# Home Assistant Custom Component Cookiecutter

**Oncleben31**

**Dec 15, 2020**

# CONTENTS

CHAPTER

ONE

# QUICKSTART GUIDE

## 1.1 Requirements

Install Cookiecutter:

```
$ pipx install cookiecutter
```

pipx is preferred, but you can also install with `pip install --user`.

It is recommended to set up Python 3.7, 3.8, or 3.9 using pyenv.

## 1.2 Creating a project

Generate a Home Assistant custom component project by using the following command:

```
$ cookiecutter gh:oncleben31/cookiecutter-homeassistant-custom-component \
  --checkout="2020.11.16"
```

Follow the instructions to customize the generated project

| Setting | Definition |
|---|---|
| `friendly_name` | Integration name used in configuration UI. |
| `project_name` | Project name on GitHub. |
| `domain_name` | Integration domain name |
| `class_name_prefix` | Prefix to be use in classes name |
| `github_user` | GitHub user hosting the repository |
| `version` | Initial version of the component |

Change to the root directory of your new project, and create a Git repository:

```
$ git init
$ git add .
$ git commit
```

## 1.3 Setup the development container

The development container allows to work in a local and dedicated Home Assitant instance to test your custom component. To launch it you need to have already installed Docker, Visual Studio Code (VSC) and the Visual Studio Code Remote - Containers extension.

Open your local copy of the repository with VSC:

```
$ code .
```

Visual Studio Code starts and you are asked to "Reopen in Container", this will start the build of the container.

When done, you can launch the local instance of Home Assistant by running the task `Run Home Assistant on port 9123`.

Use your preferred browser to open the URL `http://localhost:9123`.

Initalize your Home Assistant local instance by following the onboarding workflow.

When setup, you can go to **Configuration** -> **Integrations** menu, clic the + button and search the name you have given to the custom component.

Follow the config flow of the custom component to integrate it in Home Assistant.

Now you are all set to modify the code and develop your ideas !

## 1.4 Advanced usages

### 1.4.1 Add a logo

You have the possibility to add a logo to be used in the integrations configuration UI. To do so, visit the home-assistant/brands repository on GitHub and follow the instructions.

### 1.4.2 Step by step debugging

Step by step debugging is easy with Visual Studio Code. You have to install in Home Assistant the PTVSD integration and follow the documentation instructions to setup VSC. Then you will be able to connect the VSC debuger to the local Home Assistant instance.

### 1.4.3 Deploy with HACS

HACS is the community store. You can ease the installation of your custom component by making it compatible with HACS.

The template have already the tools do do that: `hacs.json` and `info.md` files. The Publish documentation explains how to set those files and the different options you have to integrate your custom component in the HACS network.

## 1.5 Known limitations

- **If you plan to host the generated repository in a GitHub organization you will need manual modifications**.

  Currently the template work well when the repostory is hosted in a GitHub individual account, where URL name and code owner are the same. If you want to use an organization, it is recommended to use the name of this organization for `github_user` settings and modify manually where it's needed afer generation with Cookiecutter.

CHAPTER

TWO

# CONTRIBUTOR GUIDE

Thank you for your interest in improving the Home Assistant Custom Component Cookiecutter. This project is open-source under the MIT license and welcomes contributions in the form of bug reports, feature requests, and pull requests.

Here is a list of important resources for contributors:

- Source Code
- Documentation
- Issue Tracker
- Code of Conduct

## 2.1 How to report a bug

Report bugs on the Issue Tracker.

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

- Which operating system and Python version are you using?
- Which version of this project are you using?
- What did you do?
- What did you expect to see?
- What did you see instead?

The best way to get your bug fixed is to provide a test case, and/or steps to reproduce the issue.

## 2.2 How to request a feature

Request features on the Issue Tracker.

## 2.3 How to set up your development environment

You need Python 3.7+ and the following tools:

- Cookiecutter
- Nox
- Docker
- Visual Studio Code

Fork the repository on GitHub, and clone the fork to your local machine. You can now generate a project from your development version:

```
$ cookiecutter path/to/cookiecutter-homeassistant-custom-component
```

## 2.4 How to test the project

TBD

At least you should ensure the integration generated is working in Home Assistant when launched by Visual Studio Code in a devcontainer.

## 2.5 How to submit changes

Open a pull request to submit changes to this project.

Your pull request needs to meet the following guidelines for acceptance:

- The Nox test suite must pass without errors and warnings.
- Include unit tests. This project maintains 100% code coverage.
- If your changes add functionality, update the documentation accordingly.

Feel free to submit early, though—we can always iterate on this.

It is recommended to open an issue before starting work on anything. This will allow a chance to talk it over with the owners and validate your approach.

## 2.6 How to accept changes

*You need to be a project maintainer to accept changes.*

Before accepting a pull request, go through the following checklist:

- The PR must pass all checks.
- The PR must have a descriptive title.
- The PR should be labelled with the kind of change (see below).

Release notes are pre-filled with titles and authors of merged pull requests. Labels group the pull requests into sections. The following list shows the available sections, with associated labels in parentheses:

- Breaking Changes (`breaking`)

- Features (`enhancement`)
- Removals and Deprecations (`removal`)
- Fixes (`bug`)
- Performance (`performance`)
- Testing (`testing`)
- Continuous Integration (`ci`)
- Documentation (`documentation`)
- Refactoring (`refactoring`)
- Style (`style`)
- Dependencies (`dependencies`)

To merge the pull request, follow these steps:

1. Click **Squash and Merge**. (Select this option from the dropdown menu of the merge button, if it is not shown.)
2. Click **Confirm squash and merge**.
3. Click **Delete branch**.

## 2.7 How to make a release

*You need to be a project maintainer to make a release.*

Before making a release, go through the following checklist:

- All pull requests for the release have been merged.
- The default branch passes all checks.

Releases are made by publishing a GitHub Release. A draft release is being maintained based on merged pull requests. To publish the release, follow these steps:

1. Click **Edit** next to the draft release.
2. Enter a tag with the new version.
3. Enter the release title, also the new version.
4. Edit the release description, if required.
5. Click **Publish Release**.

Version numbers adhere to Calendar Versioning, of the form `YYYY.MM.DD`.

After publishing the release, the following automated steps are triggered:

- The Git tag is applied to the repository.
- Read the Docs builds a new stable version of the documentation.

CHAPTER

THREE

# CONTRIBUTOR COVENANT CODE OF CONDUCT

## 3.1 Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

## 3.2 Our Standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

## 3.3 Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

## 3.4 Scope

This Code of Conduct applies within all community spaces, and also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

## 3.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at oncleben31@gmail.com. All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

## 3.6 Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

### 3.6.1 1. Correction

**Community Impact**: Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

**Consequence**: A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

### 3.6.2 2. Warning

**Community Impact**: A violation through a single incident or series of actions.

**Consequence**: A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

### 3.6.3 3. Temporary Ban

**Community Impact**: A serious violation of community standards, including sustained inappropriate behavior.

**Consequence**: A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

### 3.6.4 4. Permanent Ban

**Community Impact**: Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

**Consequence**: A permanent ban from any sort of public interaction within the community.

## 3.7 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 2.0, available at https://www.contributor-covenant.org/version/2/0/code_of_conduct.html.

Community Impact Guidelines were inspired by Mozilla's code of conduct enforcement ladder.

For answers to common questions about this code of conduct, see the FAQ at https://www.contributor-covenant.org/faq. Translations are available at https://www.contributor-covenant.org/translations.

CHAPTER

FOUR

# MIT LICENSE



Cookiecutter template for a Home Assistant custom component based on the blueprint template.
This project is the fusion of cookiecutter-homeassistant-component, blueprint and cookiecutter-hypermodern-python projects.

CHAPTER
FIVE

# USAGE

```
$ cookiecutter gh:oncleben31/cookiecutter-homeassistant-custom-component \
  --checkout="2020.11.16"
```

CHAPTER
SIX

# FEATURES

- Ready to use Home Assistant custom component
- UI configuration with config Flow
- Translations
- Development and testing in Visual Studio Code development container
- HACS ready
- Continuous integration with GitHub Actions

You can find a repository created with this cookiecutter template in the cookiecutter-homeassistant-custom-component-instance example.

CHAPTER
SEVEN

# FAQ

*What is this project about?*

The mission of this project is to provide Home Assistant custom component developers a ready-to-use template with best practices from Home Assistant developers documentation and from Hypermodern Python blog articles.