideally all breaking changes should be first (before being shipped with next major) communicated with deprecation logs.

Dprecation log can be configured with following steps:

1. At logic point where deprecate feature is being used, write a deprecation log with `serverless.\_logDeprecation` util. It accepts two arguments:

- `code` (e.g. `DEPRECATED\_FEATURE\_NAME`). Created to identify log programmatically, also used to construct link on documentation page

- `mesage` Deprecation message to be displayed to user

2. Document introduced deprecation at `docs/depreactions.md` (new deprecation should be listed as first, follow format of other documented deprecations)

# Testing

See [tests/README](tests/README.md)

# Our Code of Conduct

Finally, to make sure you have a pleasant experience while being in our welcoming community, please read our [code of conduct](CODE\_OF\_CONDUCT.md). It outlines our core values and believes and will make working together a happier experience.

Thanks again for being a contributor to the Serverless Community :tada:!

Cheers,

The :zap: [Serverless](http://www.serverless.com) Team