To publish your `napi-rs` package, follow these step-by-step instructions:

### 1. \*\*Create Your Own NPM Scope\*\*

You cannot publish under the `@napi-rs` scope unless you have the proper permissions. So, you'll need to create your own npm scope.

- If you don't have an npm account, sign up at [npmjs.com](https://www.npmjs.com/).

- Follow the instructions to [create a scoped package](https://docs.npmjs.com/creating-and-publishing-scoped-public-packages).

For example, let’s assume you created an npm scope named `@your-scope`.

### 2. \*\*Install NAPI Tools\*\*

If you haven't already, install the `napi` command-line tool to help rename and publish your package:

```bash

npm install -g @napi/cli

```

### 3. \*\*Rename the Package\*\*

After setting up your npm scope, rename the project to match your new scope. For example, assuming your package was originally named `cool`, and your scope is `@jarvis`, you can rename it as follows:

```bash

napi rename

```

You will be prompted to rename fields like `name` and `napi name` in your `package.json`. Here's how to proceed:

- `name`: Enter `@jarvis/cool` (or replace `jarvis` with your chosen scope).

- `napi name`: Leave as `cool` unless you want to change it.

- `repository`: Enter your GitHub repository URL (e.g., `https://github.com/yourname/cool.git`).

- `description`: You can leave this empty or provide a brief description of your package.

Example:

```bash

napi rename

? name: name field in package.json @jarvis/cool

? napi name: cool

? repository: https://github.com/yourname/cool.git

? description: A description of the cool package

```

### 4. \*\*Initialize Git and Push to GitHub\*\*

Now that your package is renamed, initialize the Git repository and push it to GitHub:

```bash

git init

git remote add origin git@github.com:yourname/cool.git

git add .

git commit -m "Init"

git push -u origin master

```

Make sure you replace `yourname` with your GitHub username and `cool` with your repository name.

### 5. \*\*Set Up GitHub Actions for Compilation\*\*

If you're using `napi-rs`, you need GitHub Actions to build and publish your package.

#### Set up `NPM\_TOKEN` in GitHub:

- Go to your repository on GitHub.

- Navigate to \*\*Settings\*\* > \*\*Secrets and Variables\*\* > \*\*Actions\*\*.

- Add a new secret named `NPM\_TOKEN` with your npm token (find it in your npm account settings under "Access Tokens").

### 6. \*\*Check the CI Matrix\*\*

If you've set everything up correctly, GitHub Actions will automatically detect your package and build it using the CI matrix (via `napi-rs` GitHub Actions setup).

- Create a `.github/workflows/ci.yml` file to define your CI workflow if you haven't already.

Example workflow:

```yaml

name: CI

on:

push:

tags:

- '\*'

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

- name: Install Node.js

uses: actions/setup-node@v2

with:

node-version: '16'

- run: npm install

- run: npm run build

```

### 7. \*\*Publish Your Package\*\*

Finally, you can publish your package:

1. Bump the version using npm:

```bash

npm version patch

```

2. Push the changes and tags:

```bash

git push --follow-tags

```

This will trigger your CI workflow, build the package, and automatically publish it to npm under your new scope.

<https://chatgpt.com/share/66fb8742-9908-800f-899d-f4e61fc9c75e>

### 8. \*\*Verify the Package on NPM\*\*

Once the CI matrix finishes successfully, your package should be available on npm under the new scope (e.g., `@jarvis/cool`).

npm login | npm publish –access public | <https://napi.rs/docs/introduction/simple-package.en#create-napi-rscool>

You can verify it by visiting [npmjs.com](https://www.npmjs.com/) and searching for your package.