# Getting help

If you have a bug or a feature, we welcome new issues. We also answer questions on StackOverflow (use the `neo4j.rb` tag) and are generally around on [Gitter](https://gitter.im/neo4jrb/neo4j)

# Code standards

Pull request with high test coverage and good [code climate](https://codeclimate.com/github/neo4jrb/neo4j) values will be accepted faster.

# Rubocop

We use [rubocop](https://github.com/bbatsov/rubocop) for sanity checks to make sure our code doesn't get too ugly. Because of this, a pull request can fail on Travis CI when the specs all run fine. We recommend installing [overcommit](https://github.com/causes/overcommit) so that rubocop is run in pre-commit hooks so that this causes minimal frustration. We already have an overcommit configuration, so you simply need to run `overcommit --install` to set up the git hooks (note: if using our Vagrantfile, this is handled for you). A `Guardfile` is also provided so that you can use rspec and rubocop continuously while working.

# Travis CI

We use Travis CI to test across multiple verisons of ruby and versions of Neo4j, so please watch for failures there.

# Guard

A `Guardfile` has been provided for automatic running of `rspec` and `rubocop` to make development easier. To run, simply execute `bundle exec guard`

# Docker

A `Dockerfile` has been provided to make it easier to run commands like `rspec`, `rubocop`, and `guard` in Docker containers. There is also a [configuration file](https://gist.github.com/cheerfulstoic/c2bb5c4a1eb5e8c19c570d0da151c7a5) for tmuxinator (a Ruby gem which allows for quick setup of tmux sessions) which is useful for uses of tmux but is also useful as a reference on how to use Docker.

# Vagrant

A `Vagrantfile` has been provided to make setting up a dev environment easier. To use it, [install vagrant](https://www.vagrantup.com) and then run `vagrant up` and `vagrant ssh` from the root folder of this repository and you're good to go! Inside the VM, run `rake neo4j:start` and then `rspec` (after the server has started) to make sure everything's been properly set up. Vagrant will install & configure a VM running ubuntu. If one doesn't exist already, it will also install and configure a neo4j development server in './db/neo4j/development' (this check is only performed the first time you `vagrant up`). As part of the installation procedure, Vagrant may need to automatically accept various TOS and EULA's (on your behalf) related to the software it is installing. View the `Vagrantfile` before running `vagrant up` if you'd like to learn more.

## Usage notes:

- If you're using vagrant with an IDE on the host machine, remember that the host machine doesn't necessarily have this repository's gems installed. For example, if you try to use your IDE to make a git commit, it might fail with a message ~"this repository has git hooks installed with overcommit, but the overcommit gem is not installed." This is because overcommit hasn't been installed on the host machine. To make your commit, simply `vagrant ssh` and `git commit` the old fashioned way.

### If you want to use the Vagrantfile but have already installed neo4j:

- If you already have neo4j installed in './db/neo4j/development', vagrant will leave your installation alone. This includes \_not configuring your neo4j installation for use with vagrant\_. If you wish to use the `Vagrantfile`, it is recommended that you delete your './db/neo4j/development' folder prior to running `vagrant up`. This will ensure that, while provisioning your VM, vagrant installs the latest version of neo4j community edition and configures is appropriately.

- If you already have neo4j installed but not in './db/neo4j/development', then vagrant will leave your installation alone AND install the latest version of neo4j into './db/neo4j/development'. This is because, duing the initial provisioning of your VM, vagrant looks for the existance of './db/neo4j/development' to determine if it should install neo4j. To stop vagrant from installing a copy of neo4j into './db/neo4j/development', simply add an empty `.keep` file to './db/neo4j/development/.keep' prior to running `vagrant up`.

### Mac users:

- By default, Timemachine will back up your vagrant VMs. To stop this, open Timemachine preferences and exclude your VMs folder. By default, vagrant uses VirtualBox to power the VMs and the VMs are saved into ~/"VirtualBox VMs"