# Contributing

Hi there!

We're thrilled that you'd like to contribute to this project.

Your help is essential for keeping it great.

Contributions to this project are [released](https://help.github.com/articles/github-terms-of-service/#6-contributions-under-repository-license) to the public under the [project's open source license](LICENSE).

This project adheres to the [Contributor Covenant Code of Conduct](http://contributor-covenant.org/).

By participating, you are expected to uphold this code.

The majority of contributions won't need to touch any Ruby code at all.

### Dependencies

Linguist uses the [`charlock\_holmes`](https://github.com/brianmario/charlock\_holmes) character encoding detection library which in turn uses [ICU](http://site.icu-project.org/), and the libgit2 bindings for Ruby provided by [`rugged`](https://github.com/libgit2/rugged).

[Bundler](https://bundler.io/) v1.10.0 or newer is required for installing the Ruby gem dependencies.

[Docker](https://www.docker.com/) is also required when adding or updating grammars.

These components have their own dependencies - `icu4c`, and `cmake` and `pkg-config` respectively - which you may need to install before you can install Linguist.

For example, on macOS with [Homebrew](http://brew.sh/):

```bash

brew install cmake pkg-config icu4c

brew cask install docker

```

On Ubuntu:

```bash

apt-get install cmake pkg-config libicu-dev docker-ce

```

The latest version of Bundler can be installed with `gem install bundler`.

## Getting started

Before you can start contributing to Linguist, you'll need to set up your environment first.

Clone the repo and run `script/bootstrap` to install its dependencies.

```bash

git clone https://github.com/github/linguist.git

cd linguist/

script/bootstrap

```

To run Linguist from the cloned repository, you will need to generate the code samples first:

```bash

bundle exec rake samples

```

Run this command each time a [sample][samples] has been modified.

To run Linguist from the cloned repository:

```bash

bundle exec bin/github-linguist --breakdown

```

## Adding an extension to a language

We try only to add new extensions once they have some usage on GitHub.

In most cases we prefer that extensions be in use in hundreds of repositories before supporting them in Linguist.

To add support for a new extension:

1. Add your extension to the language entry in [`languages.yml`][languages].

Keep the extensions in alphabetical order, sorted case-sensitively (uppercase before lowercase).

The exception is the primary extension: it should always be first.

1. Add at least one sample for your extension to the [samples directory][samples] in the correct subdirectory.

We prefer examples of real-world code showing common usage.

The more representative of the structure of the language, the better.

1. Open a pull request, linking to a [GitHub search result][search-example] showing in-the-wild usage.

If you are adding a sample, please state clearly the license covering the code.

If possible, link to the original source of the sample.

Additionally, if this extension is already listed in [`languages.yml`][languages] and associated with another language, then sometimes a few more steps will need to be taken:

1. Make sure that example `.yourextension` files are present in the [samples directory][samples] for each language that uses `.yourextension`.

1. Test the performance of the Bayesian classifier with a relatively large number (1000s) of sample `.yourextension` files (ping \*\*@lildude\*\* to help with this).

This ensures we're not misclassifying files.

1. If the Bayesian classifier does a bad job with the sample `.yourextension` files then a [heuristic][] may need to be written to help.

## Adding a language

We try only to add languages once they have some usage on GitHub.

In most cases we prefer that each new file extension be in use in hundreds of repositories before supporting them in Linguist.

To add support for a new language:

1. Add an entry for your language to [`languages.yml`][languages].

Omit the `language\_id` field for now.

1. Add a syntax-highlighting grammar for your language using:

```bash

script/add-grammar https://github.com/JaneSmith/MyGrammar

```

This command will analyze the grammar and, if no problems are found, add it to the repository.

If problems are found, please report them to the grammar maintainer as you will otherwise be unable to add it.

\*\*Please only add grammars that have [one of these licenses][licenses].\*\*

1. Add samples for your language to the [samples directory][samples] in the correct subdirectory.

1. Generate a unique ID for your language by running `script/update-ids`.

1. Open a pull request, linking to [GitHub search results][search-example] showing in-the-wild usage.

Please state clearly the license covering the code in the samples.

Link directly to the original source if possible.

In addition, if your new language defines an extension that's already listed in [`languages.yml`][languages] (such as `.foo`) then sometimes a few more steps will need to be taken:

1. Make sure that example `.foo` files are present in the [samples directory][samples] for each language that uses `.foo`.

1. Test the performance of the Bayesian classifier with a relatively large number (1000s) of sample `.foo` files (ping \*\*@lildude\*\* to help with this).

This ensures we're not misclassifying files.

1. If the Bayesian classifier does a bad job with the sample `.foo` files, then a [heuristic][] may need to be written to help.

Remember, the goal here is to try and avoid false positives!

## Fixing a misclassified language

Most languages are detected by their file extension defined in [`languages.yml`][languages].

For disambiguating between files with common extensions, Linguist applies some [heuristics](/lib/linguist/heuristics.rb) and a [statistical classifier](lib/linguist/classifier.rb).

This process can help differentiate between, for example, `.h` files which could be either C, C++, or Obj-C.

Misclassifications can often be solved by either adding a new filename or extension for the language or adding more [samples][] to make the classifier smarter.

## Fixing syntax highlighting

Syntax highlighting in GitHub is performed using TextMate-compatible grammars.

These are the same grammars used by TextMate, Sublime Text, and Atom.

Every language in [`languages.yml`][languages] is mapped to its corresponding TextMate `scopeName`.

This scope name will be used when picking up a grammar for highlighting.

Assuming your code is being detected as the right language, in most cases syntax highlighting problems are due to a bug in the language grammar rather than a bug in Linguist.

[`vendor/README.md`][grammars] lists all the grammars we use for syntax highlighting on GitHub.com.

Find the one corresponding to your code's programming language and submit a bug report upstream.

If you can, try to reproduce the highlighting problem in the text editor that the grammar is designed for (TextMate, Sublime Text, or Atom) and include that information in your bug report.

You can also try to fix the bug yourself and submit a pull-request.

[TextMate's documentation](https://manual.macromates.com/en/language\_grammars) offers a good introduction on how to work with TextMate-compatible grammars.

Note that Linguist uses [PCRE](https://www.pcre.org/) regular expressions, while TextMate uses [Oniguruma](https://github.com/kkos/oniguruma).

Although they are mostly compatible there might be some differences in syntax and semantics between the two.

You can test grammars using [Lightshow](https://github-lightshow.herokuapp.com).

Once the bug has been fixed upstream, we'll pick it up for GitHub in the next release of Linguist.

## Changing the source of a syntax highlighting grammar

We'd like to ensure Linguist and GitHub.com are using the latest and greatest grammars that are consistent with the current usage but understand that sometimes a grammar can lag behind the evolution of a language or even stop being developed.

This often results in someone grasping the opportunity to create a newer and better and more actively maintained grammar, and we'd love to use it and pass on it's functionality to our users.

Switching the source of a grammar is really easy:

```bash

script/add-grammar --replace MyGrammar https://github.com/PeterPan/MyGrammar

```

This command will analyze the grammar and, if no problems are found, add it to the repository.

If problems are found, please report these problems to the grammar maintainer as you will not be able to add the grammar if problems are found.

\*\*Please only add grammars that have [one of these licenses][licenses].\*\*

Please then open a pull request for the updated grammar.

## Changing the color associated with a language

Many of the colors associated with the languages within Linguist have been in place for a very long time.

The colors were often chosen based on the colors used by the language at the time and since then users will have become familiar with those colors as they appear on GitHub.com.

If you would like to change the color of a language, we ask that you propose your suggested color change to the wider community for your language to gain consensus before submitting a pull request.

Please do this in a community forum or repository used and known by the wider community of that language, not the Linguist repository.

Once you've received consensus that the community is happy with your proposed color change, please feel free to open a PR making the change and link to the public discussion where this was agreed by the community.

If there are official branding guidelines to support the colour choice, please link to those too.

Please note that Linguist currently implements a [color proximity test][] to ensure colors are sufficiently different from one another so you may not be able to use the precise color you want - reds and blues are really popular.

As such, we recommend you test the color change locally before making your plea to the wider language community.

## Testing

You can run the tests locally with:

```bash

bundle exec rake test

```

Sometimes getting the tests running can be too much work, especially if you don't have much Ruby experience.

It's okay: be lazy and let [GitHub Actions](https://github.com/features/actions) run the tests for you.

Just open a pull request and the bot will start cranking away.

Here's our current build status: [![Actions Status](https://github.com/github/linguist/workflows/Run%20Tests/badge.svg)](https://github.com/github/linguist/actions)

## Maintainers

Linguist is maintained with :heart: by:

- \*\*@Alhadis\*\*

- \*\*@BenEddy\*\* (GitHub staff)

- \*\*@Caged\*\* (GitHub staff)

- \*\*@larsbrinkhoff\*\*

- \*\*@lildude\*\* (GitHub staff)

- \*\*@pchaigno\*\*

- \*\*@rafer\*\* (GitHub staff)

- \*\*@shreyasjoshis\*\* (GitHub staff)

As Linguist is a production dependency for GitHub we have a couple of workflow restrictions:

- Anyone with commit rights can merge Pull Requests provided that there is a :+1: from a GitHub staff member.

- Releases are performed by GitHub staff so we can ensure GitHub.com always stays up to date with the latest release of Linguist and there are no regressions in production.

### Releasing

If you are the current maintainer of this gem:

1. Create a branch for the release: `git checkout -b release-vxx.xx.xx`

1. Make sure your local dependencies are up to date: `script/bootstrap`

1. If grammar submodules have not been updated recently, update them: `git submodule update --remote`.

If any submodules are updated:

1. update the `grammars.yml`: `script/grammar-compiler update -f`

1. update the license cache: `bundle exec licensed cache -c vendor/licenses/config.yml`

1. double check no license problems found: `bundle exec licensed status -c vendor/licenses/config.yml`

1. confirm the updated grammars still compile and no new errors have been introduced and none have gone missing: `bundle exec rake check\_grammars`

1. verify and fix any problems identified in the two steps above

1. commit all changes: `git commit -a`

1. Ensure that samples are updated: `bundle exec rake samples`

1. Ensure that tests are green: `bundle exec rake test`

1. Build a test gem `GEM\_VERSION=$(git describe --tags 2>/dev/null | sed 's/-/./g' | sed 's/v//') bundle exec rake build\_gem`

1. Test the test gem:

1. Bump the Gemfile and Gemfile.lock versions for an app which relies on this gem

1. Install the new gem locally

1. Test behavior locally, branch deploy, whatever needs to happen

1. Bump gem version in `lib/linguist/VERSION`, [like this](https://github.com/github/linguist/commit/3212355400974ce5f7873a71eb8b85b1c5f4a6d2).

1. Make a PR to github/linguist, [like this](https://github.com/github/linguist/pull/1238).

1. Build a local gem: `bundle exec rake build\_gem`

1. Merge github/linguist PR

1. Tag and push: `git tag vx.xx.xx; git push --tags`

1. Create a GitHub release with the pushed tag (https://github.com/github/linguist/releases/new) and populate it with a list of the commits from `git log --pretty=format:"- %s" --reverse refs/tags/[OLD TAG]...refs/tags/[NEW TAG]` [like this](https://github.com/github/linguist/releases/tag/v7.2.0)

1. Build a grammars tarball (`./script/build-grammars-tarball`) and attach it to the GitHub release

1. Push to rubygems.pkg.github.com -- `gem push --key github --host https://rubygems.pkg.github.com/github github-linguist-3.0.0.gem`. See [Configuring RubyGems for use with GitHub Package Registry][gpr] for more details.

1. Push to rubygems.org -- `gem push github-linguist-3.0.0.gem`

[grammars]: /vendor/README.md

[heuristic]: https://github.com/github/linguist/blob/master/lib/linguist/heuristics.rb

[languages]: /lib/linguist/languages.yml

[licenses]: https://github.com/github/linguist/blob/257425141d4e2a5232786bf0b13c901ada075f93/vendor/licenses/config.yml#L2-L11

[new-issue]: https://github.com/github/linguist/issues/new

[color proximity test]: https://github.com/github/linguist/blob/master/test/test\_color\_proximity.rb

[samples]: /samples

[search-example]: https://github.com/search?utf8=%E2%9C%93&q=extension%3Aboot+NOT+nothack&type=Code&ref=searchresults

[gpr]: https://help.github.com/en/articles/configuring-rubygems-for-use-with-github-package-registry