# Contribution Guide

\*\*cdk8s\*\* is licensed under [Apache 2.0](./LICENSE) and accepts contributions via

GitHub pull requests. This document outlines some of the conventions on commit

message formatting, contact points for developers, and other resources to help

get contributions into cdk8s.

You can contribute to \*\*cdk8s\*\* in many ways. Contributions of all shapes and sizes are

welcome and celebrated:

- [Reporting Issues](#reporting-issues)

- [Code Contributions](#code-contributions)

- [RFCs](#rfcs)

- [For Maintainers](#for-maintainers)

We follows the [CNCF Community Code of

Conduct](https://github.com/cncf/foundation/blob/master/code-of-conduct.md)

## Reporting Issues

If any part of the project has bugs or documentation mistakes, please let us

know by [raising an issue][new-issue]. We treat bugs and mistakes very seriously

and believe no issue is too small. Before creating a bug report, please check

that an issue reporting the same problem does not already exist.

An issue can either be a \*\*bug report\*\* or a \*\*suggestion\*\*. If you wish to ask

a question or seek guidance, please consider one of the other [support

channels](#getting-help).

[new-issue]: ./issues/new/choose

### Bug reports

To make the bug report accurate and easy to understand, please try to create bug

reports that are:

- \*\*Specific\*\*: Include as much details as possible: which version, what

environment, what configuration, etc.

- \*\*Reproducible\*\*: Include the steps to reproduce the problem. We understand

some issues might be hard to reproduce, please includes the steps that might

lead to the problem. If possible, please provide a minimal code snippet that

reproduces the bug.

- \*\*Isolated\*\*: Please try to isolate and reproduce the bug with minimum

dependencies. It would significantly slow down the speed to fix a bug if too

many dependencies are involved in a bug report.

- \*\*Unique\*\*: Do not duplicate existing bug report.

- \*\*Scoped\*\*: One bug per report. Do not follow up with another bug inside one

report.

We might ask for further information to locate a bug. A duplicated bug report

will be closed.

Submit a bug report [here](new-issue).

### Suggestions

We also accept suggestions for new features or missing capabilities as GitHub

issues. The most important aspect of a suggestion issue is to provide as many

details as possible about your \*\*use case\*\* and less focus on the solution. It

is usually possible to support different use cases in many different ways, and

we need to understand the motivation before we dive into a solution.

If you wish to suggest a major change to the project, please consider to submit

an [RFC](#rfcs) instead of a simple issue. An RFC also starts with a GitHub

issue.

Submit a suggestion [here](new-issue)

## Code Contributions

The general workflow for code contributions:

1. Submit/find an issue

2. Clone this repo

3. Make your code change

4. Write tests & update docs

5. Build & test locally

6. Submit a pull request

7. (Iterate)

8. Your PR will be approved and merged

### Tracking issue

All pull requests should be tracked with a GitHub issue.

You should search for an [existing

issue](https://github.com/awslabs/cdk8s/issues) or raise a [new

bug or suggestion](#reporting-issues).

Add a comment indicating you are willing to pick it up in order to ensure no one

else is currently working on it.

If this is a major contribution, consider submitting an [RFC](#rfcs) to obtain

feedback from the community and maintainers.

### Development environment

Prerequisites:

- [Node.js](https://nodejs.org/en/) >= 12.14.0

- [yarn](https://classic.yarnpkg.com/lang/en/) >= 1.22.0

Prepare your environment:

1. Fork this repo and obtain a local clone.

2. Install all dependencies: `yarn install`

3. Run `yarn build` to build all modules.

### Linking against this repository

The script `./tools/link-all.sh` can be used to generate symlinks to all modules in this repository under some `node\_module`

directory. This can be used to develop against this repo as a local dependency.

One can use the `postinstall` script to symlink this repo:

```json

{

"scripts": {

"postinstall": "../cdk8s/tools/link-all.sh"

}

}

```

This assumes this repo is a sibling of the target repo and will install the CDK as a linked dependency during

\_\_yarn install\_\_.

### Testing

This project includes per-module unit tests and project-wide integration tests.

#### Unit tests

Unit tests are located under the `test/` directory within each module and use the [jest](https://jestjs.io/) framework.

To run unit tests, execute `yarn test` either from the root of the repo (to unit test all modules) or from individual module directories:

yarn test

Out tests utilize [jest snapshot testing](https://jestjs.io/docs/en/snapshot-testing). In case a snapshot needs to be updated, just run:

yarn test -u

#### Integration tests

Integration tests are executed \*after\* we bundle the release. This means that in order to execute integration tests you'll need to create a bundle by running the following command from the root of the repo:

yarn run package

This will result in `./dist` that contains all the ready-to-publish artifacts.

Now, you can run integration tests via:

yarn integ

Our integration tests also utilize snapshot testing. To update integration test snapshots, run:

yarn integ:update

Read [this](./test/README.md) for more details about integration testing in this project.

### Pull Requests

We use the PR title when we automatically generate the change log for each

release. Therefore please following these guidelines to the letter:

- \*\*PR title\*\*:

- Must adhere to [conventional

commits](https://www.conventionalcommits.org/en/v1.0.0/).

- All lowercase with no period at the end of the title

- If this is a `fix` (bug) the title should describe the bug

- If this is a `feat` (feature) the title should describe the feature

- \*\*PR description\*\*:

- Describe how did you fix the bug or what changes you had to make in order to

implement the feature

- Indicate `fixes #NNN` or `resolves #NNN` with the tracking issue number.

- If you \*had\* to test your change manually, describe how you tested it and

paste the test results.

- If this is a breaking change, the last paragraph should describe the

breaking change with the prefix `BREAKING CHANGE: xxxxxx`.

## RFCs

An RFC (request for comments) is a document that proposes and details a change

or addition to cdk8s. It also is a process for reviewing and discussing the

proposal and tracking its implementation. "Request for comments" means a request

for discussion and oversight about the future of cdk8s from contributors and

users. It is an open forum for suggestions, questions, and feedback.

To create an RFC follow this process:

1. Create an [issue][new-issue] which will be the tracking issue for this RFC.

- Title should represent the title of the RFC.

- Description should provide the motivation for the RFC.

2. Create a markdown file based off of

[rfc/0000-template.md](rfc/0000-template.md) under the

`rfc/<nnnn>-<title-of-rfc>` where `<nnnn>` is the tracking issue number and

`<title-of-rfc>` is a symbolic name for the title. For example:

`rfc/0030-construct-operators.md`.

3. File a pull request with this markdown file. The title of the PR should

indicate `rfc: <nnnn> <same as issue title>`.

4. The RFC will be reviewed as a pull request and once merged it means it is

ready for implementation.

## For Maintainers

This section includes information that is relevant for the maintainers of the project.

### Version

The current version of the project is mastered in the root `lerna.json` file. All other

package.json files use `0.0.0`. This allows bumping a new version without

needing to modify multiple files and avoid merge conflicts in post-release rebases.

### Release Protocol

To release a new version of cdk8s following these steps:

```shell

$ yarn bump

```

This will create a new CHANGELOG entry (from conventional commits), bump the version in

`package.json` and create a bump commit.

Now, push to `master` (in the future we will release from a release branch, but in the meantime we release directly from master):

```shell

$ git push origin master

```

This will trigger the [release workflow](./.github/workflows/release.yml) which will release to all package managers and will also create a GitHub release with a tag.

---

Portions of this guide were adopted from the contribution guides of the [AWS

CDK](https://github.com/aws/aws-cdk) and [etcd](https://github.com/etcd-io/etcd).