Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'npm-ci.1' command

$ man npm-ci.1

NPM-CI(1) NPM-CI(1)

NAME

npm-ci - Clean install a project

Synopsis

npm ci

aliases: clean-install, ic, install-clean, isntall-clean

Description

This command is similar to npm help install, except it's meant to be used in automated environments such as test platforms, continuous inte? gration, and deployment -- or any situation where you want to make sure you're doing a clean install of your dependencies.

The main differences between using npm install and npm ci are:

? The project must have an existing package-lock.json or npm-shrinkwrap.json.

? If dependencies in the package lock do not match those in pack? age.json, npm ci will exit with an error, instead of updating the package lock.

? npm ci can only install entire projects at a time: individual depen? dencies cannot be added with this command.

? If a node_modules is already present, it will be automatically re? moved before npm ci begins its install.

? It will never write to package.json or any of the package-locks: in? stalls are essentially frozen.

NOTE: If you create your package-lock.json file by running npm install with flags that can affect the shape of your dependency tree, such as --legacy-peer-deps or --install-links, you must provide the same flags to npm ci or you are likely to encounter errors. An easy way to do this is to run, for example, npm config set legacy-peer-deps=true --loca? tion=project and commit the .npmrc file to your repo.

Example

Make sure you have a package-lock and an up-to-date install:

```
$ cd ./my/npm/project
$ npm install
added 154 packages in 10s
$ ls | grep package-lock
```

Run npm ci in that project

```
$ npm ci
added 154 packages in 5s
```

Configure Travis CI to build using npm ci instead of npm install:

```
# .travis.yml
install:
- npm ci
# keep the npm cache around to speed up installs
cache:
  directories:
  - "$HOME/.npm"
```

Configuration

save

? Default: true unless when using npm update where it defaults to false

? Type: Boolean

Save installed packages to a package.json file as dependencies.

When used with the npm rm command, removes the dependency from pack? age.json.

Will also prevent writing to package-lock.json if set to false.

save-exact

? Default: false

? Type: Boolean

Dependencies saved to package.json will be configured with an exact version rather than using npm's default semver range operator.

global

? Default: false

? Type: Boolean

Operates in "global" mode, so that packages are installed into the pre?fix folder instead of the current working directory. See npm help fold?ers for more on the differences in behavior.

? packages are installed into the {prefix}/lib/node_modules folder, in?stead of the current working directory.

? bin files are linked to {prefix}/bin

? man pages are linked to {prefix}/share/man

global-style

? Default: false

? Type: Boolean

Causes npm to install the package into your local node_modules folder with the same layout it uses with the global node_modules folder. Only your direct dependencies will show in node_modules and everything they depend on will be flattened in their node_modules folders. This obvi?ously will eliminate some deduping. If used with legacy-bundling, legacy-bundling will be preferred.

legacy-bundling

? Default: false

? Type: Boolean

Causes npm to install the package such that versions of npm prior to 1.4, such as the one included with node 0.8, can install the package. This eliminates all automatic deduping. If used with global-style this option will be preferred.

omit

? Default: 'dev' if the NODE_ENV environment variable is set to 'pro?duction', otherwise empty.

? Type: "dev", "optional", or "peer" (can be set multiple times)

Dependency types to omit from the installation tree on disk.

Note that these dependencies are still resolved and added to the pack?

age-lock.json or npm-shrinkwrap.json file. They are just not physically

installed on disk.

If a package type appears in both the --include and --omit lists, then

it will be included.

If the resulting omit list includes 'dev', then the NODE_ENV environ?

ment variable will be set to 'production' for all lifecycle scripts.

strict-peer-deps

? Default: false

? Type: Boolean

If set to true, and --legacy-peer-deps is not set, then any conflicting

peerDependencies will be treated as an install failure, even if npm

could reasonably guess the appropriate resolution based on non-peer de?

pendency relationships.

By default, conflicting peerDependencies deep in the dependency graph

will be resolved using the nearest non-peer dependency specification,

even if doing so will result in some packages receiving a peer depen?

dency outside the range set in their package's peerDependencies object.

When such and override is performed, a warning is printed, explaining

the conflict and the packages involved. If --strict-peer-deps is set,

then this warning is treated as a failure.

package-lock

? Default: true

? Type: Boolean

If set to false, then ignore package-lock.json files when installing.

This will also prevent writing package-lock.json if save is true.

This configuration does not affect npm ci.

foreground-scripts

? Default: false

? Type: Boolean

Run all build scripts (ie, preinstall, install, and postinstall)

scripts for installed packages in the foreground process, sharing stan?

dard input, output, and error with the main npm process.

Note that this will generally make installs run slower, and be much noisier, but can be useful for debugging.

ignore-scripts

? Default: false

? Type: Boolean

If true, npm does not run scripts specified in package.json files.

Note that commands explicitly intended to run a particular script, such as npm start, npm stop, npm restart, npm test, and npm run-script will still run their intended script if ignore-scripts is set, but they will not run any pre- or post-scripts.

audit

? Default: true

? Type: Boolean

When "true" submit audit reports alongside the current npm command to the default registry and all registries configured for scopes. See the documentation for npm help audit for details on what is submitted.

bin-links

? Default: true

? Type: Boolean

Tells npm to create symlinks (or .cmd shims on Windows) for package ex?ecutables.

Set to false to have it not do this. This can be used to work around the fact that some file systems don't support symlinks, even on osten?sibly Unix systems.

fund

? Default: true

? Type: Boolean

When "true" displays the message at the end of each npm install ac?knowledging the number of dependencies looking for funding. See npm help fund for details.

dry-run

? Default: false

? Type: Boolean

Indicates that you don't want npm to make any changes and that it should only report what it would have done. This can be passed into any of the commands that modify your local installation, eg, install, up? date, dedupe, uninstall, as well as pack and publish.

Note: This is NOT honored by other network related commands, eg dist-tags, owner, etc.

workspace

? Default:

? Type: String (can be set multiple times)

Enable running a command in the context of the configured workspaces of the current project while filtering by running only the workspaces de? fined by this configuration option.

Valid values for the workspace config are either:

? Workspace names

? Path to a workspace directory

? Path to a parent workspace directory (will result in selecting all workspaces within that folder)

When set for the npm init command, this may be set to the folder of a workspace which does not yet exist, to create the folder and set it up as a brand new workspace within the project.

This value is not exported to the environment for child processes.

workspaces

? Default: null

? Type: null or Boolean

Set to true to run the command in the context of all configured workspaces.

Explicitly setting this to false will cause commands like install to ignore workspaces altogether. When not set explicitly:

? Commands that operate on the node_modules tree (install, update, etc.) will link workspaces into the node_modules folder. - Commands that do other things (test, exec, publish, etc.) will operate on the root project, unless one or more workspaces are specified in the

workspace config.

This value is not exported to the environment for child processes.

include-workspace-root

? Default: false

? Type: Boolean

Include the workspace root when workspaces are enabled for a command.

When false, specifying individual workspaces via the workspace config,

or all workspaces via the workspaces flag, will cause npm to operate

only on the specified workspaces, and not on the root project.

This value is not exported to the environment for child processes.

install-links

? Default: false

? Type: Boolean

When set file: protocol dependencies that exist outside of the project

root will be packed and installed as regular dependencies instead of

creating a symlink. This option has no effect on workspaces.

See Also

? npm help install

? package-lock.json /configuring-npm/package-lock-json