**Introduction**

**Write something nice here!**

First off, thank you for considering contributing to Active Admin. It's people like you that make Active Admin such a great tool.

[source: [Active Admin](https://github.com/activeadmin/activeadmin/blob/master/CONTRIBUTING.md)] **Need more inspiration?** [1] [Read The Docs](http://read-the-docs.readthedocs.org/en/latest/contribute.html) [2] [Mustache.js](https://github.com/janl/mustache.js/#contributing)

**Tell them why they should read your guidelines.**

Following these guidelines helps to communicate that you respect the time of the developers managing and developing this open source project. In return, they should reciprocate that respect in addressing your issue, assessing changes, and helping you finalize your pull requests.

[source: [Hoodie](https://github.com/hoodiehq/hoodie/blob/master/CONTRIBUTING.md)]

**Explain what kinds of contributions you are looking for.**

Keep an open mind! Improving documentation, bug triaging, or writing tutorials are all examples of helpful contributions that mean less work for you.

Elasticsearch is an open source project and we love to receive contributions from our community — you! There are many ways to contribute, from writing tutorials or blog posts, improving the documentation, submitting bug reports and feature requests or writing code which can be incorporated into Elasticsearch itself.

[source: [Elasticsearch](https://github.com/elastic/elasticsearch/blob/master/CONTRIBUTING.md)] **Need more inspiration?** [1] [Devise](https://github.com/plataformatec/devise/wiki/Contributing) [2] [Geocoder](https://github.com/alexreisner/geocoder#known-issue) (“known issue”)

**Explain contributions you are NOT looking for (if any).**

Again, defining this up front means less work for you. If someone ignores your guide and submits something you don’t want, you can simply close it and point to your policy.

Please, don't use the issue tracker for [support questions]. Check whether the #pocoo IRC channel on Freenode can help with your issue. If your problem is not strictly Werkzeug or Flask specific, #python is generally more active. Stack Overflow is also worth considering.

[source: [Flask](https://github.com/pallets/flask/blob/master/CONTRIBUTING.rst)] **Need more inspiration?** [1] [cucumber-ruby](https://github.com/cucumber/cucumber-ruby/blob/master/CONTRIBUTING.md#about-to-create-a-new-github-issue) [2] [Read the Docs](http://read-the-docs.readthedocs.org/en/latest/open-source-philosophy.html#unsupported)

**Ground Rules**

**Set expectations for behavior (yours, and theirs).**

This includes not just how to communicate with others (being respectful, considerate, etc) but also technical responsibilities (importance of testing, project dependencies, etc). Mention and link to your code of conduct, if you have one.

Responsibilities

* Ensure cross-platform compatibility for every change that's accepted. Windows, Mac, Debian & Ubuntu Linux.
* Ensure that code that goes into core meets all requirements in this checklist: <https://gist.github.com/audreyr/4feef90445b9680475f2>
* Create issues for any major changes and enhancements that you wish to make. Discuss things transparently and get community feedback.
* Don't add any classes to the codebase unless absolutely needed. Err on the side of using functions.
* Keep feature versions as small as possible, preferably one new feature per version.
* Be welcoming to newcomers and encourage diverse new contributors from all backgrounds. See the [Python Community Code of Conduct](https://www.python.org/psf/codeofconduct/).

[source: [cookiecutter](https://github.com/audreyr/cookiecutter/blob/master/CONTRIBUTING.rst)] **Need more inspiration?** [1] [Celery](https://github.com/celery/celery/blob/master/CONTRIBUTING.rst#community-code-of-conduct) [2] [geocoder](https://github.com/alexreisner/geocoder#contributing)

**Your First Contribution**

Help people who are new to your project understand where they can be most helpful. This is also a good time to let people know if you follow a label convention for flagging beginner issues.

Unsure where to begin contributing to Atom? You can start by looking through these beginner and help-wanted issues: Beginner issues - issues which should only require a few lines of code, and a test or two. Help wanted issues - issues which should be a bit more involved than beginner issues. Both issue lists are sorted by total number of comments. While not perfect, number of comments is a reasonable proxy for impact a given change will have.

[source: [Atom](https://github.com/atom/atom/blob/master/CONTRIBUTING.md#your-first-code-contribution)] **Need more inspiration?** [1] [Read the Docs](http://docs.readthedocs.org/en/latest/contribute.html#contributing-to-development) [2] [Django](https://docs.djangoproject.com/en/dev/internals/contributing/new-contributors/#first-steps) (scroll down to "Guidelines" as well)

**Bonus points: Add a link to a resource for people who have never contributed to open source before.**

Here are a couple of friendly tutorials you can include: <http://makeapullrequest.com/> and <http://www.firsttimersonly.com/>

Working on your first Pull Request? You can learn how from this *free* series, [How to Contribute to an Open Source Project on GitHub](https://egghead.io/series/how-to-contribute-to-an-open-source-project-on-github).

[source: [React](https://github.com/facebook/react/blob/master/CONTRIBUTING.md#pull-requests)]

As a side note, it helps to use newcomer-friendly language throughout the rest of your document. Here are a couple of examples from [Active Admin](https://github.com/activeadmin/activeadmin/blob/master/CONTRIBUTING.md):

At this point, you're ready to make your changes! Feel free to ask for help; everyone is a beginner at first 😸

If a maintainer asks you to "rebase" your PR, they're saying that a lot of code has changed, and that you need to update your branch so it's easier to merge.

**Getting started**

**Give them a quick walkthrough of how to submit a contribution.**

How you write this is up to you, but some things you may want to include:

* Let them know if they need to sign a CLA, agree to a DCO, or get any other legal stuff out of the way
* If tests are required for contributions, let them know, and explain how to run the tests
* If you use anything other than GitHub to manage issues (ex. JIRA or Trac), let them know which tools they’ll need to contribute

For something that is bigger than a one or two line fix:

1. Create your own fork of the code
2. Do the changes in your fork
3. If you like the change and think the project could use it: \* Be sure you have followed the code style for the project. \* Sign the Contributor License Agreement, CLA, with the jQuery Foundation. \* Note the jQuery Foundation Code of Conduct. \* Send a pull request indicating that you have a CLA on file.

[source: [Requirejs](http://requirejs.org/docs/contributing.html)] **Need more inspiration?** [1] [Active Admin](https://github.com/activeadmin/activeadmin/blob/master/CONTRIBUTING.md#1-where-do-i-go-from-here) [2] [Node.js](https://github.com/nodejs/node/blob/master/CONTRIBUTING.md#code-contributions) [3] [Ember.js](https://github.com/emberjs/ember.js/blob/master/CONTRIBUTING.md#pull-requests)

**If you have a different process for small or "obvious" fixes, let them know.**

Small contributions such as fixing spelling errors, where the content is small enough to not be considered intellectual property, can be submitted by a contributor as a patch, without a CLA.

As a rule of thumb, changes are obvious fixes if they do not introduce any new functionality or creative thinking. As long as the change does not affect functionality, some likely examples include the following:

* Spelling / grammar fixes
* Typo correction, white space and formatting changes
* Comment clean up
* Bug fixes that change default return values or error codes stored in constants
* Adding logging messages or debugging output
* Changes to ‘metadata’ files like Gemfile, .gitignore, build scripts, etc.
* Moving source files from one directory or package to another

[source: [Chef](https://github.com/chef/chef/blob/master/CONTRIBUTING.md#chef-obvious-fix-policy)] **Need more inspiration?** [1] [Puppet](https://github.com/puppetlabs/puppet/blob/master/CONTRIBUTING.md#making-trivial-changes)

**How to report a bug**

**Explain security disclosures first!**

At bare minimum, include this sentence:

If you find a security vulnerability, do NOT open an issue. Email XXXX instead.

If you don’t want to use your personal contact information, set up a “security@” email address. Larger projects might have more formal processes for disclosing security, including encrypted communication. (Disclosure: I am not a security expert.)

Any security issues should be submitted directly to [security@travis-ci.org](mailto:security@travis-ci.org) In order to determine whether you are dealing with a security issue, ask yourself these two questions:

* Can I access something that's not mine, or something I shouldn't have access to?
* Can I disable something for other people?

If the answer to either of those two questions are "yes", then you're probably dealing with a security issue. Note that even if you answer "no" to both questions, you may still be dealing with a security issue, so if you're unsure, just email us at [security@travis-ci.org](mailto:security@travis-ci.org).

[source: [Travis CI](https://github.com/travis-ci/travis-ci/blob/master/CONTRIBUTING.md)] **Need more inspiration?** [1] [Celery](https://github.com/celery/celery/blob/master/CONTRIBUTING.rst#security) [2] [Express.js](https://github.com/expressjs/express/blob/master/Security.md)

**Tell your contributors how to file a bug report.**

You can even include a template so people can just copy-paste (again, less work for you).

When filing an issue, make sure to answer these five questions:

1. What version of Go are you using (go version)?
2. What operating system and processor architecture are you using?
3. What did you do?
4. What did you expect to see?
5. What did you see instead? General questions should go to the golang-nuts mailing list instead of the issue tracker. The gophers there will answer or ask you to file an issue if you've tripped over a bug.

[source: [Go](https://github.com/golang/go/blob/master/CONTRIBUTING.md#filing-issues)] **Need more inspiration?** [1] [Celery](https://github.com/celery/celery/blob/master/CONTRIBUTING.rst#other-bugs) [2] [Atom](https://github.com/atom/atom/blob/master/CONTRIBUTING.md#reporting-bugs) (includes template)

**How to suggest a feature or enhancement**

**If you have a particular roadmap, goals, or philosophy for development, share it here.**

This information will give contributors context before they make suggestions that may not align with the project’s needs.

The Express philosophy is to provide small, robust tooling for HTTP servers, making it a great solution for single page applications, web sites, hybrids, or public HTTP APIs.

Express does not force you to use any specific ORM or template engine. With support for over 14 template engines via Consolidate.js, you can quickly craft your perfect framework.

[source: [Express](https://github.com/expressjs/express#philosophy)] **Need more inspiration?** [Active Admin](https://github.com/activeadmin/activeadmin#goals)

**Explain your desired process for suggesting a feature.**

If there is back-and-forth or signoff required, say so. Ask them to scope the feature, thinking through why it’s needed and how it might work.

If you find yourself wishing for a feature that doesn't exist in Elasticsearch, you are probably not alone. There are bound to be others out there with similar needs. Many of the features that Elasticsearch has today have been added because our users saw the need. Open an issue on our issues list on GitHub which describes the feature you would like to see, why you need it, and how it should work.

[source: [Elasticsearch](https://github.com/elastic/elasticsearch/blob/master/CONTRIBUTING.md#feature-requests)] **Need more inspiration?** [1] [Hoodie](https://github.com/hoodiehq/hoodie/blob/master/CONTRIBUTING.md#feature-requests) [2] [Ember.js](https://github.com/emberjs/ember.js/blob/master/CONTRIBUTING.md#requesting-a-feature)

**Code review process**

**Explain how a contribution gets accepted after it’s been submitted.**

Who reviews it? Who needs to sign off before it’s accepted? When should a contributor expect to hear from you? How can contributors get commit access, if at all?

The core team looks at Pull Requests on a regular basis in a weekly triage meeting that we hold in a public Google Hangout. The hangout is announced in the weekly status updates that are sent to the puppet-dev list. Notes are posted to the Puppet Community community-triage repo and include a link to a YouTube recording of the hangout. After feedback has been given we expect responses within two weeks. After two weeks we may close the pull request if it isn't showing any activity.

[source: [Puppet](https://github.com/puppetlabs/puppet/blob/master/CONTRIBUTING.md#submitting-changes)] **Need more inspiration?** [1] [Meteor](https://meteor.hackpad.com/Responding-to-GitHub-Issues-SKE2u3tkSiH) [2] [Express.js](https://github.com/expressjs/express/blob/master/Contributing.md#becoming-a-committer)

**Community**

If there are other channels you use besides GitHub to discuss contributions, mention them here. You can also list the author, maintainers, and/or contributors here, or set expectations for response time.

You can chat with the core team on <https://gitter.im/cucumber/cucumber>. We try to have office hours on Fridays.

[source: [cucumber-ruby](https://github.com/cucumber/cucumber-ruby/blob/master/CONTRIBUTING.md#talking-with-other-devs)] **Need more inspiration?** [1] [Chef](https://github.com/chef/chef/blob/master/CONTRIBUTING.md#-developer-office-hours) [2] [Cookiecutter](https://github.com/audreyr/cookiecutter#community)

**BONUS: Code, commit message and labeling conventions**

These sections are not necessary, but can help streamline the contributions you receive.

**Explain your preferred style for code, if you have any.**

**Need inspiration?** [1] [Requirejs](http://requirejs.org/docs/contributing.html#codestyle) [2] [Elasticsearch](https://github.com/elastic/elasticsearch/blob/master/CONTRIBUTING.md#contributing-to-the-elasticsearch-codebase)

**Explain if you use any commit message conventions.**

**Need inspiration?** [1] [Angular](https://github.com/angular/material/blob/master/.github/CONTRIBUTING.md#submit) [2] [Node.js](https://github.com/nodejs/node/blob/master/CONTRIBUTING.md#step-3-commit)

**Explain if you use any labeling conventions for issues.**

**Need inspiration?** [1] [StandardIssueLabels](https://github.com/wagenet/StandardIssueLabels#standardissuelabels) [2] [Atom](https://github.com/atom/atom/blob/master/CONTRIBUTING.md#issue-and-pull-request-labels)