# Guidelines for Contributing Code

New Relic welcomes code contributions by the Ruby community, and

have taken effort to make this process easy for both contributors and our

development team.

When contributing, keep in mind that the agent runs in a wide variety of Ruby

language implementations (e.g. 2.x.x, jruby, etc.) as well as a wide variety of

application environments (e.g. Rails, Sinatra, roll-your-own, etc.) See [Ruby agent requirements and supported frameworks]

(https://docs.newrelic.com/docs/agents/ruby-agent/getting-started/ruby-agent-requirements-supported-frameworks)

for the current full list.

Because of this, we need to be more defensive in our coding practices than most

projects. Syntax must be compatible with all supported Ruby implementations and

we can't assume the presence of any specific libraries, including `ActiveSupport`,

`ActiveRecord`, etc.

## Process

### Version Support

When contributing, keep in mind that New Relic customers (that's you!) are

running many different versions of Ruby, some of them pretty old. Changes that

depend on the newest version of Ruby will probably be rejected, especially if

they replace something backwards compatible.

Be aware that the instrumentation needs to work with a wide range of versions of

the instrumented modules, and that code that looks nonsensical or

overcomplicated may be that way for compatibility-related reasons. Read all the

comments and check the related tests before deciding whether existing code is

incorrect.

If you are planning on contributing a new feature or an otherwise complex

contribution, we kindly ask you to start a conversation with the maintainer team

by opening up an Github issue first.

### General Guidelines

The Ruby agent avoids requiring any dependencies in the main agent code base.

Instrumentations and features that would otherwise require a dependency (such as

Infinite Tracing, which require gRPC and protobuf) are built as separate gems.

If you have a feature or instrumentation request that would require a

dependency, please open an Issue to discuss with the maintainers before

proceeding.

Your code will be evaluated for completeness and accuracy in

implementation and must be accompanied with appropriate unit tests. New

additions that do not break existing tests are the easiest and quickest to be

accepted and merged. New features and improvements that break existing

functionality are slower to be accepted and merged as they require agreement

with maintainers across a majority of the languages New Relic supports. Any

such breaking changes will require a major version bump whereas non-breaking

additions only lead to minor version bumps.

Please be aware that the maintainers of New Relic’s agents aim to have as much

commonality of functionality across all language agents as makes sense, so we are

always working to reconcile language-specific changes against the cross-language

community set of agreements.

### Feature Requests

Feature requests should be submitted in the [Issue tracker](../../issues), with

a description of the expected behavior & use case. Before submitting an Issue,

please search for similar ones in the [closed

issues](../../issues?q=is%3Aissue+is%3Aclosed).

### Pull Requests

We can only accept PRs for version v6.12.0 or greater due to open source

licensing restrictions.

### Code of Conduct

Before contributing please read the [code of conduct](https://github.com/newrelic/.github/blob/master/CODE\_OF\_CONDUCT.md)

Note that our [code of conduct](https://github.com/newrelic/.github/blob/master/CODE\_OF\_CONDUCT.md) applies to all platforms

and venues related to this project; please follow it in all your interactions

with the project and its participants.

## Branches

The head of `main` will generally have New Relic's latest release. However,

New Relic reserves the ability to push an edge to the `main`. If you download a

release from this repo, use the appropriate tag. New Relic usually pushes beta

versions of a release to a working branch before tagging them for General

Availability.

## Testing

The agent includes a suite of unit and functional tests which should be used to

verify your changes don't break existing functionality.

Unit tests are stored in the `test/new\_relic` directory.

Functional tests are stored in the `test/multiverse` directory.

### Running Tests

Running the test suite is simple. Just invoke:

bundle

bundle exec rake

This will run the unit tests in standalone mode, bootstrapping a basic Rails

3.2 environment for the agent to instrument, then executing the test suite.

These tests are setup to run automatically in

[Travis CI](https://travis-ci.org/newrelic/rpm) under several Ruby implementations.

When you've pushed your changes to GitHub, you can confirm that the Travis

build passes for your fork.

Additionally, our own CI jobs runs these tests under multiple versions of Rails

to verify compatibility.

### Writing Tests

For most contributions it is strongly recommended to add additional tests which

exercise your changes.

This helps us efficiently incorporate your changes into our mainline codebase

and provides a safeguard that your change won't be broken by future development.

There are some rare cases where code changes do not result in changed

functionality (e.g. a performance optimization) and new tests are not required.

In general, including tests with your pull request dramatically increases the

chances it will be accepted.

### Functional Testing

For cases where the unit test environment is not sufficient for testing a

change (e.g. instrumentation for a non-Rails framework, not available in the

unit test environment), we have a functional testing suite called multiverse.

These tests can be run by invoking:

bundle

bundle exec rake test:multiverse

More details are available in

[test/multiverse/README.md](https://github.com/newrelic/newrelic-ruby-agent/blob/main/test/multiverse/README.md).

## Contributor License Agreement

Keep in mind that when you submit your Pull Request, you'll need to sign the CLA

via the click-through using CLA-Assistant. If you'd like to execute our

corporate CLA, or if you have any questions, please drop us an email at

opensource@newrelic.com.

For more information about CLAs, please check out Alex Russell’s excellent post,

[“Why Do I Need to Sign This?”](https://infrequently.org/2008/06/why-do-i-need-to-sign-this/).

## Slack

We host a public Slack with a dedicated channel for contributors and maintainers

of open source projects hosted by New Relic. If you are contributing to this

project, you're welcome to request access to the #oss-contributors channel in

the newrelicusers.slack.com workspace. To request access, see

https://newrelicusers-signup.herokuapp.com/.

## Explorer's Hub

New Relic hosts and moderates an online forum where customers can interact with

New Relic employees as well as other customers to get help and share best

practices. Like all official New Relic open source projects, there's a related

Community topic in the New Relic Explorers Hub. You can find this project's

topic/threads here:

[Explorer's Hub](https://discuss.newrelic.com/c/support-products-agents/ruby-agent)

## And Finally...

If you have any feedback on how we can make contributing easier, please get in

touch at [support.newrelic.com](http://support.newrelic.com) and let us know!