# Contributing to Sovereign

## Intellectual property

Make sure you agree with the license (GPLv3). See [LICENSE.md](./LICENSE.md) for details.

Code that is committed to the master branch should work with both Debian 8 "Jessie" and Ubuntu 16.04 LTS "Xenial".

## Development environment

You'll want to set up a [local development environment](https://github.com/sovereign/sovereign/wiki/Development-Environment) so that you don't have to test on a remote server.

## Module design principles

Sovereign is an Ansible playbook that uses the modules in this repository to configure a server. Modules should conform to the following design principles.

### Naming

Modules should be named after the software they add (as opposed to the functionality they provide). Soverign is currently inconsistent on this. For example, there are the `ircbouncer` and `blog` modules, but there are also the `owncloud` and `tarsnap` modules. Please name modules after the software used, though, so that it is possible to provide alternatives for functionality.

### Making decisions

A module exists to make decisions about how a service should be installed and configured. Make these decisions and minimize or eliminate configuration options exposed to the user. When in doubt, make a decision, and if the community feedback is vocal enough, only then expose an option.

### Idempotency

A module must be idempotent. If it's run once or many times, the result should be the same. This means that in some cases the user will be left with post-installation finalization work to do. Post-install finalization should be reduced or eliminated if possible, but not at the cost of idempotency.

### Databases

A module that introduces a database-backed service must use PostgreSQL if possible. In order to minimize server load of having two database servers running, MySQL should not be used unless absolutely necessary. Sqlite may be used if persistent data requirements are bounded for all users and are within Sqlite's design limits.

### Registrations

A module should configure the server in a way that minimizes the data posted to other services. This includes names, email addresses, and other personally-identifable information.

### Upgrades

A module's design should anticipate upgrades to the services it provides. Configuration files that work for the current version of the service may become out of date on future versions of the service and lead to difficult-to-find bugs. This also introduces work for maintaining the module. Whenever possible, design the module to use the service to handle initial configuration and upgrades.

### Performance

A module should be designed and implemented to run as quickly as possible in order to minimize the time to run an entire playbook or even the role itself. A small performance penalty here and a small penalty there eventually adds to a very slow deployment system. Performance is important.

### Tests

A module should have tests. TBD: more about this and what the expectation is.

### Design document

A module should have a design description explaining the approach to implementing a service and what tradeoffs were made when choosing the design that was implemented. Do not leave this for comments in a pull request as we want this close to the code for the sake of future maintainers.

The design description should be succinct and to the point. Assume the reader is familiar with Sovereign but not your module. As a rule of thumb, 500-1000 words is about the right length for a module design description.

## Design checklist

Consider the following checklist when reviewing a module's design.

- Does the role create data on the server that is impossible or difficult to reproduce, e.g., private keys? If so, update the tarsnap role to include precious data in backups.

- Does the role need an SSL certificate for a new subdomain? If so, update the letsencrypt tasklist in the common role.

- Does the role add an Apache virtual site? If so, has somebody knowledgable in Apache configuration and security reviewed the configuration?

- Does README.md need to be updated based on new or changed finalization instructions?

## Submitting pull requests

Verify that your changes pass [ansible-lint](https://github.com/willthames/ansible-lint) before submitting a pull request.

Use good commit practices to document your changes. Don't assume the developer reviewing your commits has access to GitHub. The developer could be a future maintainer in a different environment. Similarly, as you address feedback on the pull request, do not assume the reviewer has access to GitHub.

When you issue a pull request, please specify what distribution you used for testing (if any). Code that is committed to the master branch should work with both Debian 7 and Ubuntu 14.04 LTS. Support for Debian 8 is coming.