# Contributing to Nx

We would love for you to contribute to Nx! Read this document to see how to do it.

## How to Get Started Video

Watch this 5-minute video:

<a href="https://www.youtube.com/watch?v=o11p0zSm0No&feature=youtu.be" target="\_blank">

<p align="center"><img src="https://raw.githubusercontent.com/nrwl/nx/master/images/how-to-contribute.png" width="600"></p>

</a>

## Got a Question?

We are trying to keep GitHub issues for bug reports and feature requests. Using the `nrwl` tag on [Stack Overflow](https://stackoverflow.com/questions/tagged/nrwl) is a much better place to ask general questions about how to use Nx.

## 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](https://github.com/nrwl/nx/blob/master/CONTRIBUTING.md#submit-issue) to [our GitHub Repository](https://github.com/nrwl/nx). Even better, you can [submit a Pull Request](https://github.com/nrwl/nx/blob/master/CONTRIBUTING.md#submit-pr) with a fix.

## Project Structure

Source code and documentation are included in the top-level folders listed below.

- `docs` - Markdown and configuration files for documentation including tutorials, guides for each supported platform, and API docs.

- `e2e` - E2E tests.

- `packages` - Source code for Nx packages such as Angular, React, Web, NestJS, Next and others including schematics and builders.

- `scripts` - Miscellaneous scripts for project tasks such as building documentation, testing, and code formatting.

- `tmp` - Folder used by e2e tests. If you are a WebStorm user, make sure to mark this folder as excluded.

## Building the Project

After cloning the project to your machine, to install the dependencies, run:

```bash

yarn

```

To build all the packages, run:

```bash

yarn build

```

## Publishing to a local registry

To test if your changes will actually work once the changes are published,

it can be useful to publish to a local registry.

Check out [this video for a live walkthrough](https://youtu.be/Tx257WpNsxc) or follow the instructions below:

```bash

# Starts the local registry. Keep this running in a separate terminal.

yarn local-registry start

# Set npm and yarn to use the local registry.

# Note: This reroutes your installs to your local registry

yarn local-registry enable

# Revert npm and yarn to use their default registries

yarn local-registry disable

```

To publish packages to a local registry, do the following:

- Run `yarn local-registry start` in Terminal 1 (keep it running)

- Run `yarn local-registry enable` in Terminal 2

- Run `yarn nx-release 999.9.9 --local` in Terminal 2

- Run `cd /tmp` in Terminal 2

- Run `npx create-nx-workspace@999.9.9` in Terminal 2

### Running Unit Tests

To make sure your changes do not break any unit tests, run the following:

```bash

yarn test

```

For example, if you need to only run the tests for the jest package, run:

```bash

nx test jest

```

### Running E2E Tests

To make sure your changes do not break any E2E tests, run:

```bash

yarn e2e

```

Running E2E tests can take some time, so it is often useful to run a single test. To run a single suite of tests, run:

```bash

yarn e2e e2e-cli

```

### Developing on Windows

To build Nx on Windows, you need to use WSL.

- Run `yarn install` in WSL. Yarn will compile several dependencies. If you don't run `install` in WSL, they will be compiled for Windows.

- Run `yarn test` and other commands in WSL.

## Submission Guidelines

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

Before you submit an issue, please search the issue tracker. An issue for your problem may already exist and has been resolved, or 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 and forth with you requiring additional information, such as:

- the output of `nx report`

- `yarn.lock` or `package-lock.json`

- and most importantly - a use-case that fails

A minimal reproduction 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 reproduction 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.

You can file new issues by filling out our [issue form](https://github.com/nrwl/nx/issues/new).

### <a name="submit-pr"></a> Submitting a PR

Please follow the following guidelines:

- Make sure unit tests pass (`yarn test`)

- Target a specific unit test (i.e. `/build/packages/angular/src/utils/ast-utils.spec.js`) with `yarn test angular/src/utils/ast-utils`

- Debug with `node --inspect-brk ./node\_modules/jest/bin/jest.js build/packages/angular/src/utils/ast-utils.spec.js`

- Make sure e2e tests pass (this can take a while, so you can always let CI check those) (`yarn e2e`)

- Target a specific e2e test (i.e. `/build/e2e/cypress.test.js`) with `yarn e2e cypress`

- Debug with `node --inspect-brk ./node\_modules/jest/bin/jest.js build/e2e/cypress.test.js`

- Make sure you run `yarn format`

- Update documentation with `yarn documentation`. For documentation, check for spelling and grammatical errors.

- Update your commit message to follow the guidelines below (use `yarn commit` to automate compliance)

- `yarn checkcommit` will check to make sure your commit messages are formatted correctly

#### Commit Message Guidelines

The commit message should follow the following format:

```

type(scope): subject

BLANK LINE

body

```

##### Type

The type must be one of the following:

- chore

- feat

- fix

- cleanup

- docs

##### Scope

The scope must be one of the following:

- angular - anything Angular specific

- bazel - anything Bazel specific

- core - anything Nx core specific

- docs - anything related to docs infrastructure

- nextjs - anything Next specific

- nest - anything Nest specific

- node - anything Node specific

- linter - anything Linter specific

- react - anything React specific

- storybook - anything Storybook specific

- testing - anything testing specific (e.g., jest or cypress)

- repo - anything related to managing the repo itself

- misc - misc stuff

##### Subject and Body

The subject must contain a description of the change, and the body of the message contains any additional details to provide more context about the change.

Including the issue number that the PR relates to also helps with tracking.

#### Example

```

feat(schematics): add an option to generate lazy-loadable modules

`nx generate lib mylib --lazy` provisions the mylib project in tslint.json

Closes #157

```

#### Commitizen

To simplify and automate the process of committing with this format,

\*\*Nx is a [Commitizen](https://github.com/commitizen/cz-cli) friendly repository\*\*, just do `git add` and execute `yarn commit`.