This version of GitHub Enterprise was discontinued on 2023-03-15. No patch releases will be made, even for critical security issues. For better performance, improved security, and new features, upgrade to the latest version of GitHub Enterprise. For help with the upgrade, contact GitHub Enterprise support.

Publishing Node.js packages

In this article

You can publish Node.js packages to a registry as part of your continuous integration (CI) workflow.

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

This guide shows you how to create a workflow that publishes Node.js packages to the GitHub Packages and npm registries after continuous integration (CI) tests pass.

Prerequisites

We recommend that you have a basic understanding of workflow configuration options and how to create a workflow file. For more information, see "Learn GitHub Actions."

For more information about creating a CI workflow for your Node.js project, see "Building and testing Node.js."

You may also find it helpful to have a basic understanding of the following:

- "Working with the npm registry"
- "Variables"
- "Encrypted secrets"
- "Automatic token authentication"

About package configuration

The `name` and `version` fields in the *package.json* file create a unique identifier that registries use to link your package to a registry. You can add a summary for the package listing page by including a `description` field in the *package.json* file. For more

information, see "Creating a package.json file" and "Creating Node.js modules" in the npm documentation.

When a local *.npmrc* file exists and has a `registry` value specified, the `npm publish` command uses the registry configured in the *.npmrc* file. You can use the `setup-node` action to create a local *.npmrc* file on the runner that configures the default registry and scope. The `setup-node` action also accepts an authentication token as input, used to access private registries or publish node packages. For more information, see `setup-node`.

You can specify the Node.js version installed on the runner using the `setup-node` action.

If you add steps in your workflow to configure the `publishConfig` fields in your *package.json* file, you don't need to specify the registry-url using the `setup-node` action, but you will be limited to publishing the package to one registry. For more information, see "publishConfig" in the npm documentation.

Publishing packages to the npm registry

You can trigger a workflow to publish your package every time you publish a new release. The process in the following example is executed when the release event of type `published` is triggered. If the CI tests pass, the process uploads the package to the npm registry. For more information, see "Managing releases in a repository."

To perform authenticated operations against the npm registry in your workflow, you'll need to store your npm authentication token as a secret. For example, create a repository secret called `NPM_TOKEN`. For more information, see "Encrypted secrets."

By default, npm uses the `name` field of the *package.json* file to determine the name of your published package. When publishing to a global namespace, you only need to include the package name. For example, you would publish a package named `my-package` to `https://www.npmjs.com/package/my-package`.

If you're publishing a package that includes a scope prefix, include the scope in the name of your *package.json* file. For example, if your npm scope prefix is "octocat" and the package name is "hello-world", the `name` in your *package.json* file should be `@octocat/hello-world`. If your npm package uses a scope prefix and the package is public, you need to use the option `npm publish --access public`. This is an option that npm requires to prevent someone from publishing a private package unintentionally.

This example stores the `NPM_TOKEN` secret in the `NODE_AUTH_TOKEN` environment variable. When the `setup-node` action creates an *.npmrc* file, it references the token from the `NODE_AUTH_TOKEN` environment variable.

YAML

```yaml
name: Publish Package to npmjs
on:
  release:
    types: [published]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      # Setup .npmrc file to publish to npm
      - uses: actions/setup-node@v2
        with:
          node-version: '16.x'
          registry-url: 'https://registry.npmjs.org'
      - run: npm ci
      - run: npm publish
        env:
```

```
          NODE_AUTH_TOKEN: ${{ secrets.NPM_TOKEN }}
```

In the example above, the `setup-node` action creates an *.npmrc* file on the runner with the following contents:

```
//registry.npmjs.org/:_authToken=${NODE_AUTH_TOKEN}
registry=https://registry.npmjs.org/
always-auth=true
```

Please note that you need to set the `registry-url` to `https://registry.npmjs.org/` in `setup-node` to properly configure your credentials.

Publishing packages to GitHub Packages

You can trigger a workflow to publish your package every time you publish a new release. The process in the following example is executed when the release event of type `published` is triggered. If the CI tests pass, the process uploads the package to GitHub Packages. For more information, see "Managing releases in a repository."

Configuring the destination repository

Linking your package to GitHub Packages using the `repository` key is optional. If you choose not to provide the `repository` key in your *package.json* file, then GitHub Packages publishes a package in the GitHub repository you specify in the `name` field of the *package.json* file. For example, a package named `@my-org/test` is published to the `my-org/test` GitHub repository. If the `url` specified in the `repository` key is invalid, your package may still be published however it won't be linked to the repository source as intended.

If you do provide the `repository` key in your *package.json* file, then the repository in that key is used as the destination npm registry for GitHub Packages. For example, publishing the below *package.json* results in a package named `my-package` published to the `octocat/my-other-repo` GitHub repository. Once published, only the repository source is updated, and the package doesn't inherit any permissions from the destination repository.

```
{
  "name": "@octocat/my-package",
  "repository": {
    "type": "git",
    "url": "https://github.com/octocat/my-other-repo.git"
  },
```

Authenticating to the destination repository

To perform authenticated operations against the GitHub Packages registry in your workflow, you can use the `GITHUB_TOKEN`. The `GITHUB_TOKEN` secret is set to an access token for the repository each time a job in a workflow begins. You should set the permissions for this access token in the workflow file to grant read access for the `contents` scope and write access for the `packages` scope. For more information, see "Automatic token authentication."

If you want to publish your package to a different repository, you must use a personal access token that has permission to write to packages in the destination repository. For more information, see "Managing your personal access tokens" and "Encrypted secrets."

Example workflow

This example stores the `GITHUB_TOKEN` secret in the `NODE_AUTH_TOKEN` environment variable. When the `setup-node` action creates an *.npmrc* file, it references the token from the `NODE_AUTH_TOKEN` environment variable.

YAML

```yaml
name: Publish package to GitHub Packages
on:
  release:
    types: [published]
jobs:
  build:
    runs-on: ubuntu-latest
    permissions:
      contents: read
      packages: write
    steps:
      - uses: actions/checkout@v2
      # Setup .npmrc file to publish to GitHub Packages
      - uses: actions/setup-node@v2
        with:
          node-version: '16.x'
          registry-url: 'https://npm.pkg.github.com'
          # Defaults to the user or organization that owns the workflow file
          scope: '@octocat'
      - run: npm ci
      - run: npm publish
        env:
          NODE_AUTH_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

The `setup-node` action creates an *.npmrc* file on the runner. When you use the `scope` input to the `setup-node` action, the *.npmrc* file includes the scope prefix. By default, the `setup-node` action sets the scope in the *.npmrc* file to the account that contains that workflow file.

```
//npm.pkg.github.com/:_authToken=${NODE_AUTH_TOKEN}
@octocat:registry=https://npm.pkg.github.com
always-auth=true
```

Publishing packages using yarn

If you use the Yarn package manager, you can install and publish packages using Yarn.

YAML

```yaml
name: Publish Package to npmjs
on:
  release:
    types: [published]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      # Setup .npmrc file to publish to npm
      - uses: actions/setup-node@v2
        with:
          node-version: '16.x'
          registry-url: 'https://registry.npmjs.org'
          # Defaults to the user or organization that owns the workflow file
          scope: '@octocat'
      - run: yarn
      - run: yarn npm publish // for Yarn version 1, use `yarn publish` instead
```

```
        env:
          NODE_AUTH_TOKEN: ${{ secrets.NPM_TOKEN }}
```

Legal

© 2023 GitHub, Inc. Terms Privacy Status Pricing Expert services Blog