# Contributing

First, thank you for contributing!

We love pull requests from everyone. By participating in this project, you

agree to abide by the Hound [code of conduct].

[code of conduct]: https://github.com/houndci/hound/blob/master/CODE\_OF\_CONDUCT.md

Here are a few technical guidelines to follow:

1. Open an [issue][issues] to discuss a new feature.

1. Write tests.

1. Make sure the entire test suite passes locally and on CI.

1. Open a Pull Request.

1. [Squash your commits][squash] after receiving feedback.

1. Party!

[issues]: https://github.com/houndci/hound/issues

[squash]: https://github.com/thoughtbot/guides/tree/master/protocol/git#write-a-feature

## Configure Hound on Your Local Development Environment

1. After cloning the repository, run the setup script: `./bin/setup`

1. Make sure that Postgres, and Redis, are both installed and running locally.

1. Log into your GitHub account and go to your [developer application settings].

1. Under the Developer applications panel - Click on "Register new application"

and fill in the details:

\* Application Name: Hound Development

\* Homepage URL: `http://localhost:5000`

\* Authorization Callback URL: `http://localhost:5000`

1. On the confirmation screen, copy the `Client ID` and `Client Secret` to

`GITHUB\_CLIENT\_ID` and `GITHUB\_CLIENT\_SECRET` in the `.env.local` file.

1. Run `foreman start`. Foreman will start the web server and the Sidekiq

background job queue. \*\*NOTE\*\*: `rails server` will not load the appropriate

environment variables and you'll get a "Missing `secret\_key\_base` for

'development' environment" error. Similarly, `heroku local` and `forego start`

will fail to properly load `.env.local`.

1. Open `localhost:5000` in a browser.

## Setup Ngrok to Allow Webhooks

Ngrok allows Hound to receive webhooks from GitHub. If you'd like to develop or

test a feature involving GitHub sending a pull request notification to your

local Hound server you'll need to have ngrok or something similar set up.

To get started with ngrok, sign up for an [ngrok] account and configure ngrok

locally by installing ngrok and running:

```sh

ngrok authtoken <your-token>

```

1. Launch ngrok on port 5000 (we recommend running ngrok with a custom subdomain

for easy and persistent configuration, but this requires a paid ngrok account.

You can still run Hound with a free ngrok account, but it will require keeping

the GitHub developer application configuration and your `.env.local` files up

to date if your ngrok subdomain changes).

\* If you're using a custom subdomain:

`ngrok http -subdomain=<your-initials>-hound 5000`

\* If you're using a free ngrok plan: `ngrok http 5000`

1. Set the `HOST` variable in your `.env.local` to your ngrok host, e.g.

`<your-subdomain>.ngrok.io`.

1. Change `ENABLE\_HTTPS` to 'yes' in the `.env.local` file. You might need to allow

insecure access to localhost (see [this link] for a possible workaround).

1. Log into your GitHub account and go to your [developer application settings].

1. Under the Developer applications panel - Click on "Register new

application" and fill in the details:

\* Application Name: Hound Development

\* Homepage URL: `https://<your-subdomain>.ngrok.io`

\* Authorization Callback URL: `https://<your-subdomain>.ngrok.io`

1. On the confirmation screen, copy the `Client ID` and `Client Secret` to

`GITHUB\_CLIENT\_ID` and `GITHUB\_CLIENT\_SECRET` in the `.env.local` file.

1. On the [personal access token] page, click "Generate new token" and fill in

token details:

\* Token description: Hound Development

\* Select scopes: `repo` and `user:email`

1. On the confirmation screen, copy the generated token to `HOUND\_GITHUB\_TOKEN`

in the `.env.local` file. Also update `HOUND\_GITHUB\_USERNAME` to be your username.

1. Run `foreman start`. Foreman will start the web server and the Sidekiq

background job queue. \*\*NOTE\*\*: `rails server` will not load the appropriate

environment variables and you'll get a "Missing `secret\_key\_base` for

'development' environment" error. Similarly, `heroku local` and `forego start`

will fail to properly load `.env.local`.

1. Open `https://<your-subdomain>.ngrok.io` in a browser.

[ngrok]: https://ngrok.com

[personal access token]: https://github.com/settings/tokens

[developer application settings]: https://github.com/settings/developers

[this link]: https://superuser.com/questions/772762/how-can-i-disable-security-checks-for-localhost#903159

## Testing

1. Set up your `development` environment as per above.

1. Run `rake` to execute the full test suite.

To test Stripe payments on staging use this fake credit card number.

<table>

<thead>

<tr>

<th>Card</th>

<th>Number</th>

<th>Expiration</th>

<th>CVV</th>

</tr>

</thead>

<tbody>

<tr>

<td>Visa</td>

<td>4242424242424242</td>

<td>Any future date</td>

<td>Any 3 digits</td>

</tr>

</tbody>

</table>

## Linters

The main Hound app (this app) receives PR hooks from GitHub, then it

communicates with the [Linters](https://github.com/houndci/linters) app

(or a few individual linter services) to review changed files in the PR.

Linters communicate back with violations they found, and the Hound app sends

comments back to GitHub.

Here is the list of all the linters and where to find them,

as well as any default configuration they might use:

1. Ruby

\* RuboCop

\* [RuboCop](https://github.com/bbatsov/rubocop)

\* [houndci/linters](https://github.com/houndci/linters/tree/master/lib/linters/rubocop)

\* Flog

\* [Flog](https://github.com/seattlerb/flog)

\* [houndci/linters](https://github.com/houndci/linters/tree/master/lib/linters/flog)

\* Reek

\* [Reek](https://github.com/troessner/reek)

\* [houndci/linters](https://github.com/houndci/linters/tree/master/lib/linters/reek)

1. CoffeeScript

\* [CoffeeLint](https://github.com/clutchski/coffeelint)

\* [houndci/linters](https://github.com/houndci/linters/tree/master/lib/linters/coffeelint)

1. JavaScript

\* JSHint

\* [JSHint](https://github.com/jshint/jshint)

\* [houndci/linters](https://github.com/houndci/linters/tree/master/lib/linters/jshint)

\* ESLint

\* [ESLint](https://github.com/eslint/eslint)

\* [houndci/linters](https://github.com/houndci/linters/tree/master/lib/linters/eslint)

\* [default config](https://raw.githubusercontent.com/houndci/linters/master/config/eslintrc)

1. TypeScript

\* [TSLint](https://github.com/palantir/tslint)

\* [houndci/linters](https://github.com/houndci/linters/tree/master/lib/linters/tslint)

1. SCSS

\* [SCSS-Lint](https://github.com/brigade/scss-lint)

\* [houndci/linters](https://github.com/houndci/linters/tree/master/lib/linters/scss\_lint)

\* [default config](https://raw.githubusercontent.com/houndci/linters/master/config/scss.yml)

1. Haml

\* [HAML-Lint](https://github.com/brigade/haml-lint)

\* [houndci/linters](https://github.com/houndci/linters/tree/master/lib/linters/haml\_lint)

\* [default config](https://raw.githubusercontent.com/houndci/linters/master/config/haml.yml)

1. ERB

\* [ERB Lint](https://github.com/Shopify/erb-lint)

\* [houndci/linters](https://github.com/houndci/linters/tree/master/lib/linters/erb\_lint)

1. Elixir

\* [Credo](https://github.com/rrrene/credo)

\* [houndci/linters](https://github.com/houndci/linters/tree/master/lib/linters/credo)

1. Go

\* [Golint](https://github.com/golang/lint)

\* [houndci/linters](https://github.com/houndci/linters/tree/master/lib/linters/golint)

1. Markdown

\* [Remark](https://github.com/wooorm/remark)

\* [houndci/linters](https://github.com/houndci/linters/tree/master/lib/linters/remark)

1. Swift

\* [SwiftLint](https://github.com/realm/SwiftLint)

\* [houndci/swift](https://github.com/houndci/swift)

\* [default config](https://raw.githubusercontent.com/houndci/swift/master/config/default.yml)

1. Shell scripts

\* [ShellCheck](https://github.com/koalaman/shellcheck)

\* [houndci/linters](https://github.com/houndci/linters/tree/master/lib/linters/shellcheck)

### Writing a Linter

Linters check code snippets for style violations. They operate by polling a

Sidekiq queue.

Linter jobs are created with the following arguments:

\* `commit\_sha` - The git commit SHA of the code snippet. This should be passed

through to the outbound queue.

\* `config` - The configuration for the linter. This will be linter specific.

\* `content` - The source code to check for violations.

\* `filename` - The name of the source file for the code snippet. This should be

passed through to the outbound queue.

\* `linter\_name` - A string that identifies which linter is assigned to this

linter job. This must be passed through to the outbound queue unmodified.

\* `patch` - The patch content from GitHub for the file being reviewed. This

should be passed through to the outbound queue.

\* `pull\_request\_number` - The GitHub pull request number. This should be passed

through to the outbound queue.

Once linting is complete, resulting violations should be posted to the outbound

"CompletedFileReviewJob" queue:

\* `violations` - An array of violation objects. Each violation requires the

following:

\* `line` - The line number where the violation occurred.

\* `message` - A message describing the violation. This will be the contents

of the Pull Request comment.

\* `filename` - The name of the source file for the code snippet. This is

provided by the inbound queue.

\* `commit\_sha` - The git commit SHA of the code snippet. This is provided by the

inbound queue.

\* `linter\_name` - A string that identifies which linter is assigned to this

linter job. This is provided by the inbound queue.

\* `patch` - The patch content from GitHub for the file being reviewed. This is

provided by the inbound queue.

\* `commit\_sha` - The git commit SHA of the code snippet. This is provided by the

inbound queue.

\* `linter\_name` - The name of the linter that received an invalid config file.

\* `pull\_request\_number` - The GitHub pull request number. This is provided by

the inbound queue.

## Deploying

If you have previously run the `bin/setup` script, you can deploy to staging

and production with:

```sh

% bin/deploy staging

% bin/deploy production

```

## Contributor License Agreement

If you submit a Contribution to this application's source code, you hereby grant

to Hound Group LLC a worldwide, royalty-free, exclusive, perpetual and

irrevocable license, with the right to grant or transfer an unlimited number of

non-exclusive licenses or sublicenses to third parties, under the Copyright

covering the Contribution to use the Contribution by all means, including but

not limited to:

\* to publish the Contribution,

\* to modify the Contribution, to prepare Derivative Works based upon or

containing the Contribution and to combine the Contribution with other

software code,

\* to reproduce the Contribution in original or modified form,

\* to distribute, to make the Contribution available to the public, display and

publicly perform the Contribution in original or modified form.