# Contribution Guidelines

When contributing a new feature, a bug fix, a new theme, or any other change to Bash-it, please consider the following guidelines.

Most of this is common sense, but please try to stick to the conventions listed here.

## Issues

\* When opening a new issue in the issue tracker, please include information about which \_Operating System\_ you're using, and which version of \_Bash\_.

\* In many cases, it also makes sense to show which Bash-it plugins you are using.

This information can be obtained using `bash-it show plugins`.

\* If the issue happens while loading Bash-it, please also include your `~/.bash\_profile` or `~/.bashrc` file,

as well as the install location of Bash-it (default should be `~/.bash\_it`).

\* When reporting a bug or requesting a new feature, consider providing a Pull Request that fixes the issue or can be used as a starting point for the new feature.

Don't be afraid, most things aren't that complex...

## Pull Requests

\* Fork the Bash-it repo, create a new feature branch from \_master\_ and apply your changes there.

Create a \_Pull Request\_ from your feature branch against Bash-it's \_master\_ branch.

\* Limit each Pull Request to one feature.

Don't bundle multiple features/changes (e.g. a new \_Theme\_ and a fix to an existing plugin) into a single Pull Request - create one PR for the theme, and a separate PR for the fix.

\* For complex changes, try to \_squash\_ your changes into a single commit before

pushing code. Once you've pushed your code and opened a PR, please refrain

from force-pushing changes to the PR branch – remember, Bash-it is a

distributed project and your branch may be in use already.

\* When in doubt, open a PR with too many commits. Bash-it is a learning project

for everyone involved. Showing your work provides a great history for folks

to learn what works and what didn't.

## Code Style

\* Try to stick to the existing code style. Please don't reformat or change the syntax of existing code simply because you don't like that style.

\* Indentation is using spaces, not tabs. Most of the code is indented with 2 spaces, some with 4 spaces. Please try to stick to 2 spaces.

If you're using an editor that supports [EditorConfig](http://EditorConfig.org), the editor should automatically use the settings defined in Bash-it's [.editorconfig file](.editorconfig).

\* When creating new functions, please use a dash ("-") to separate the words of the function's name, e.g. `my-new-function`.

Don't use underscores, e.g. `my\_new\_function`.

\* Internal functions that aren't to be used by the end user should start with an underscore, e.g. `\_my-new-internal-function`.

\* Use the provided meta functions to document your code, e.g. `about-plugin`, `about`, `group`, `param`, `example`.

This will make it easier for other people to use your new functionality.

Take a look at the existing code for an example (e.g. [the base plugin](plugins/available/base.plugin.bash)).

\* When adding files, please use the existing file naming conventions, e.g. plugin files need to end in `.plugin.bash`.

This is important for the installation functionality.

\* When using the `$BASH\_IT` variable, please always enclose it in double quotes to ensure that the code also works when Bash-it is installed in a directory that contains spaces in its name: `for f in "${BASH\_IT}/plugins/available"/\*.bash ; do echo "$f" ; done`

\* Bash-it supports Bash 3.2 and higher. Please don't use features only available in Bash 4, such as associative arrays.

## Unit Tests

When adding features or making changes/fixes, please run our growing unit test suite to ensure that you did not break existing functionality.

The test suite does not cover all aspects of Bash-it, but please run it anyway to verify that you did not introduce any regression issues.

Any code pushed to GitHub as part of a Pull Request will automatically trigger a continuous integration build on [Travis CI](https://travis-ci.org/Bash-it/bash-it), where the test suite is run on both Linux and macOS.

The Pull Request will then show the result of the Travis build, indicating whether all tests ran fine, or whether there were issues.

Please pay attention to this, Pull Requests with build issues will not be merged.

Adding new functionality or changing existing functionality is a good opportunity to increase Bash-it's test coverage.

When you're changing the Bash-it codebase, please consider adding some unit tests that cover the new or changed functionality.

Ideally, when fixing a bug, a matching unit test that verifies that the bug is no longer present, is added at the same time.

To run the test suite, simply execute the following in the directory where you cloned Bash-it:

```bash

test/run

```

This command will ensure that the [Bats Test Framework](https://github.com/bats-core/bats-core) is available in the local `test\_lib` directory (Bats is included as a Git submodule) and then run the test suite found in the [test](test) folder.

The test script will execute each test in turn, and will print a status for each test case.

When adding new test cases, please take a look at the existing test cases for examples.

The following libraries are used to help with the tests:

\* Test Framework: https://github.com/bats-core/bats-core

\* Support library for Bats-Assert: https://github.com/ztombol/bats-support

\* General `assert` functions: https://github.com/ztombol/bats-assert

\* File `assert` functions: https://github.com/ztombol/bats-file

When verifying test results, please try to use the `assert` functions found in these libraries.

## Features

\* When adding new completions or plugins, please don't simply copy existing tools into the Bash-it codebase, try to load/integrate the tools instead.

An example is using `nvm`: Instead of copying the existing `nvm` script into Bash-it, the `nvm.plugin.bash` file tries to load an existing installation of `nvm`.

This means an additional step for the user (installing `nvm` from its own repo, or through a package manager),

but it will also ensure that `nvm` can be upgraded in an easy way.

## Themes

\* When adding a new theme, please include a screenshot and a short description about what makes this theme unique in the Pull Request's description field.

Please do not add theme screenshots to the repo itself, as they will add unnecessary bloat to the repo.

The project's Wiki has a \_Themes\_ page where you can add a screenshot if you want.

\* Ideally, each theme's folder should contain a `README.md` file describing the theme and its configuration options.