

---

## setup-node



This action provides the following functionality for GitHub Actions users:

- Optionally downloading and caching distribution of the requested Node.js version, and adding it to the PATH
- Optionally caching npm/yarn/pnpm dependencies
- Registering problem matchers for error output
- Configuring authentication for GPR or npm

## Usage

See action.yml

```
1 - uses: actions/setup-node@v4
2   with:
3     # Version Spec of the version to use in SemVer notation.
4     # It also emits such aliases as lts, latest, nightly and canary
5     # builds
6     # Examples: 12.x, 10.15.1, >=10.15.0, lts/Hydrogen, 16-nightly,
7     # latest, node
8     node-version: ''
9     # File containing the version Spec of the version to use. Examples
10    # : package.json, .nvmrc, .node-version, .tool-versions.
11    # If node-version and node-version-file are both provided the
12    # action will use version from node-version.
13    node-version-file: ''
14    # Set this option if you want the action to check for the latest
15    # available version
16    # that satisfies the version spec.
17    # It will only get affect for lts Nodejs versions (12.x, >=10.15.0,
18    # lts/Hydrogen).
19    # Default: false
20    check-latest: false
21    # Target architecture for Node to use. Examples: x86, x64. Will use
22    # system architecture by default.
23    # Default: ''. The action use system architecture by default
24    architecture: ''
25    # Used to pull node distributions from https://github.com/actions/
26    # node-versions.
```

---

```
23 # Since there's a default, this is typically not supplied by the
24 # user.
25 # When running this action on github.com, the default value is
26 # sufficient.
27 # When running on GHES, you can pass a personal access token for
28 # github.com if you are experiencing rate limiting.
29 #
30 # We recommend using a service account with the least permissions
31 # necessary. Also
32 # when generating a new PAT, select the least scopes necessary.
33 #
34 # [Learn more about creating and using encrypted secrets](https://
35 # help.github.com/en/actions/automating-your-workflow-with-github-
36 # actions/creating-and-using-encrypted-secrets)
37 #
38 # Default: ${github.server_url == 'https://github.com' && github.
39 # token || ''}
40 token: ''
41
42 # Used to specify a package manager for caching in the default
43 # directory. Supported values: npm, yarn, pnpm.
44 # Package manager should be pre-installed
45 # Default: ''
46 cache: ''
47
48 # Used to specify the path to a dependency file: package-lock.json,
49 # yarn.lock, etc.
50 # It will generate hash from the target file for primary key. It
51 # works only If cache is specified.
52 # Supports wildcards or a list of file names for caching multiple
53 # dependencies.
54 # Default: ''
55 cache-dependency-path: ''
56
57 # Optional registry to set up for auth. Will set the registry in a
58 # project level .npmrc and .yarnrc file,
59 # and set up auth to read in from env.NODE_AUTH_TOKEN.
60 # Default: ''
61 registry-url: ''
62
63 # Optional scope for authenticating against scoped registries.
64 # Will fall back to the repository owner when using the GitHub
65 # Packages registry (https://npm.pkg.github.com/).
66 # Default: ''
67 scope: ''
68
69 # Set always-auth option in npmrc file.
70 # Default: ''
71 always-auth: ''
```

## Basic:

---

```
1 steps:
2 - uses: actions/checkout@v4
3 - uses: actions/setup-node@v4
4   with:
5     node-version: 18
6 - run: npm ci
7 - run: npm test
```

The `node-version` input is optional. If not supplied, the node version from PATH will be used. However, it is recommended to always specify Node.js version and don't rely on the system one.

The action will first check the local cache for a semver match. If unable to find a specific version in the cache, the action will attempt to download a version of Node.js. It will pull LTS versions from node-versions releases and on miss or failure will fall back to the previous behavior of downloading directly from node dist.

For information regarding locally cached versions of Node.js on GitHub hosted runners, check out GitHub Actions Runner Images.

### Supported version syntax

The `node-version` input supports the Semantic Versioning Specification, for more detailed examples please refer to the semver package documentation.

Examples:

- Major versions: 18, 20
- More specific versions: 10.15, 16.15.1, 18.4.0
- NVM LTS syntax: `lts/erbium`, `lts/fermium`, `lts/*`, `lts/-n`
- Latest release: `*` or `latest/current/node`

**Note:** Like the other values, `*` will get the latest locally-cached Node.js version, or the latest version from actions/node-versions, depending on the `check-latest` input.

`current/latest/node` always resolve to the latest dist version. That version is then downloaded from actions/node-versions if possible, or directly from Node.js if not. Since it will not be cached always, there is possibility of hitting rate limit when downloading from dist

### Checking in lockfiles

It's **always** recommended to commit the lockfile of your package manager for security and performance reasons. For more information consult the "Working with lockfiles" section of the Advanced usage guide.

---

## Caching global packages data

The action has a built-in functionality for caching and restoring dependencies. It uses actions/cache under the hood for caching global packages data but requires less configuration settings. Supported package managers are `npm`, `yarn`, `pnpm` (v6.10+). The `cache` input is optional, and caching is turned off by default.

The action defaults to search for the dependency file (`package-lock.json`, `npm-shrinkwrap.json` or `yarn.lock`) in the repository root, and uses its hash as a part of the cache key. Use `cache-dependency-path` for cases when multiple dependency files are used, or they are located in different subdirectories.

**Note:** The action does not cache `node_modules`

See the examples of using cache for `yarn/pnpm` and `cache-dependency-path` input in the Advanced usage guide.

### Caching npm dependencies:

```
1 steps:
2   - uses: actions/checkout@v4
3   - uses: actions/setup-node@v4
4     with:
5       node-version: 20
6       cache: 'npm'
7   - run: npm ci
8   - run: npm test
```

### Caching npm dependencies in monorepos:

```
1 steps:
2   - uses: actions/checkout@v4
3   - uses: actions/setup-node@v4
4     with:
5       node-version: 20
6       cache: 'npm'
7       cache-dependency-path: subdir/package-lock.json
8   - run: npm ci
9   - run: npm test
```

## Matrix Testing

```
1 jobs:
2   build:
3     runs-on: ubuntu-latest
4     strategy:
5       matrix:
```

---

```
6     node: [ 14, 16, 18 ]
7     name: Node ${{ matrix.node }} sample
8     steps:
9       - uses: actions/checkout@v4
10      - name: Setup node
11        uses: actions/setup-node@v4
12        with:
13          node-version: ${{ matrix.node }}
14      - run: npm ci
15      - run: npm test
```

## Using setup-node on GHES

`setup-node` comes pre-installed on the appliance with GHES if Actions is enabled. When dynamically downloading Nodejs distributions, `setup-node` downloads distributions from `actions/node-versions` on github.com (outside of the appliance). These calls to `actions/node-versions` are made via unauthenticated requests, which are limited to 60 requests per hour per IP. If more requests are made within the time frame, then you will start to see rate-limit errors during downloading that looks like: `##[error]API rate limit exceeded for...` After that error the action will try to download versions directly from the official site, but it also can have rate limit so it's better to put token.

To get a higher rate limit, you can generate a personal access token on github.com and pass it as the `token` input for the action:

```
1 uses: actions/setup-node@v4
2 with:
3   token: ${{ secrets.GH_DOTCOM_TOKEN }}
4   node-version: 20
```

If the runner is not able to access github.com, any Nodejs versions requested during a workflow run must come from the runner's tool cache. See "Setting up the tool cache on self-hosted runners without internet access" for more information.

## Advanced usage

- Check latest version
- Using a node version file
- Using different architectures
- Using v8 canary versions
- Using nightly versions
- Using rc versions

- 
- Caching packages data
  - Using multiple operating systems and architectures
  - Publishing to npmjs and GPR with npm
  - Publishing to npmjs and GPR with yarn
  - Using private packages

## **License**

The scripts and documentation in this project are released under the MIT License

## **Contributions**

Contributions are welcome! See Contributor's Guide

## **Code of Conduct**

:wave: Be nice. See our code of conduct