LernaJS

Tuesday, October 10, 2017 10:18 AM

Basics (reference)

LernaJS is "a tool for managing JavaScript projects with multiple packages." It should provide a much better workflow than having to produce a new repository and publish modules every time you want to share functionality between related projects.

Getting started (making a Lerna "repo")

I had tried lerna-wizard, but it isn't very popular and I ran into an issue quickly.

In testing this, I did these steps:

1. lerna init --independent
    a. Independent Mode (reference) makes it so that you can have individual version numbers for the "sub" projects.
2. Commit to the repo that gets created
    a. I added "lerna-debug.log" to my .gitignore so that errors wouldn't get checked in.
3. lerna import <path to external repository>
    a. **WARNING: DO NOT EVEN RUN THIS COMMAND UNLESS YOU HAVE CHECKED THE STATUS OF THIS ISSUE (see troubleshooting section of this note)**
    b. Note: the path is on your filesystem still, but it's imported locally to the brand new repo that you committed to in the last step. This is really helpful since by the end of these steps, you'll be able to test everything in your Lerna repo without having modified the source repos. That way, if there's a problem, you can just wipe out the entire Lerna repo if you want.
    c. This will ask to merge git history so that your commits remain intact. It doesn't do an "npm install" or anything though.
    d. Note: this will import *without* interleaving Git history. For example, if you import RepoA and then RepoB, you'll get all of RepoA's history first regardless of what dates that may contain, then all of RepoB's history second regardless of whether it came before RepoA.
4. lerna bootstrap
    a. This doesn't take args. It will "npm install" everything, link everything, run prepublish, and a bunch of other goodness that you need to do probably after every import.

Cloning a Lerna "repo"

This is what the "bootstrap" command is for: it installs all package.json files and links them together.

```
lerna bootstrap
```

Adding a new module to an existing Lerna repo

Notes:

- This assumes you've set up the repo using "lerna init", "import", "bootstrap", etc. as shown in the "Getting started" section.
- The module that you're adding needs to have a package.json in it, otherwise it can't be published. If you don't want it to be published, then you can put "private": true in the package.json

1. Make a new folder in "packages"
2. "npm init" (feel free to put "-y" if you want so that you don't have to answer questions)
    a. Don't forget to put "@namespace/" before the name if you don't want it to be publicly published
3. Add a ".gitignore" to this folder for node_modules if you haven't already blocked it recursively.

    a. Remember: .gitignore is used for .npmignore if you didn't define a .npmignore. They're both structured the same way (e.g. just put "node_modules" in there if you want to ignore everything under node_modules).
4. Import your module from any other modules that may need it (even if you haven't finished writing it yet, that way all of these steps can be compact and done before you get to the code).
    a. The way I did this so that I didn't have to publish first was to just manually type into package.json with version 1.0.0 (which is what "npm init" gives you by default).
5. Run "lerna link" at the end

Publishing

First, run "git status" to make sure you have no unmodified files. I had modified "package.json" by adding a dependency, and my first "lerna publish" didn't actually detect that the file already had changes, so it happily bumped the version number and committed without warning me.

After publishing for the first time, "lerna updated" will be used so that you only publish what changed.

"--exact" is useful so that you're using exact dependencies instead of ranges when you update the users of those dependencies.

Testing local changes

I needed to use a fork that someone had made in a pull request, so I got the URL of the fork by going to that person's GitHub and finding their Lerna fork. I cloned the repo, switched to their fork branch (in this case it was "git checkout improve-import-edge-cases"), did "npm install", "npm build", "npm link" (so that it was the global "Lerna"), and I was good to go.

Troubleshooting

Import fails due to a merge-related issue

I got an error that looked like this

```
lerna ERR! import Failed to apply commit b20444c7.
lerna ERR! import Error: Command failed: git am -3
lerna ERR! import error: Failed to merge in the changes.
lerna ERR! import Applying: * Changed init_database.js so that it can be called from Ansible and correctly report when it fails. To accommodate this, I needed to add an unhandledRejection handler and also change the knex calls such that there wouldn't be errors if the database or the user already existed.
lerna ERR! import Using index info to reconstruct a base tree...
lerna ERR! import M     packages/bot-land/central_server/database/init_database.js
lerna ERR! import Falling back to patching base and 3-way merge...
lerna ERR! import Auto-merging packages/bot-land/central_server/database/init_database.js
lerna ERR! import CONFLICT (content): Merge conflict in packages/bot-land/central_server/database/init_database.js
lerna ERR! import Patch failed at 0001 * Changed init_database.js so that it can be called from Ansible and correctly report when it fails. To accommodate this, I needed to add an unhandledRejection handler and also change the knex calls such that there wouldn't be errors if the database or the user already existed.
lerna ERR! import The copy of the patch that failed is found in: .git/rebase-apply/patch
lerna ERR! import When you have resolved this problem, run "git am --continue".
lerna ERR! import If you prefer to skip this patch, run "git am --skip" instead.
lerna ERR! import To restore the original branch and stop patching, run "git am --abort".
lerna ERR! import
lerna ERR! import Rolling back to previous HEAD (commit
```

8ab9f82d58b5daefefdd780ece869372e3e78dd3).
lerna ERR! import You may try with --flatten to import flat history.
lerna ERR! execute callback with error
(node:16948) UnhandledPromiseRejectionWarning: Unhandled promise rejection (rejection id: 3):
TypeError: Cannot read property 'split' of undefined

I fixed this by specifying the "flatten" argument (reference).

Git is reporting two distinct items despite having the same path

These repro steps caused me to run into this existing issue:

1. mkdir test_git_paths4
2. cd test_git_paths4
3. lerna init
4. git commit -m "init"
5. lerna import D:\Code\BotLand\util
    a. This is just some arbitrary package that happens to have an "index.js" as shown below

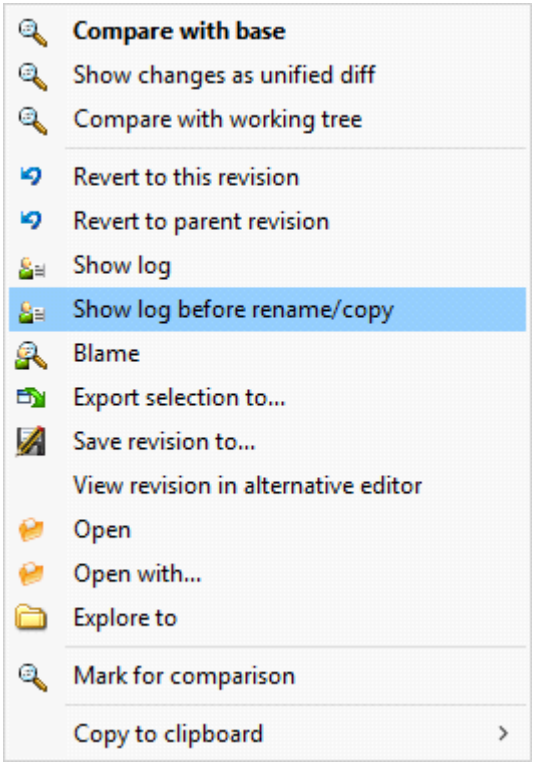
Bug: Lerna's first commit looks like this (notice the backslash):

Path	Extension	Status	Lines added	Lines removed
packages\util/.gitignore	.gitignore	Added	5	0
packages\util/.npmignore	.npmignore	Added	1	0
packages\util/package.json	.json	Added	25	0
packages\util/src/index.js	.js	Added	450	0

I worked around this as described in my comment on the issue itself by adding a line of code that replaces the backslashes with a forward slash. This fix gets broken every time Lerna gets updated until they fix the issue properly.

Can't see history before the Lerna changes in Bot Land

This is really just a generic Git issue where you need to see the history of a file that's been renamed (or moved):


Compare with base
Show changes as unified diff
Compare with working tree
Revert to this revision
Revert to parent revision
Show log
Show log before rename/copy
Blame
Export selection to...
Save revision to...
View revision in alternative editor
Open
Open with...
Explore to
Mark for comparison
Copy to clipboard