node-gyp

6.1.0 • Public • Published 2 months ago

Readme

Explore BETA

11 Dependencies

777 Dependents

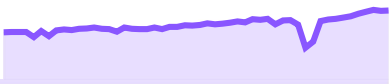
123 Versions

Install

```
> npm i node-gyp
```

Weekly Downloads

5,886,147



Version

6.1.0

License

MIT

Unpacked Size

1.77 MB

Total Files

121

Issues
45

Pull Requests
14

Homepage
github.com/nodejs/node-gyp#readme

Repository
github.com/nodejs/node-gyp

Last publish
a month ago

Collaborators




Try on RunKit

Report a vulnerability

node-gyp - Node.js native addon build tool

`node-gyp` is a cross-platform command-line tool written in Node.js for compiling native addon modules for Node.js. It contains a fork of the **gyp** project that was previously used by the Chromium team, extended to support the development of Node.js native addons.

Note that `node-gyp` is *not* used to build Node.js itself.

Multiple target versions of Node.js are supported (i.e. `0.8`, ..., `4`, `5`, `6`, etc.), regardless of what version of Node.js is actually installed on your system (`node-gyp` downloads the necessary development files or headers for the target version).

Features

- The same build commands work on any of the supported platforms

- Supports the targeting of different versions of Node.js

Installation

You can install `node-gyp` using `npm`:

```
$ npm install -g node-gyp
```

Depending on your operating system, you will need to install:

On Unix

- Python v2.7, v3.5, v3.6, or v3.7
- `make`
- A proper C/C++ compiler toolchain, like GCC

On macOS

- Python v2.7, v3.5, v3.6, or v3.7
- Xcode
  - You also need to install the `XCode Command Line Tools` by running `xcode-select --install`. Alternatively, if you already have the full Xcode installed, you can find them under the menu `Xcode -> Open Developer Tool -> More Developer Tools...`. This step will install `clang`, `clang++`, and `make`.
- If your Mac has been *upgraded* to macOS Catalina (10.15), please read macOS_Catalina.md.

On Windows

Install the current version of Python from the **Microsoft Store package**.

Option 1

Install all the required tools and configurations using Microsoft's **windows-build-tools** using `npm install --global --production windows-build-tools` from an elevated PowerShell or CMD.exe (run as Administrator).

Option 2

Install tools and configuration manually:

- Install Visual C++ Build Environment: Visual Studio Build Tools (using "Visual C++ build tools" workload) or Visual Studio 2017 Community (using the "Desktop development with C++" workload)
- Launch cmd, `npm config set msvs_version 2017`

If the above steps didn't work for you, please visit **Microsoft's Node.js Guidelines for Windows** for additional tips.

To target native ARM64 Node.js on Windows 10 on ARM, add the components "Visual C++ compilers and libraries for ARM64" and "Visual C++ ATL for ARM64".

Configuring Python Dependency

`node-gyp` requires that you have installed a compatible version of Python, one of: v2.7, v3.5, v3.6, or v3.7. If you have multiple Python versions installed, you can identify which Python version `node-gyp` should use in one of the following ways:

1. by setting the `--python` command-line option, e.g.:

```
$ node-gyp <command> --python /path/to/executable/python
```

1. If `node-gyp` is called by way of `npm`, *and* you have multiple versions of Python installed, then you can set `npm`'s 'python' config key to the appropriate value:

```
$ npm config set python /path/to/executable/python
```

1. If the `PYTHON` environment variable is set to the path of a Python executable, then that version will be used, if it is a compatible version.
2. If the `NODE_GYP_FORCE_PYTHON` environment variable is set to the path of a Python executable, it will be used instead of any of the other configured or builtin Python search paths. If it's not a compatible version, no further searching will be done.

How to Use

To compile your native addon, first go to its root directory:

```
$ cd my_node_addon
```

The next step is to generate the appropriate project build files for the current platform. Use `configure` for that:

```
$ node-gyp configure
```

Auto-detection fails for Visual C++ Build Tools 2015, so `--msvs_version=2015` needs to be added (not needed when run by npm as configured above):

```
$ node-gyp configure --msvs_version=2015
```

Note: The `configure` step looks for a `binding.gyp` file in the current directory to process. See below for instructions on creating a `binding.gyp` file.

Now you will have either a `Makefile` (on Unix platforms) or a `vcxproj` file (on Windows) in the `build/` directory. Next, invoke the `build` command:

```
$ node-gyp build
```

Now you have your compiled `.node` bindings file! The compiled bindings end up in `build/Debug/` or `build/Release/`, depending on the build mode. At this point, you can require the `.node` file with Node.js and run your tests!

Note: To create a *Debug* build of the bindings file, pass the `--debug` (or `-d`) switch when running either the `configure`, `build` or `rebuild` commands.

The `binding.gyp` file

A `binding.gyp` file describes the configuration to build your module, in a JSON-like format. This file gets placed in the root of your package, alongside `package.json`.

A barebones `gyp` file appropriate for building a Node.js addon could look like:

```
{
  "targets": [
    {
      "target_name": "binding",
      "sources": [ "src/binding.cc" ]
```

```
      }
    ]
  }
```

Further reading

Some additional resources for Node.js native addons and writing `gyp` configuration files:

- "Going Native" a nodeschool.io tutorial
- "Hello World" node addon example
- gyp user documentation
- gyp input format reference
- *"binding.gyp" files out in the wild* wiki page

Commands

`node-gyp` responds to the following commands:

Command	Description
`help`	Shows the help dialog
`build`	Invokes `make` / `msbuild.exe` and builds the native addon
`clean`	Removes the `build` directory if it exists
`configure`	Generates project build files for the current platform
`rebuild`	Runs `clean`, `configure` and `build` all in a row
`install`	Installs Node.js header files for the given version
`list`	Lists the currently installed Node.js header versions
`remove`	Removes the Node.js header files for the given version

Command Options

`node-gyp` accepts the following command options:

Command	Description
`-j n, --jobs n`	Run `make` in parallel. The value m
`--target=v6.2.1`	Node.js version to build for (defaul
`--silly, --loglevel=silly`	Log all progress to console
`--verbose, --loglevel=verbose`	Log most progress to console
`--silent, --loglevel=silent`	Don't log anything to console
`debug, --debug`	Make Debug build (default is `Rele`
`--release, --no-debug`	Make Release build
`-C $dir, --directory=$dir`	Run command in different director
`--make=$make`	Override `make` command (e.g. gr
`--thin=yes`	Enable thin static libraries
`--arch=$arch`	Set target architecture (e.g. ia32)
`--tarball=$path`	Get headers from a local tarball
`--devdir=$path`	SDK download directory (default is
`--ensure`	Don't reinstall headers if already pr
`--dist-url=$url`	Download header tarball from cust
`--proxy=$url`	Set HTTP(S) proxy for downloading
`--noproxy=$urls`	Set urls to ignore proxies when dov
`--cafile=$cafile`	Override default CA chain (to down
`--nodedir=$path`	Set the path to the node source cod
`--python=$path`	Set path to the Python binary

Command	Description
`--msvs_version=$version`	Set Visual Studio version (Windows
`--solution=$solution`	Set Visual Studio Solution version (

Configuration

Environment variables

Use the form `npm_config_OPTION_NAME` for any of the command options listed above (dashes in option names should be replaced by underscores).

For example, to set `devdir` equal to `/tmp/.gyp`, you would:

Run this on Unix:

```
$ export npm_config_devdir=/tmp/.gyp
```

Or this on Windows:

```
> set npm_config_devdir=c:\temp\.gyp
```

`npm` configuration

Use the form `OPTION_NAME` for any of the command options listed above.

For example, to set `devdir` equal to `/tmp/.gyp`, you would run:

```
$ npm config set [--global] devdir /tmp/.gyp
```

Note: Configuration set via `npm` will only be used when `node-gyp` is run via `npm`, not when `node-gyp` is run directly.

License

`node-gyp` is available under the MIT license. See the **LICENSE file** for details.

Keywords

native addon module c c++ bindings gyp

Help

Documentation

Community

Resources

Advisories

Status

Contact

About

Company

Blog

Careers

Webinars

Press

Newsletter

Terms & Policies

Policies

Terms of Use

Code of Conduct

Privacy