# Contributing to Angular DevKit

We would love for you to contribute to DevKit and help make it even better

than it is today! As a contributor, here are the guidelines we would like you

to follow:

- [Code of Conduct](#coc)

- [Question or Problem?](#question)

- [Issues and Bugs](#issue)

- [Feature Requests](#feature)

- [Submission Guidelines](#submit)

- [Coding Rules](#rules)

- [Commit Message Guidelines](#commit)

- [Signing the CLA](#cla)

- [Updating the Public API](#public-api)

## <a name="coc"></a> Code of Conduct

Help us keep Angular open and inclusive. Please read and follow our [Code of Conduct][coc].

## <a name="question"></a> Got a Question or Problem?

Please, do not open issues for the general support questions as we want to keep GitHub issues for

bug reports and feature requests. You've got much better chances of getting your question answered

on [StackOverflow](https://stackoverflow.com/questions/tagged/angular-devkit) where the questions

should be tagged with tag `angular-cli` or `angular-devkit`.

StackOverflow is a much better place to ask questions since:

- there are thousands of people willing to help on StackOverflow

- questions and answers stay available for public viewing so your question / answer might help someone else

- StackOverflow's voting system assures that the best answers are prominently visible.

To save your and our time we will be systematically closing all the issues that are requests for

general support and redirecting people to StackOverflow.

If you would like to chat about the question in real-time, you can reach out via [our gitter channel][gitter].

## <a name="issue"></a> Found an Issue?

If you find a bug in the source code or a mistake in the documentation, you can help us by

[submitting an issue](#submit-issue) to our [GitHub Repository][github]. Even better, you can

[submit a Pull Request](#submit-pr) with a fix.

## <a name="feature"></a> Want a Feature?

You can \*request\* a new feature by [submitting an issue](#submit-issue) to our [GitHub

Repository][github]. If you would like to \*implement\* a new feature, please submit an issue with

a proposal for your work first, to be sure that we can use it.

Please consider what kind of change it is:

\* For a \*\*Major Feature\*\*, first open an issue and outline your proposal so that it can be

discussed. This will also allow us to better coordinate our efforts, prevent duplication of work,

and help you to craft the change so that it is successfully accepted into the project.

\* \*\*Small Features\*\* can be crafted and directly [submitted as a Pull Request](#submit-pr).

## <a name="submit"></a> Submission Guidelines

### <a name="submit-issue"></a> Submitting an Issue

Before you submit an issue, please search the issue tracker, maybe an issue for your problem already exists and the discussion might inform you of workarounds readily available.

We want to fix all the issues as soon as possible, but before fixing a bug we need to reproduce and confirm it. Having a reproducible scenario gives us wealth of important information without going back & forth to you with additional questions like:

- version of Angular CLI used

- `.angular-cli.json` or `angular.json` configuration

- version of Angular DevKit used

- 3rd-party libraries and their versions

- and most importantly - a use-case that fails

A minimal reproduce scenario using allows us to quickly confirm a bug (or point out coding problem) as well as confirm that we are fixing the right problem.

We will be insisting on a minimal reproduce scenario in order to save maintainers time and ultimately be able to fix more bugs. Interestingly, from our experience users often find coding problems themselves while preparing a minimal repository. We understand that sometimes it might be hard to extract essentials bits of code from a larger code-base but we really need to isolate the problem before we can fix it.

Unfortunately we are not able to investigate / fix bugs without a minimal reproduction, so if we don't hear back from you we are going to close an issue that don't have enough info to be reproduced.

You can file new issues by selecting from our [new issue templates](https://github.com/angular/angular-cli/issues/new/choose) and filling out the issue template.

### <a name="submit-pr"></a> Submitting a Pull Request (PR)

Before you submit your Pull Request (PR) consider the following guidelines:

\* Search [GitHub](https://github.com/angular/angular-cli/pulls) for an open or closed PR

that relates to your submission. You don't want to duplicate effort.

\* Please sign our [Contributor License Agreement (CLA)](#cla) before sending PRs.

We cannot accept code without this.

\* Make your changes in a new git branch:

```shell

git checkout -b my-fix-branch master

```

\* Create your patch, \*\*including appropriate test cases\*\*.

\* Follow our [Coding Rules](#rules).

\* Run the full Angular CLI and DevKit test suite, as described in the [developer documentation][dev-doc],

and ensure that all tests pass (coming soon).

\* Commit your changes using a descriptive commit message that follows our

[commit message conventions](#commit). Adherence to these conventions

is necessary because release notes are automatically generated from these messages.

```shell

git commit -a

```

Note: the optional commit `-a` command line option will automatically "add" and "rm" edited files.

\* Push your branch to GitHub:

```shell

git push origin my-fix-branch

```

\* In GitHub, send a pull request to `angular/angular-cli:master`.

\* If we suggest changes then:

\* Make the required updates.

\* Re-run the Angular DevKit test suites to ensure tests are still passing.

\* Once your PR is approved and you are done with any follow up changes:

\* Rebase to the current master to pre-emptively address any merge conflicts.

```shell

git rebase master -i

git push -f

```

\* Add the `PR action: merge` label and the correct

[target label](https://github.com/angular/angular/blob/master/docs/TRIAGE\_AND\_LABELS.md#pr-target)

(if PR author has the project collaborator status, or else the last reviewer

should do this).

\* The current caretaker will merge the PR to the target branch(es) within 1-2

business days.

That's it! ? Thank you for your contribution!

#### After your pull request is merged

After your pull request is merged, you can safely delete your branch and pull the changes

from the main (upstream) repository:

\* Delete the remote branch on GitHub either through the GitHub web UI or your local shell as follows:

```shell

git push origin --delete my-fix-branch

```

\* Check out the master branch:

```shell

git checkout master -f

```

\* Delete the local branch:

```shell

git branch -D my-fix-branch

```

\* Update your master with the latest upstream version:

```shell

git pull --ff upstream master

```

## <a name="rules"></a> Coding Rules

To ensure consistency throughout the source code, keep these rules in mind as you are working:

\* All features or bug fixes \*\*must be tested\*\* by one or more specs (unit-tests or e2e-tests).

\* All public API methods \*\*must be documented\*\*. (Details TBC).

\* We follow [Google's JavaScript Style Guide][js-style-guide], but wrap all code at

\*\*100 characters\*\*.

## <a name="commit"></a> Commit Message Guidelines

We have very precise rules over how our git commit messages can be formatted. This leads to \*\*more

readable messages\*\* that are easy to follow when looking through the \*\*project history\*\*. But also,

we use the git commit messages to \*\*generate the Angular DevKit change log\*\*.

### Commit Message Format

Each commit message consists of a \*\*header\*\*, a \*\*body\*\* and a \*\*footer\*\*. The header has a special

format that includes a \*\*type\*\*, a \*\*scope\*\* and a \*\*subject\*\*:

```

<type>(<scope>): <subject>

<BLANK LINE>

<body>

<BLANK LINE>

<footer>

```

The \*\*header\*\* is mandatory and the \*\*scope\*\* of the header is optional.

Any line of the commit message cannot be longer 100 characters! This allows the message to be easier

to read on GitHub as well as in various git tools.

### Revert

If the commit reverts a previous commit, it should begin with `revert: `, followed by the header of the reverted commit. In the body it should say: `This reverts commit <hash>.`, where the hash is the SHA of the commit being reverted.

### Type

Must be one of the following:

\* \*\*build\*\*: Changes that affect the build system or external dependencies. [2]

\* \*\*ci\*\*: Changes to our CI configuration files and scripts. [2]

\* \*\*docs\*\*: Documentation only changes.

\* \*\*feat\*\*: A new feature. [1]

\* \*\*fix\*\*: A bug fix. [1]

\* \*\*refactor\*\*: A code change that neither fixes a bug nor adds a feature

\* \*\*release\*\*: A release commit. Must only include version changes. [2]

\* \*\*revert\*\*: A git commit revert. The description must include the original commit message. [2]

\* \*\*style\*\*: Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc).

\* \*\*test\*\*: Adding missing tests or correcting existing tests.

<sup>[1] This type MUST have a scope. See the next section for more information.</sup><br/>

<sup>[2] This type MUST NOT have a scope. It only applies to general scripts and tooling.</sup>

### Scope

The scope should be the name of the npm package affected as perceived by the person reading changelog generated from the commit messages.

The following is the list of supported scopes:

\* \*\*@angular/cli\*\*

\* \*\*@angular/pwa\*\*

\* \*\*@angular-devkit/architect\*\*

\* \*\*@angular-devkit/architect-cli\*\*

\* \*\*@angular-devkit/build-angular\*\*

\* \*\*@angular-devkit/build-ng-packagr\*\*

\* \*\*@angular-devkit/build-optimizer\*\*

\* \*\*@angular-devkit/build-webpack\*\*

\* \*\*@angular-devkit/core\*\*

\* \*\*@angular-devkit/schematics\*\*

\* \*\*@angular-devkit/schematics-cli\*\*

\* \*\*@ngtools/webpack\*\*

\* \*\*@schematics/angular\*\*

\* \*\*@schematics/schematics\*\*

\* \*\*@schematics/update\*\*

### Subject

The subject contains succinct description of the change:

\* use the imperative, present tense: "change" not "changed" nor "changes"

\* don't capitalize first letter

\* be concise and direct

\* no dot (.) at the end

### Examples

Examples of valid commit messages:

\* `fix(@angular/cli): prevent the flubber from grassing`

\* `refactor(@schematics/angular): move all JSON classes together`

Examples of invalid commit messages:

\* `fix(@angular/cli): add a new XYZ command`

This is a feature, not a fix.

\* `ci(@angular/cli): fix publishing workflow`

The `ci` type cannot have a scope.

### Body

Just as in the \*\*subject\*\*, use the imperative, present tense: "change" not "changed" nor "changes".

The body should include the motivation for the change and contrast this with previous behavior.

### Footer

The footer should contain any information about \*\*Breaking Changes\*\* and is also the place to

reference GitHub issues that this commit \*\*Closes\*\*.

\*\*Breaking Changes\*\* should start with the word `BREAKING CHANGE:` with a space or two newlines. The rest of the commit message is then used for this.

A detailed explanation can be found in this [document][commit-message-format].

## <a name="cla"></a> Signing the CLA

Please sign our Contributor License Agreement (CLA) before sending pull requests. For any code

changes to be accepted, the CLA must be signed. It's a quick process, we promise!

\* For individuals we have a [simple click-through form][individual-cla].

\* For corporations we'll need you to

[print, sign and one of scan+email, fax or mail the form][corporate-cla].

[coc]: https://github.com/angular/code-of-conduct/blob/master/CODE\_OF\_CONDUCT.md

[commit-message-format]: https://docs.google.com/document/d/1QrDFcIiPjSLDn3EL15IJygNPiHORgU1\_OOAqWjiDU5Y/edit#

[corporate-cla]: http://code.google.com/legal/corporate-cla-v1.0.html

[dev-doc]: https://github.com/angular/angular-cli/blob/master/packages/angular/cli/README.md#development-hints-for-working-on-angular-cli

[GitHub]: https://github.com/angular/angular-cli

[gitter]: https://gitter.im/angular/angular-cli

[individual-cla]: http://code.google.com/legal/individual-cla-v1.0.html

[js-style-guide]: https://google.github.io/styleguide/jsguide.html

[stackoverflow]: http://stackoverflow.com/questions/tagged/angular-devkit

## <a name="public-api"></a> Updating the Public API

Our Public API is protected with TS API Guardian. This is a tool that keeps track of public API surface of our packages.

To test if your change effect the public API you need to run the API guardian on that particular package.

For example in case `@angular-devkit/core` package was modified you need to run:

```bash

yarn bazel test //etc/api:angular\_devkit\_core\_api

```

You can also test all packages by running:

```bash

yarn bazel test //etc/api ...

```

If you modified the public API, the test will fail. To update the golden files you need to run:

```bash

yarn bazel run //etc/api:angular\_devkit\_core\_api.accept

```

\*\*Note\*\*: In some cases we use aliased symbols to create namespaces.

Example:

```javascript

import \* as foo from './foo';

export { foo };

```

There are currently not supported by the API guardian.

To overcome this limitation we created `\_golden-api.ts` in certain packages.

When adding a new API, it might be the case that you need to add it to `\_golden-api.ts`.