# Backup contribution guide

Thank you for your interest in contributing to Backup! Backup is community

driven project, which means it keeps existing and improves because \*you\* and

people like you decide to spend time improving it in the various ways described

below.

The goal of this document is to create a contribution process that:

\* Encourages new contributions.

\* Encourages contributors to remain involved.

\* Avoids unnecessary processes and bureaucracy whenever possible.

\* Creates a transparent decision making process which makes it clear how

contributors can be involved in decision making.

This document is partially based on the Node.js Community Contributing Guide 1.0

## Vocabulary

\* A \*\*Contributor\*\* is any individual creating or commenting on an issue, pull

request or message on the project gitter channel

\* A \*\*Committer\*\* is a subset of contributors who have been given write access to the repository.

\* A \*\*TC (Technical Committee)\*\* is a group of committers representing the

required technical expertise to resolve rare disputes and/or having the

credentials to perform administrative actions on the project ressources (like

releasing on rubygems.org). The list of current TC members can be found in the

TC.md file in the master branch of the project repository.

## Code of Conduct

Please be courteous and respectful. Remember: \*Matz is nice and so we are nice\*

## Bugs

To report a bug, use the [GitHub issue tracker](https://github.com/backup/backup/issues). Please check the Open issues, and create a new issue if you do not see a report that matches the bug that you have found.

Backup is used with many different systems and configurations, so the first step for us to solve a problem is knowing how to reproduce it. You can help us solve your issue by including the versions of Ruby,

Backup and the operating system on the computer that you used.

To submit a bug fix, please refere to the [Pull Requests section](#pull-requests)

## Asking for (or Giving) Help

If you would like to talk about either using Backup or writing code for it,

please leave a message in our [Gitter room](https://gitter.im/backup/backup).

Committers & TC members are not online at all times, so please be patient.

## Triage Issues [![Open Source Helpers](https://www.codetriage.com/backup/backup/badges/users.svg)](https://www.codetriage.com/backup/backup)

Triaging issues is a great way to help the project. This can include reproducing bug reports or asking for additional information, such as version numbers or reproduction instructions. If you would like to start triaging issues, one easy way to get started is to [subscribe to backup on CodeTriage](https://www.codetriage.com/backup/backup).

## Pull Requests

Any change to resources in this repository must be through [pull

requests](https://help.github.com/articles/about-pull-requests/). This applies

to all changes to documentation, code, binary files, etc. Even long term

committers and TC members must use pull requests.

If you would like to discuss some details before you start working on a feature

or bug fix, [open an issue](https://github.com/backup/backup/issues).

No pull request can be merged without being reviewed, except for critical

security fixes or administrative commits by TC members (e.g. version bumps for

release cutting)

For non-trivial contributions, pull requests should sit for at least 36 hours to

ensure that contributors in other timezones have time to review. Consideration

should also be given to weekends and other holiday periods to ensure active

committers all have reasonable time to become involved in the discussion and

review process if they wish.

The default for each contribution is that it is accepted once no committer has

an objection. During review committers may also request that a specific

contributor who is most versed in a particular area gives a "LGTM" before the PR

can be merged. There is no additional "sign off" process for contributions to

land. Once all issues brought by committers are addressed it can be landed by

any committer.

In the case of an objection being raised in a pull request by another committer,

all involved committers should seek to arrive at a consensus by way of

addressing concerns being expressed by discussion, compromise on the proposed

change, or withdrawal of the proposed change.

If a contribution is controversial and committers cannot agree about how to get

it to land or if it should land then it should be escalated to the TC. TC

members should regularly discuss pending contributions in order to find a

resolution. It is expected that only a small minority of issues be brought to

the TC for resolution and that discussion and compromise among committers be the

default resolution mechanism.

### PR standards

To help the committers review your code and speed up the merge:

\* Use the latest version of the `master` branch as the base for your topic branch

\* Be sure to use the latest version of Ruby 2 when you write and test your code

\* Write tests for your changes

\* In the comment box for your pull request, specify the operating system(s) and Ruby version that you have tested your code on

\* Write [clear commit messages](http://chris.beams.io/posts/git-commit/):

the first line should be 50 characters or less, and be a clear summary of the commit, e.g. "Fix Nokogiri compile issue on macOS Sierra, GH #305".

## Becoming a Committer

All contributors who land a non-trivial contribution should be on-boarded in a

timely manner, and added as a committer, and be given write access to the

repository.

Committers are expected to follow this policy and continue to send pull

requests, go through proper review, and have other committers merge their pull

requests.

## TC Process

The TC uses a "consensus seeking" process for issues that are escalated to the TC.

The group tries to find a resolution that has no open objections among TC members.

If a consensus cannot be reached that has no objections then a majority wins vote

is called. It is also expected that the majority of decisions made by the TC are via

a consensus seeking process and that voting is only used as a last-resort.

Resolution may involve returning the issue to committers with suggestions on how to

move forward towards a consensus. It is not expected that a meeting of the TC

will resolve all issues on its agenda during that meeting and may prefer to continue

the discussion happening among the committers.

### Adding new TC members

Members can be added to the TC at any time. Any committer can nominate another

committer to the TC and the TC uses its standard consensus seeking process to

evaluate whether or not to add this new member. Members who do not participate

consistently at the level of a majority of the other members are expected to

resign.

TC members who wants to receive administrative access on the project's resources

(like rubygems.org push rights) are required to have the most secure possible

configuration on their respective accounts (e.g. unique, strong password with

2FA enabled when available)

### Release cutting

Any committer can suggest the release of a new version of the software and the

TC uses its standard consensus seeking process to evaluate whether or not to

perform it. The releases are performed by a TC member with the required

administrative access.