# Monorepo

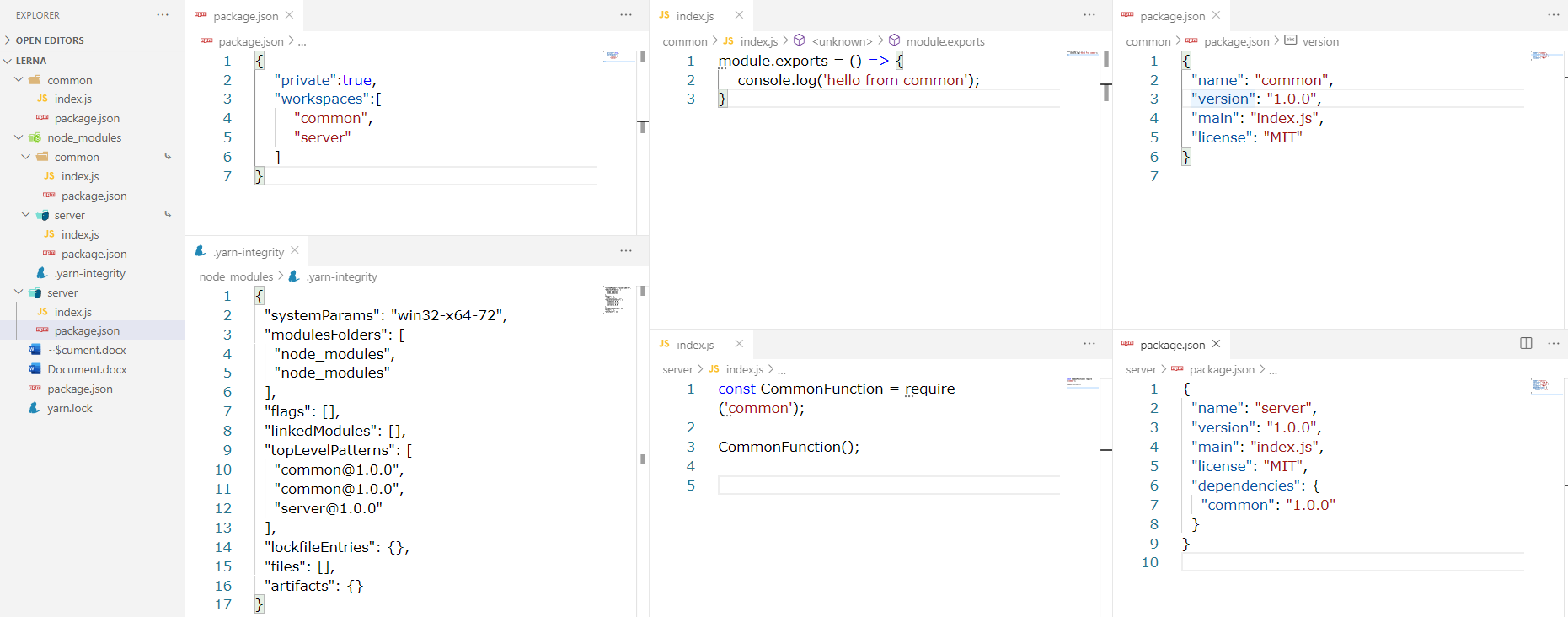
## Yarn Workspaces

Babel is using Yarn Workspaces to include all individual projects under packages folder.

It seems like a ton of overhead to maintain over just splitting up the projects. One of big advantages of this is:

1. You could share dependencies,
2. You could very easily share code between the packages.

### 1. Following project contains two libraries: **common** and **server**. **Server** has dependency on **common** library.

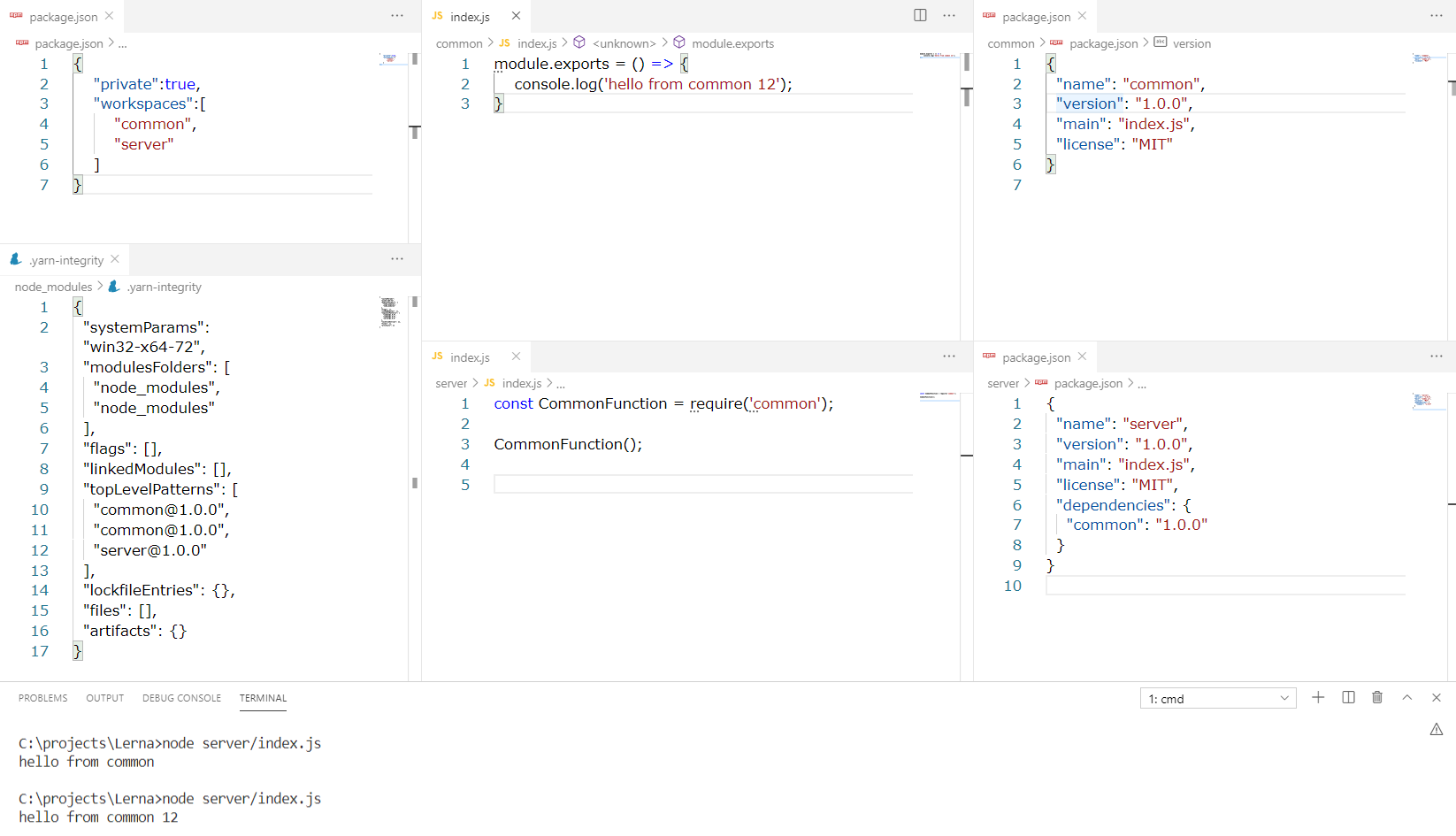


\*: We have to specify “common” and “server” projects under the **workspaces** node in packages.json.

\*: We have to specify dependency on common library at server/package.json.

|  |
| --- |
| C:\projects\Lerna\common>yarn init -y  yarn init v1.22.4  warning The yes flag has been set. This will automatically answer yes to all questions, which may have security implications.  success Saved package.json  Done in 0.04s.  C:\projects\Lerna\common>cd ../server  C:\projects\Lerna\server>yarn init -y  yarn init v1.22.4  warning The yes flag has been set. This will automatically answer yes to all questions, which may have security implications.  success Saved package.json  Done in 0.06s.  C:\projects\Lerna>yarn install  yarn install v1.22.4  info No lockfile found.  [1/4] Resolving packages...  [2/4] Fetching packages...  [3/4] Linking dependencies...  [4/4] Building fresh packages...  success Saved lockfile.  Done in 0.11s. |
| C:\projects\Lerna>node server/index.js  hello from common |

And if we modify common/index.js directly, and the change will be reflected at server side without re-compiling.

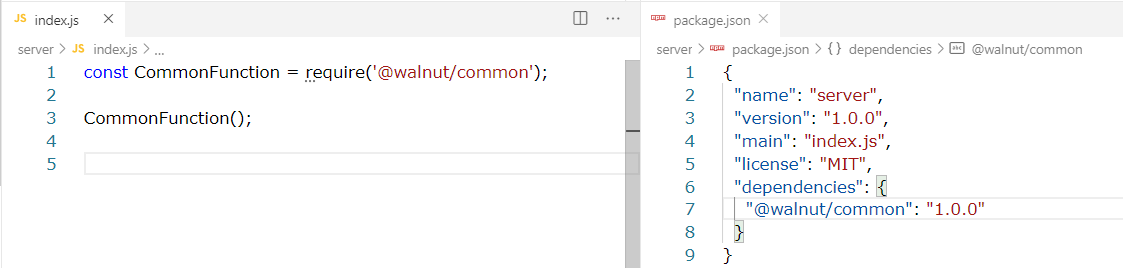


### 2. With Yarn Workspaces, we can also use a better naming structure.

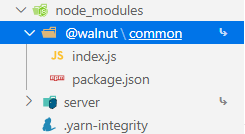
Inside const CommonFunction = require('common'); because ‘common’ is very common library, what people will do is they will change the name of the project (library) inside the project’s package.json file.

|  |
| --- |
| {    "name": "@walnut/common",    "version": "1.0.0",    "main": "index.js",    "license": "MIT"  } |

Then at server project side, we add @walnut/ as well:



Now we have to **yarn install** again, and the folder structure becomes:



### 3. As in Babel.js, we could also move common and server projects into packages folder. Then we have to change in packge.json at the root folder:

|  |  |
| --- | --- |
|  | {      "private":true,      "workspaces":[          "packages/\*"      ]  }  \*: We could just specify the **packages** folder now. |

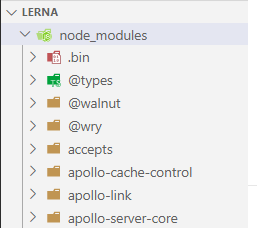
Then we have to **remove all modules** and run **yarn install** again.

|  |
| --- |
| C:\projects\Lerna>rmdir /Q/S node\_modules  C:\projects\Lerna>yarn install  yarn install v1.22.4  [1/4] Resolving packages...  [2/4] Fetching packages...  [3/4] Linking dependencies...  [4/4] Building fresh packages...  success Saved lockfile.  Done in 0.09s.  C:\projects\Lerna>node packages/server/index.js  hello from common 13 |

### 4. Another cool thing with Yarn Workspace is, when I install dependencies, they go to the top level.

|  |
| --- |
| C:\projects\Lerna>cd packages/server  C:\projects\Lerna\packages\server>yarn add graphql-yoga  yarn add v1.22.4  [1/4] Resolving packages...  …  ├─ source-map@0.6.1  ├─ string\_decoder@0.10.31  ├─ subscriptions-transport-ws@0.9.18  ├─ symbol-observable@1.2.0  ├─ type-is@1.6.18  ├─ unpipe@1.0.0  ├─ utils-merge@1.0.1  ├─ uuid@3.4.0  ├─ vary@1.1.2  ├─ ws@5.2.2  ├─ zen-observable-ts@0.8.21  └─ zen-observable@0.8.15  Done in 19.65s. |

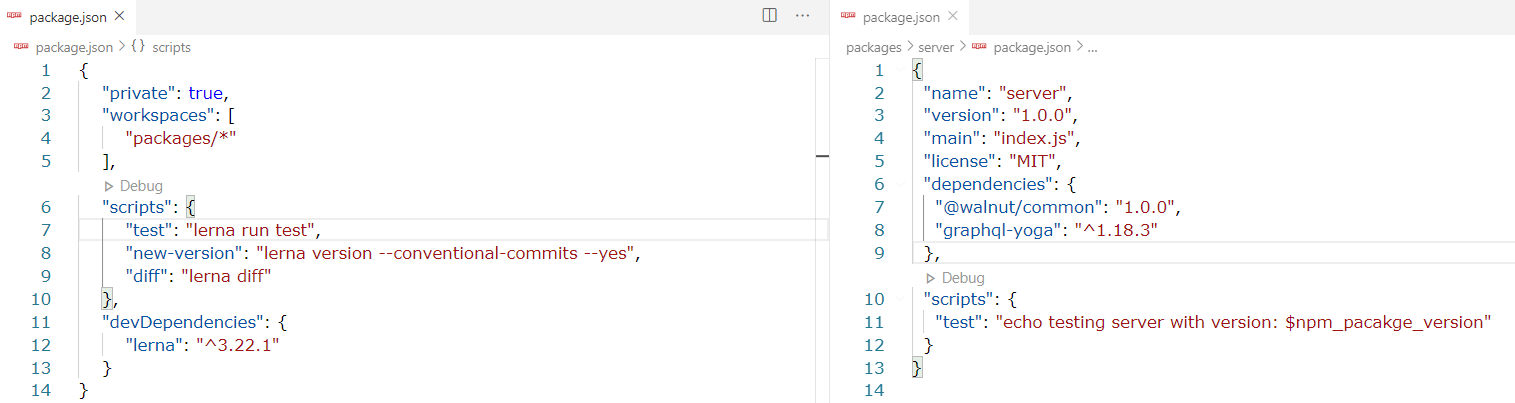
So all installed dependencies are put under the root folder’s node\_modules rather than creating a node\_modules folder under server path.



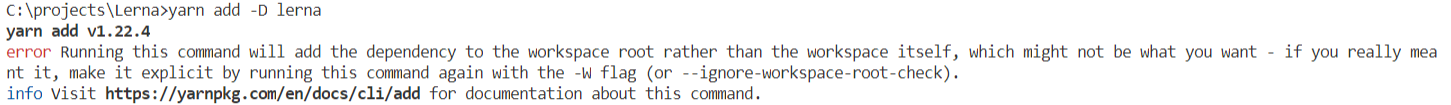
Now even if both common and sever projects need graphql-yoga package, we just need one graphql-yoga package installed. You will have a big node\_modules but you won’t have a node\_module for every single module now.

## Lerna

First thing we’re going to do is add “scripts” section to **root/package.json** and **packages/server/package.json** and **packages/common/package.json**.



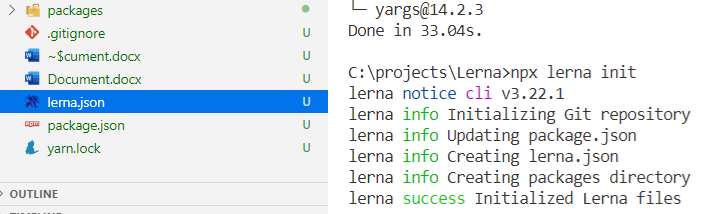
Then we add Lerna to the project as dev dependency.



Yarn doesn’t like us to directly install at workspace root. What we can do is add -W flag.

|  |
| --- |
| C:\projects\Lerna>yarn add -D -W lerna  yarn add v1.22.4  [1/4] Resolving packages...  warning lerna > @lerna/bootstrap > @lerna/symlink-binary > @lerna/create-symlink > @zkochan/cmd-shim > mkdirp-promise@5.0.1: This package is broken and no longer maintained. 'mkdirp' itself supports promises now, please switch to that.  warning lerna > @lerna/bootstrap > @lerna/run-lifecycle > npm-lifecycle > node-gyp > request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142  warning lerna > @lerna/bootstrap > @lerna/run-lifecycle > npm-lifecycle > node-gyp > request > har-validator@5.1.5: this library is no longer supported  warning lerna > @lerna/create > globby > fast-glob > micromatch > snapdragon > source-map-resolve > resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated  warning lerna > @lerna/create > globby > fast-glob > micromatch > snapdragon > source-map-resolve > urix@0.1.0: Please see https://github.com/lydell/urix#deprecated  [2/4] Fetching packages...  [3/4] Linking dependencies...  …  ├─ wide-align@1.1.3  ├─ windows-release@3.3.3  ├─ wordwrap@1.0.0  ├─ wrap-ansi@5.1.0  ├─ write-file-atomic@2.4.3  ├─ xtend@4.0.2  ├─ yallist@3.1.1  ├─ yargs-parser@15.0.1  └─ yargs@14.2.3  Done in 33.04s. |

Then we have to initialize lerna: **npx lerna init**



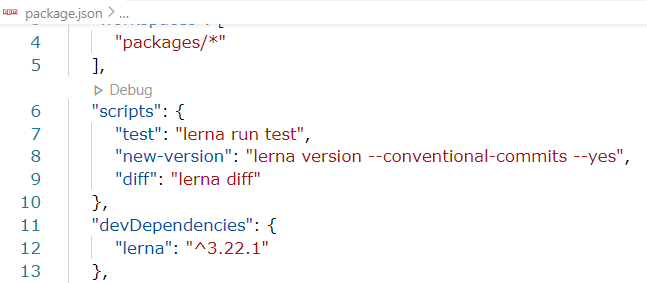
|  |
| --- |
| Those of you upgrading npm to its latest version, [npm@5.2.0](https://github.com/npm/npm/releases/tag/v5.2.0), might notice that it installs a new binary alongside the usual npm: [npx](https://npm.im/npx" \t "_blank).  npx is a tool intended to help round out the experience of using packages from the npm registry — the same way npm makes it super easy to install and manage dependencies hosted on the registry, npx makes it easy to use CLI tools and other executables hosted on the registry. It greatly simplifies a number of things that, until now, required a bit of ceremony to do with plain npm.  <https://medium.com/@maybekatz/introducing-npx-an-npm-package-runner-55f7d4bd282b> |

You can use Lerna without Yarn, but it’s suggested to do so:



### 1. Lerna enabes us to run script across packages

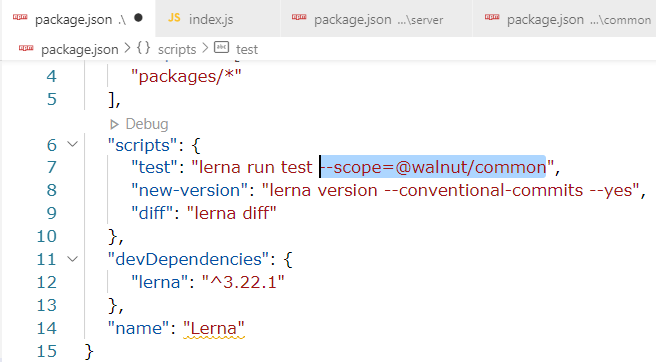
Then the next thing we are going to do is “lerna run test”, which could be called through “npm run test”.



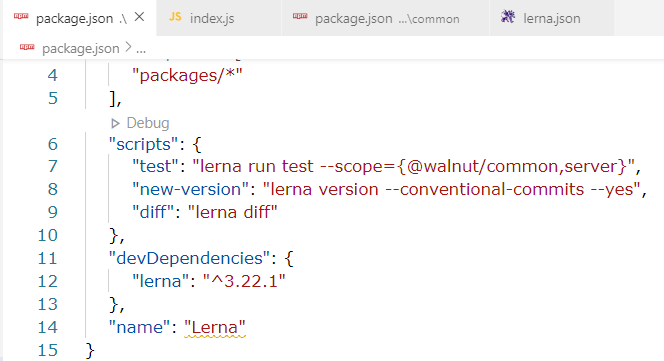
Then it will run test in each project:

|  |
| --- |
| C:\projects\Lerna>npm run test  > Lerna@ test C:\projects\Lerna  > lerna run test  lerna notice cli v3.22.1  lerna info Executing command in 2 packages: "yarn run test"  lerna info run Ran npm script 'test' in '@walnut/common' in 0.6s:  yarn run v1.22.4  **$ echo testing server with version: $npm\_pacakge\_version**  testing server with version: $npm\_pacakge\_version  Done in 0.17s.  lerna info run Ran npm script 'test' in 'server' in 0.6s:  yarn run v1.22.4  **$ echo testing common with version: $npm\_pacakge\_version**  testing server with version: $npm\_pacakge\_version  Done in 0.16s.  lerna success run Ran npm script 'test' in 2 packages in 1.3s:  lerna success - @walnut/common  lerna success - server |

Usually we don’t run across all packages, and we could narrow it down:



For more than one pacakges, we have to use: lerna run test --scope={@walnut/common,server}



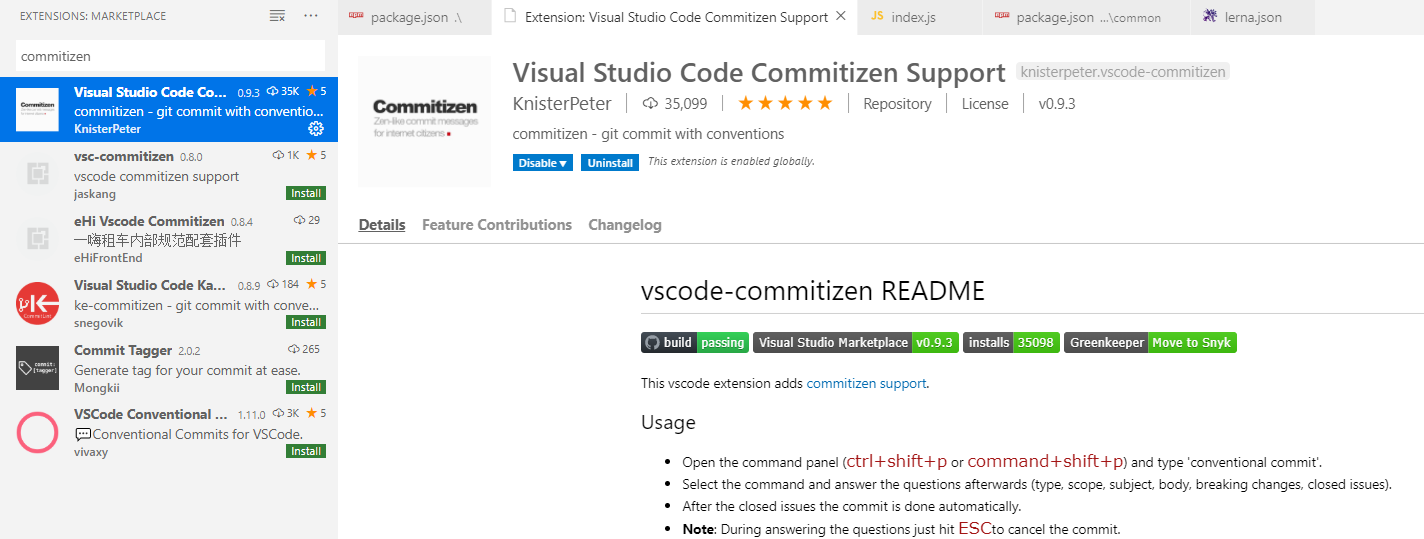
### 2. Versioning our code

Currently our common and server libraries are of version 1.0.0. We could use Lerna to increment the version number as we change the code by using “lerna version” command. "lerna version --conventional-commits --yes"

--conventional-commits: <https://www.conventionalcommits.org/en/v1.0.0/>

|  |
| --- |
| Commit message with description and breaking change footer feat: allow provided config object to extend other configs  BREAKING CHANGE: `extends` key in config file is now used for extending other config files Commit message with ! to draw attention to breaking change refactor!: drop support for Node 6 Commit message with both ! and BREAKING CHANGE footer refactor!: drop support for Node 6  BREAKING CHANGE: refactor to use JavaScript features not available in Node 6. Commit message with no body docs: correct spelling of CHANGELOG Commit message with scope feat(lang): add polish language Commit message with multi-paragraph body and multiple footers fix: correct minor typos in code  see the issue for details  on typos fixed.  Reviewed-by: Z  Refs #133 |

It requires conventional commit, so we could use a tool called Commitizen to input conventional commit message.



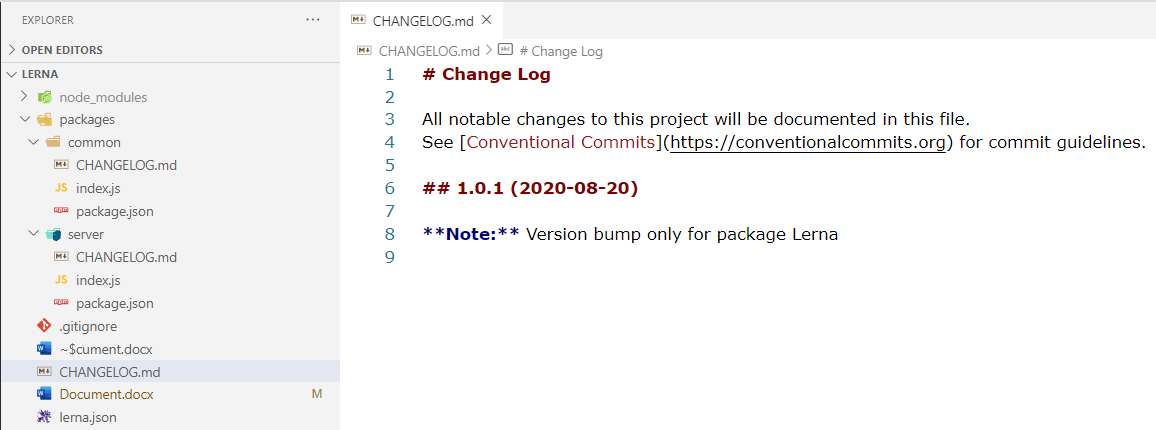
Then we could run “npm run new-version”:

|  |
| --- |
| C:\projects\Lerna>npm run new-version  > Lerna@ new-version C:\projects\Lerna  > lerna version --conventional-commits --yes  lerna notice cli v3.22.1  lerna info current version 0.0.0  lerna info Assuming all packages changed  lerna info getChangelogConfig Successfully resolved preset "conventional-changelog-angular"  Changes:  - @walnut/common: 1.0.0 => 1.0.1  - server: 1.0.0 => 1.0.1  lerna info auto-confirmed  lerna info execute Skipping releases  lerna info git Pushing tags...  lerna success version finished |

We could see the versions of @walnut/common and server have been changed.

There will be a CHANGELOG.md file auto generated as well:

For now there is a auto-generated note saying the commit is **version bump only**.



Let’s do some actual change: