

---

## npm\_lazy

A lazy local cache for npm

### Why?

- npm can be slow, down or return random errors if you have large deploys
- npm\_lazy caches packages on your local network, making things faster and more predictable
- If 100 servers request the same package metadata at the same time, npm\_lazy makes sure that (at most) only one request goes out to the npm registry.
- No database to install, replicate or manage. Data is stored under `./db/` as JSON and tar files.
- Lazy caching: When a package is requested the first time, it is cached locally. No explicit need to manage packages or replication.
- Metadata is expired periodically (default: 1 hour) so that the latest versions of packages are fetched.

Here are all the ways in which npm\_lazy is resilient to registry failures:

- All HTTP requests are retried.
- All HTTP requests are subject to a maximum fetch timeout (default: 5000 ms). If this fails, the request is retried (or failed).
- Invalid responses are rejected and retried:
  - Tarfiles are checked against the expected shasum, and cached forever if they match; if not, they are retried.
  - Metadata files must parse as JSON; if not, they are retried.
- Metadata files are never discarded until a newer version can be fetched successfully. If the JSON metadata is older than `cacheAge` (default: 1 hour), we will attempt to contact the registry first. However, if contacting the registry fails, then the old version of the metadata is sent instead. This means that even when outages occur, you can install any package that has been installed at least once before.

### New in version 1.13.x

- Updated package.json dependencies

---

### **New in version 1.12.x**

- The NPM registry now uses https for all connections, updated all internals and defaults to use https. Thanks @DanielDent!

### **New in version 1.11.x**

- Added support for private npm packages, thanks @scottnonnenberg!

### **New in version 1.10.x**

- Added support for scoped packages.

### **New in version 1.9.x**

- Added port, host, remote-url and external-url CLI command configurations (#47, thanks @albertosouza)

### **New in version 1.8.x**

- Better handling of npm private modules (#52, thanks @CL0SeY)

### **New in version 1.7.x**

- introducing @CL0SeY as a co-maintainer / core contributor, and a solid set of improvements to the error handling in npm\_lazy.
- improved remote error handling (404's, 500's) for resources that are not in the cache (thanks @CL0SeY)
  - 404's are returned immediately (previously, npm\_lazy returned a generic 500 error for 404's).
  - for other errors are requests retried `maxRetries` times. The error response content and error status code are also now returned up from the registry to the npm\_lazy clients.

---

## New in version 1.6.x

- improved etags handling (thanks @CL0SeY)
- use mikeal/request to improve compatibility with Windows-world proxies (thanks @garytaylor)
- fixed an error where hashing a package would incorrectly report a failure (thanks @kleini)
- fixed tests (thanks @CL0SeY) and converted tests to use Mocha BDD style
- 302 support (thanks @guig)
- fixed a broken reference (thanks @kwizzn)

## New in version 1.5.x

Added support for using a http proxy (note: not a Socks5 proxy). This can be configured either via the config file or via the `http_proxy` environment variable, see the config at the end for an example. Thanks @migounette! As I am not using a proxy myself, please report any issues via GH (pull requests welcome!).

Note: if you already have a proxy for npm, make sure you don't run into an issue where npm uses the proxy when accessing npm\_lazy. You don't want to have `npm install -> proxy -> npm_lazy -> proxy`, but rather `npm install -> npm_lazy -> proxy` since your proxy probably doesn't know how to connect to npm\_lazy. You will need to disable npm's internal proxy config, see this comment for the details.

Check out the changelog for version history.

## Installation

*Requires node >= 0.10.x*

v1.1.x adds a command called `npm_lazy` to make things even easier. Install via npm:

```
1 sudo npm install -g npm_lazy
```

To start the server, run:

```
1 npm_lazy
```

To edit the configuration, start by initializing a file from the default config file:

```
1 npm_lazy --init > ~/npm_lazy.config.js
```

To start the server with a custom configuration:

```
1 npm_lazy --config ~/npm_lazy.config.js
```

---

Make sure you also empty out any npm caches by running `npm cache clean`, as npm does its own local caching, which means that some files might still point directly to the registry rather than to the npm\_lazy endpoints.

## Installation by cloning the repo

Or alternatively, if you don't want to install this globally, you can just clone the repo: `git clone git@github.com:mixu/npm_lazy.git` && `cd npm_lazy` && `npm install` and edit `config.js`.

## Pointing npm to npm\_lazy

To temporarily set the registry:

```
1 npm --registry http://localhost:8080/ install socket.io
```

To permanently set the registry via command line:

```
1 npm config set registry http://localhost:8080/
```

To permanently set the registry via config file, in `~/.npmrc`:

```
1 registry = http://localhost:8080/
```

For more info, see “npm help config” and “npm help registry”.

## Tips and tricks:

A few things that might be useful to know:

- Start by running `npm cache clean` so that your local npm command will request every package you want at least once from npm\_lazy.
- Starting with v1.2.0, npm\_lazy will only return a 500 error if it does not have specific file.
- To clear out the npm\_lazy cache, simply remove the cache directory and restart npm\_lazy. npm\_lazy prints out the cache location when it starts, and it defaults to `~/ .npm_lazy`.
- Note that only package index metadata and package tarfiles are cached; all other endpoints are just a transparently proxied (e.g. you can always run `npm install` for cached packages but more exotic npm endpoints will not work if the registry is down; they will simply act like their non-npm\_lazy equivalents).

- 
- Also, note that if you intend to write through `npm_lazy` you must set `cacheAge` to 0 so that npm metadata is always refreshed because npm wants to know that you have the most recent package `_id` before it allows writing. This will still return cached data for package.json indexes needed for installation if the registry is down, but only after attempting to contact the registry (this seems like a decent, but not perfect compromise).
  - Restarting `npm_lazy` will clear the package.json metadata refresh timeout and the max retries counter. All cached entries, including package.json files and tarfiles are kept, so you can safely restart the server to expire the metadata `cacheAge` while retaining all cached artifacts.
  - `npm_lazy` works by rewriting the download URLs for package.json results, so old files from `npm shrinkwrap` may interfere with it since they may contain direct references to `registry.npmjs.com`. Make sure you clean up that stuff.
  - If you are using self-signed certs, set `rejectUnauthorized` to false in the config.

## Resiliency to registry failures

First, install a package successfully so that it is cached.

Next, to simulate a network failure, add `0.0.0.1 registry.npmjs.com` to `/etc/hosts` and try installing that same package again (in another folder). You should see something like this:

```
1 npm_lazy at localhost port 8080
2 npm_lazy cache directory: /home/m/.npm_lazy
3 Fetch failed (1/5): https://registry.npmjs.com/socket.io { [Error:
  connect EINVAL] code: 'EINVAL', errno: 'EINVAL', syscall: 'connect'
}
4 Fetch failed (2/5): https://registry.npmjs.com/socket.io { [Error:
  connect EINVAL] code: 'EINVAL', errno: 'EINVAL', syscall: 'connect'
}
5 Fetch failed (3/5): https://registry.npmjs.com/socket.io { [Error:
  connect EINVAL] code: 'EINVAL', errno: 'EINVAL', syscall: 'connect'
}
6 Fetch failed (4/5): https://registry.npmjs.com/socket.io { [Error:
  connect EINVAL] code: 'EINVAL', errno: 'EINVAL', syscall: 'connect'
}
7 Fetch failed (5/5): https://registry.npmjs.com/socket.io { [Error:
  connect EINVAL] code: 'EINVAL', errno: 'EINVAL', syscall: 'connect'
}
8 [OK] Reusing cached result for https://registry.npmjs.com/socket.io
```

## Configuration

Configured by editing `config.js` in the same directory:

---

```
1 var path = require('path'),
2     homePath = path.normalize(process.env[(process.platform == 'win32')
3       ? 'USERPROFILE' : 'HOME']);
4
5 module.exports = {
6   // Logging config
7   loggingOpts: {
8     // show the ip address of the machine requesting the npm package
9     logRequesterIP: true,
10    // Print to stdout with colors
11    logToConsole: true,
12    // Write to file
13    logToFile: false,
14
15    // This should be a file path.
16    filename: homePath + '/npm_lazy.log'
17  },
18
19  // Cache config
20
21  // `cacheDirectory`: Directory to store cached packages.
22  //
23  // Note: Since any relative path is resolved relative to the current
24  // working
25  // directory when the server is started, you should use a full path.
26  cacheDirectory: homePath + '/.npm_lazy',
27
28  // `cacheAge`: maximum age before an index is refreshed from
29  // remoteUrl
30  // - negative value means no refresh (e.g. once cached, never update
31  // the package.json metadata)
32  // - zero means always refresh (e.g. always ask the registry for
33  // metadata)
34  // - positive value means refresh every n milliseconds
35  // (e.g. 60 * 60 * 1000 = expire metadata every 60 minutes)
36  //
37  // Note: if you want to use `npm star` and other methods which update
38  // npm metadata, you will need to set cacheAge to 0. npm generally
39  // wants the latest
40  // package metadata version so caching package metadata will
41  // interfere with it.
42  // Recommended setting: 0
43  cacheAge: 0,
44
45  // Request config
46
47  // max milliseconds to wait for each HTTP response
```

---

```
44 httpTimeout: 10000,
45 // maximum number of retries per HTTP resource to get
46 maxRetries: 5,
47 // whether or not HTTPS requests are checked against Node's list of
   CAs
48 // set false if you are using your own npm mirror with a self-signed
   SSL cert
49 rejectUnauthorized: true,
50
51 // Remote and local URL
52
53 // external url to npm_lazy, no trailing /
54 externalUrl: 'http://localhost:8080',
55 // registry url with trailing /
56 remoteUrl: 'https://registry.npmjs.com/',
57 // bind port and host
58 port: 8080,
59 host: '0.0.0.0',
60
61 // Proxy config
62 // You can also configure this using the http_proxy and https_proxy
   environment variables
63 // cf. https://wiki.archlinux.org/index.php/proxy_settings
64 proxy: {
65   // http: 'http://1.2.3.4:80/',
66   // https: 'http://4.3.2.1:80/'
67 }
68 };
```

## Caching logic

When a resource is requested:

- Anything that we don't have locally gets fetched from registry.npmjs.com on demand.
- Metadata is updated when the resource is requested the first time after a restart, and if the resource is requested later than the max age set in configuration (which is currently set to 0. Set the max age for package metadata in the config.js file to override this).