# Contributing to n8n

Great that you are here and you want to contribute to n8n

## Contents

- [Code of Conduct](#code-of-conduct)

- [Directory Structure](#directory-structure)

- [Development Setup](#development-setup)

- [Development Cycle](#development-cycle)

- [Create Custom Nodes](#create-custom-nodes)

- [Create a new node to contribute to n8n](#create-a-new-node-to-contribute-to-n8n)

- [Checklist before submitting a new node](#checklist-before-submitting-a-new-node)

- [Extend Documentation](#extend-documentation)

- [Contributor License Agreement](#contributor-license-agreement)

## Code of Conduct

This project and everyone participating in it are governed by the Code of

Conduct which can be found in the file [CODE\_OF\_CONDUCT.md](CODE\_OF\_CONDUCT.md).

By participating, you are expected to uphold this code. Please report

unacceptable behavior to jan@n8n.io.

## Directory Structure

n8n is split up in different modules which are all in a single mono repository.

The most important directories:

- [/docker/image](/docker/image) - Dockerfiles to create n8n containers

- [/docker/compose](/docker/compose) - Examples Docker Setups

- [/packages](/packages) - The different n8n modules

- [/packages/cli](/packages/cli) - CLI code to run front- & backend

- [/packages/core](/packages/core) - Core code which handles workflow

execution, active webhooks and

workflows

- [/packages/editor-ui](/packages/editor-ui) - Vue frontend workflow editor

- [/packages/node-dev](/packages/node-dev) - CLI to create new n8n-nodes

- [/packages/nodes-base](/packages/nodes-base) - Base n8n nodes

- [/packages/workflow](/packages/workflow) - Workflow code with interfaces which

get used by front- & backend

## Development Setup

If you want to change or extend n8n you have to make sure that all needed

dependencies are installed and the packages get linked correctly. Here a short guide on how that can be done:

### Requirements

#### Build Tools

The packages which n8n uses depend on a few build tools:

Linux:

```

apt-get install -y build-essential python

```

Windows:

```

npm install -g windows-build-tools

```

#### lerna

n8n is split up in different modules which are all in a single mono repository.

To facilitate those modules management, [lerna](https://lerna.js.org) gets

used. It automatically sets up file-links between modules which depend on each

other.

So for the setup to work correctly lerna has to be installed globally like this:

```

npm install -g lerna

```

### Actual n8n setup

> \*\*IMPORTANT\*\*: All the steps bellow have to get executed at least once to get the development setup up and running!

Now that everything n8n requires to run is installed the actual n8n code can be

checked out and set up:

1. Clone the repository

```

git clone https://github.com/n8n-io/n8n.git

```

1. Go into repository folder

```

cd n8n

```

1. Install all dependencies of all modules and link them together:

```

lerna bootstrap --hoist

```

1. Build all the code:

```

npm run build

```

### Start

To start n8n execute:

```

npm run start

```

## Development Cycle

While iterating on n8n modules code, you can run `npm run dev`. It will then

automatically build your code, restart the backend and refresh the frontend

(editor-ui) on every change you make.

1. Start n8n in development mode:

```

npm run dev

```

1. hack, hack, hack

1. Check if everything still runs in production mode

```

npm run build

npm run start

```

1. Create tests

1. Run all tests

```

npm run test

```

1. Commit code and create pull request

### Test suite

The tests can be started via:

```

npm run test

```

If that gets executed in one of the package folders it will only run the tests

of this package. If it gets executed in the n8n-root folder it will run all

tests of all packages.

## Create Custom Nodes

It is very straightforward to create your own nodes for n8n. More information about that can

be found in the documentation of "n8n-node-dev" which is a small CLI which

helps with n8n-node-development.

[To n8n-node-dev](https://github.com/n8n-io/n8n/tree/master/packages/node-dev)

## Create a new node to contribute to n8n

If you want to create a node which should be added to n8n follow these steps:

1. Read the information in the [n8n-node-dev](https://github.com/n8n-io/n8n/tree/master/packages/node-dev) package as it contains a lot of generic information about node development.

1. Create the n8n development setup like described above and start n8n in develoment mode `npm run dev`

1. Create a new folder for the new node. For a service named "Example" the folder would be called: `/packages/nodes-base/nodes/Example`

1. If there is already a similar node, copy the existing one in the new folder and rename it. If none exists yet, create a boilerplate node with [n8n-node-dev](https://github.com/n8n-io/n8n/tree/master/packages/node-dev) and copy that one in the folder.

1. If the node needs credentials because it has to authenticate with an API or similar create new ones. Existing ones can be found in folder `/packages/nodes-base/credentials`. Also there it is the easiest to copy existing similar ones.

1. Add the path to the new node (and optionally credentials) to package.json of `nodes-base`. It already contains a property `n8n` with its own keys `credentials` and `nodes`.

1. Add icon for the node (60x60 PNG)

1. Start n8n. The new node will then be available via the editor UI and can be tested.

When developing n8n must get restarted and the browser reloaded every time parameters of a node change (like new ones added, removed or changed). Only then will the new data be loaded and the node displayed correctly.

If only the code of the node changes (the execute method) than it is not needed as each workflow automatically starts a new process and so will always load the latest code.

## Checklist before submitting a new node

If you'd like to submit a new node, please go through the following checklist. This will help us be quicker to review and merge your PR.

- [ ] Make failing requests to the API to ensure that the errors get displayed correctly (like malformed requests or requests with invalid credentials)

- [ ] Ensure that the default values do not change and that the parameters do not get renamed, as it would break the existing workflows of the users

- [ ] Ensure that all the top-level parameters use camelCase

- [ ] Ensure that all the options are ordered alphabetically, unless a different order is needed for a specific reason

- [ ] Ensure that the parameters have the correct type

- [ ] Make sure that the file-name and the Class name are identical (case sensitive). The name under "description" in the node-code should also be identical (except that it starts with a lower-case letter and that it will never have a space)

- [ ] Names of Trigger-Nodes always have to end with "Trigger"

- [ ] Add credentials and nodes to the `package.json` file in alphanumerical order

- [ ] Use tabs in all the files except in the `package.json` file, where 4-spaces have to get used

- [ ] To make it as simple as possible for the users, check other similar nodes to ensure that they all behave similarly

- [ ] Try to add as few parameters as possible on the main level to ensure that the node doesn't appear overwhelming. It should only contain the required parameters. All the other ones should be hidden on lower levels as "Additional Parameters" or "Options"

- [ ] Create only one node per service which can do everything via "Resource" and "Options" and not a separate one for each possible operation.

## Extend Documentation

All the files which get used in the n8n documentation on [https://docs.n8n.io](https://docs.n8n.io)

can be found in the [/docs](https://github.com/n8n-io/n8n/tree/master/docs) folder. So all changes

and additions can directly be made in there

That the markdown docs look pretty we use [docsify](https://docsify.js.org). It is possible to test

locally how it looks like rendered with the following commands:

```bash

# 1. Install docisify

npm i docsify-cli -g

# 2. Go into n8n folder (the same folder which contains this file). For example:

cd /data/n8n

# 3. Start docsificy

docsify serve ./docs

```

## Contributor License Agreement

That we do not have any potential problems later it is sadly necessary to sign a [Contributor License Agreement](CONTRIBUTOR\_LICENSE\_AGREEMENT.md). That can be done literally with the push of a button.

We used the most simple one that exists. It is from [Indie Open Source](https://indieopensource.com/forms/cla) which uses plain English and is literally only a few lines long.

A bot will automatically comment on the pull request once it got opened asking for the agreement to be signed. Before it did not get signed it is sadly not possible to merge it in.