# Quickstart for GitHub Packages

**In this article**



Publish to GitHub Packages with GitHub Actions.

> **Note:** GitHub-hosted runners are not currently supported on GitHub Enterprise Server. You can see more information about planned future support on the GitHub public roadmap.

## Introduction

In this guide, you'll create a GitHub Actions workflow to test your code and then publish it to GitHub Packages.

## Publishing your package

1. Create a new repository on GitHub, adding the `.gitignore` for Node. For more information, see "Creating a new repository."
2. Clone the repository to your local machine.

```
git clone https://YOUR-HOSTNAME/YOUR-USERNAME/YOUR-REPOSITORY.git
cd YOUR-REPOSITORY
```

3. Create an `index.js` file and add a basic alert to say "Hello world!"

JavaScript

```
console.log("Hello, World!");
```

4. Initialize an npm package with `npm init`. In the package initialization wizard, enter your package with the name: *`@YOUR-USERNAME/YOUR-REPOSITORY`*, and set the test script to `exit 0`. This will generate a `package.json` file with information about your package.

```
$ npm init
  ...
  package name: @YOUR-USERNAME/YOUR-REPOSITORY
  ...
  test command: exit 0
  ...
```

5. Run `npm install` to generate the `package-lock.json` file, then commit and push your changes to GitHub.

```
npm install
git add index.js package.json package-lock.json
git commit -m "initialize npm package"
git push
```

6. Create a `.github/workflows` directory. In that directory, create a file named `release-package.yml`.

7. Copy the following YAML content into the `release-package.yml` file, replacing `YOUR-HOSTNAME` with the name of your enterprise.

YAML

```yaml
name: Node.js Package

on:
  release:
    types: [created]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v3
        with:
          node-version: 16
      - run: npm ci
      - run: npm test

  publish-gpr:
    needs: build
    runs-on: ubuntu-latest
    permissions:
      packages: write
      contents: read
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v3
        with:
          node-version: 16
          registry-url: https://npm.YOUR-HOSTNAME.com/
      - run: npm ci
      - run: npm publish
        env:
          NODE_AUTH_TOKEN: ${{secrets.GITHUB_TOKEN}}
```

8. Tell npm which scope and registry to publish packages to using one of the following methods:

   - Add an npm configuration file for the repository by creating a `.npmrc` file in the root directory with the contents:

     ```
     @YOUR-USERNAME:registry=https://npm.pkg.github.com
     ```

   - Edit the `package.json` file and specify the `publishConfig` key:

```
"publishConfig": {
  "@YOUR-USERNAME:registry": "https://npm.pkg.github.com"
}
```

9. Commit and push your changes to GitHub.

```
$ git add .github/workflows/release-package.yml
# Also add the file you created or edited in the previous step.
$ git add .npmrc or package.json
$ git commit -m "workflow to publish package"
$ git push
```

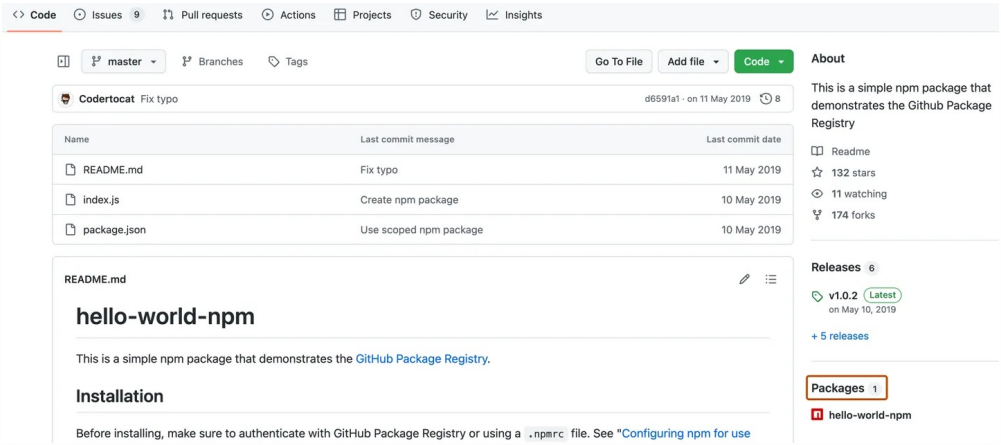
10. The workflow that you created will run whenever a new release is created in your repository. If the tests pass, then the package will be published to GitHub Packages.

    To test this out, navigate to the **Code** tab in your repository and create a new release. For more information, see "Managing releases in a repository."

## Viewing your published package

You can view all of the packages you have published.

1. On your GitHub Enterprise Server instance, navigate to the main page of the repository.
2. In the right sidebar of your repository, click **Packages**.




3. Search for and then click the name of the package that you want to view.

## Installing a published package

Now that you've published the package, you'll want to use it as a dependency across your projects. For more information, see "Working with the npm registry."

## Next steps

The basic workflow you just added runs any time a new release is created in your repository. But this is only the beginning of what you can do with GitHub Packages. You can publish your package to multiple registries with a single workflow, trigger the

workflow to run on different events such as a merged pull request, manage containers, and more.

Combining GitHub Packages and GitHub Actions can help you automate nearly every aspect of your application development processes. Ready to get started? Here are some helpful resources for taking your next steps with GitHub Packages and GitHub Actions:

- "Learn GitHub Packages" for an in-depth tutorial on GitHub Packages
- "Learn GitHub Actions" for an in-depth tutorial on GitHub Actions
- "Working with a GitHub Packages registry" for specific uses cases and examples