



Linux Ubuntu 26.04 Manual Pages on command 'npm-version.1'

\$ man npm-version.1

NPM-VERSION(1) General Commands Manual NPM-VERSION(1)

NAME

npm-version

Synopsis

<!-- AUTOGENERATED USAGE DESCRIPTIONS -->

Configuration

<!-- AUTOGENERATED CONFIG DESCRIPTIONS -->

Description

Run this in a package directory to bump the version and write the new data back to package.json, package-lock.json, and, if present, npm-shrinkwrap.json.

The newversion argument should be a valid semver string, a valid second argument to semver.inc (one of patch, minor, major, prepatch, preminor, premajor, prerelease), or from-git. In the second case, the existing version will be incremented by 1 in the specified field. from-git will try to read the latest git tag, and use that as the new npm version.

If run in a git repo, it will also create a version commit and tag. This behavior is controlled by git-tag-version (see below), and can be disabled on the command line by running npm --no-git-tag-version version.

It will fail if the working directory is not clean, unless the -f or --force flag is set.

If supplied with -m or --message config option,

npm will use it as a commit message when creating a version commit. If the message config contains %s then that will be replaced with the resulting version number. For example:

```
npm version patch -m "Upgrade to %s for reasons"
```

If the sign-git-tag config is set, then the

tag will be signed using the -s flag to git. Note that you must have a default GPG key set up in your git config for this to work properly. For example:

```
$ npm config set sign-git-tag true
```

```
$ npm version patch
```

You need a passphrase to unlock the secret key for

user: "isaacs (<http://blog.izs.me/>) <i@izs.me>"

2048-bit RSA key, ID 6C481CF6, created 2010-08-31

Enter passphrase:

If preversion, version, or postversion are in the scripts property

of the package.json, they will be executed as part of running npm version.

The exact order of execution is as follows:

? Check to make sure the git working directory is clean before we get started. Your scripts may add files to the commit in future steps.

This step is skipped if the --force flag is set.

? Run the preversion script. These scripts have access to the old version in package.json. A typical use would be running your full test suite before deploying. Any files you want added to the commit should be explicitly added using git add.

? Bump version in package.json as requested (patch, minor, major, etc).

? Run the version script. These scripts have access to the new version in package.json (so they can incorporate it into file headers in generated files for example). Again, scripts should explicitly add generated files to the commit using git add.

? Commit and tag.

? Run the postversion script. Use it to clean up the file system or automatically push the commit and/or tag.

Take the following example:

```
{  
  "scripts": {  
    "preversion": "npm test",  
    "version": "npm run build && git add -A dist",  
    "postversion": "git push && git push --tags && rm -rf build/temp"  
  }  
}
```

This runs all your tests and proceeds only if they pass. Then runs your build script, and adds everything in the dist directory to the commit.

After the commit, it pushes the new commit and tag up to the server, and deletes the build/temp directory.

See Also

- ? npm init
- ? npm run-script
- ? npm scripts
- ? package.json
- ? config

9.2.0

March 2026

NPM-VERSION(1)