# Contributor Guideline

This document provides an overview of how you can participate in improving this project or extending it. We are grateful for all your help: bug reports and fixes, code contributions, documentation or ideas. Feel free to join, we appreciate your support!!

## Communication

### GitHub repositories

Much of the issues, goals and ideas are tracked in the respective projects in GitHub. Please use this channel to report bugs and post ideas.

## git and GitHub

In order to contribute code please:

1. Fork the project on GitHub

2. Clone the project

3. Add changes (and tests)

4. Commit and push

5. Create a merge-request

To have your code merged, see the expectations listed below.

You can find a well-written guide [here](https://help.github.com/articles/fork-a-repo).

Please follow common commit best-practices. Be explicit, have a short summary, a well-written description and references. This is especially important for the merge-request.

Some great guidelines can be found [here](https://wiki.openstack.org/wiki/GitCommitMessages) and [here](http://robots.thoughtbot.com/5-useful-tips-for-a-better-commit-message).

## Expectations

### Don't reinvent the wheel

This hardening project doesn't intend to reinvent the configuration stack for services. Aim to use official configuration projects first and provide hardening as a layer on top. The goal is remove the need for a user to configure all aspects of services and maintain security configuration. This way, the user can still configure a service using the interface provided by the official project.

\* For Chef refer to the official [opscode community cookbooks](http://community.opscode.com/cookbooks).

\* For Puppet head to the [Puppet Forge](https://forge.puppetlabs.com/) and take a node of the Puppet supported modules.

\* For Ansible check the [Ansible Module Index](http://docs.ansible.com/list\_of\_all\_modules.html)

These projects are generally hosted on GitHub as well.

In some cases, we in fact create the full rollout stack, but this is generally the exception ([os-hardening](https://github.com/TelekomLabs/chef-os-hardening), [nginx-hardening](https://github.com/TelekomLabs/chef-nginx-hardening)).

### Be explicit

\* Please avoid using nonsensical property and variable names.

\* Use self-describing attribute names for user configuration.

\* In case of failures, communicate what happened and why a failure occurs to the user. Make it easy to track the code or action that produced the error. Try to catch and handle errors if possible to provide improved failure messages.

### Add tests

The security review of this project is done using integration tests.

Whenever you add a new security configuration, please start by writing a test that checks for this configuration. For example: If you want to set a new attribute in a configuration file, write a test that expects the value to be set first. Then implement your change.

You may add a new feature request by creating a test for whatever value you need.

All tests will be reviewed internally for their validity and overall project direction.

### Document your code

As code is more often read than written, please provide documentation in all projects.

Adhere to the respective guidelines for documentation:

\* Chef generally documents code based explicit readme files. For code documentation please use [yard-chef](https://github.com/rightscale/yard-chef)

\* [Puppet module documentation](http://docs.puppetlabs.com/puppet/latest/reference/modules\_documentation.html)

### Follow coding styles

We generally include test for coding guidelines:

\* Chef follows [Foodcritic](http://acrmp.github.io/foodcritic/)

\* Puppet is checked with [puppet-lint](http://puppet-lint.com/checks/)

\* Ansible is checked by running the playbook with the syntax-check option, e.g. `ansible-playbook foo.yml --syntax-check`

Remember: Code is generally read much more often than written.

### Use Markdown

Wherever possible, please refrain from any other formats and stick to simple markdown.