

Building the Monorepo

“

kchau@microsoft.com

Flywheel Team

”

About Monorepos

- JS codebases have grown, and are maintained by **thousands of devs**
- Modularity is needed: **package as unit of modularity**
- Related packages are updated at the **same commit**
- Monorepo require a **Monorepo Management Stack**

Monorepo Management Stack

1. Workspace-enabled package manager

- **Installs dependencies** for all packages
- **Links internal packages** to satisfy the node resolution algorithm
- Handles **dependency resolution** for all packages
- Optional: **hoisting**, **strictness** enforcement (phantom deps)
- On the market: `yarn` , `pnpm` , `rush` , `lerna + npm`

2. Task scheduler & runner

- **Runs npm scripts** for all packages
- **Optimize** task run speeds at the dev machine and CI
- Optionally in **topological** order or in **parallel**
- On the market: `lerna` , `wsrn` , `rush` , `pnpm`
`recursive` , `lage`

3. Package publish tool

- Automated management of **semver**
 - Change description files or commit messages
- **Validation** of description of changes
- **Synchronize versions** between npm registry and git repository
- Automated **changelog creation**
- On the market: `rush`, `lerna`, `semantic-release`, `beachball`

Our Focus: Task scheduler & runner

Problem statement

Create a task runner that optimizes package tasks in a monorepo for a single machine

Current state

- JS monorepos in the wild run with **all kinds of workspaces**
- The state-of-the-art monorepo task runners are **not optimized**
- **CPU cores sit idle** for topological scripts
- Large monorepos generally have **clustered graph** of related packages

Philosophy

- Distribute work via smaller libraries with multiple owners
- Leverage OSS as much as possible
- Support package.json scripts as the script runner

Requirements of a task runner

- **Open sourced**
 - easily shared, public development demands **polish**
 - **easily contributed** to by many groups
- Works with all workspace implementations
- Easy setup
- Minimize idle CPU cores
- **Sublinear increase** in build time per package

Prior Art

- This should sound familiar because Vincent made a version for Midgard

Lage

“

v. to make (Norwegian); pr. LAH-geh

”

- Open sourced: <https://github.com/microsoft/lage>
- Easy to integrate with existing codebase
- Scales up with pipelining
- Scales out with caching and scoping

Collaboration

- **OneDrive/SharePoint:** `rush` showed us incremental builds
- **Midgard:** `backfill` cache, `task-scheduler`
- **Flywheel:** pipeline config, `workspace-tools`, `lage` tool
- **FluidX:** `p-graph` prioritized promise-based queuing

What does it look like?

Full Build

```
$ lage build test lint --grouped --verbose --reset-cache
```

```
info codeandbox/react-template lint completed, took 11.10s
info codeandbox/react-orchestrator-template lint completed, took 12.69s
info @pinstudio/react-button build completed, took 16.87s
info @pinstudio/react-tabs build completed, took 16.79s
info @pinstudio/storybook lint completed, took 21.18s
info dev-tests build completed, took 20.15s
info @uiFabric/charting build completed, took 44.79s
info @uiFabric/date-time build completed, took 19.43s
info @uiFabric/example-app-base test completed, took 15.42s
info @uiFabric/example-app-base lint completed, took 48.42s
info @uiFabric/experiments build completed, took 16.10s
info @uiFabric/lite build completed, took 46.71s
info @uiFabric/react-cards build completed, took 16.12s
info dev-tests lint completed, took 11.09s
info @uiFabric/charting test completed, took 42.43s
info @uiFabric/charting lint completed, took 58.19s
info @pinstudio/react-button test completed, took 11.86s
info @pinstudio/react-button lint completed, took 16.12s
info @pinstudio/react-fire build completed, took 46.40s
info @pinstudio/react-tabs test completed, took 44.11s
info @pinstudio/react-tabs lint completed, took 19.27s
info @uiFabric/lite test completed, took 18.28s
info @uiFabric/lite lint completed, took 15.48s
info @uiFabric/date-time test completed, took 41.33s
info @uiFabric/date-time lint completed, took 42.84s
info @pinstudio/react-next build completed, took 16.79s
info @uiFabric/experiments test completed, took 46.40s
info @uiFabric/experiments lint completed, took 16.10s
info @pinstudio/react-fis lint completed, took 11.45s
info thesing-designer build completed, took 11.75s
info @uiFabric/api-docs build completed, took 10.23s
info @uiFabric/react-cards test completed, took 10.86s
info @uiFabric/react-docs lint completed, took 18.19s
info @uiFabric/fabric-white-resources build completed, took 28.28s
info @uiFabric/api-docs test completed, took 7.12s
info @uiFabric/api-docs lint completed, took 17.47s
info thesing-designer test completed, took 5.18s
info thesing-designer lint completed, took 16.20s
info ally-tests build completed, took 19.73s
info @uiFabric/fabric-white-resources build completed, took 20.10s
info @uiFabric/fabric-white-resources lint completed, took 14.89s
info dev-tests build completed, took 16.86s
info @pinstudio/example build completed, took 15.48s
info @pinstudio/example lint completed, took 9.87s
info ally-tests test completed, took 2.44s
info ally-tests lint completed, took 9.81s
info @uiFabric/fabric-white test completed, took 3.81s
info @uiFabric/fabric-white lint completed, took 16.11s
info dev-tests test completed, took 1.11s
info @pinstudio/docs lint completed, took 18.44s
info @pinstudio/pure lint completed, took 1.46s
info codeandbox/react-next-template build completed, took 12.61s
info perf-test build completed, took 22.17s
info text-handlers build completed, took 11.38s
info dev-tests build completed, took 40.49s
info @pinstudio/react-next test completed, took 16.42s
info @pinstudio/react-next lint completed, took 16.10s
info codeandbox/react-next-template lint completed, took 18.73s
info text-handlers test completed, took 4.41s
info text-handlers lint completed, took 19.71s
info perf-test test completed, took 4.18s
info perf-test lint completed, took 21.81s
info @uiFabric/pr-deploy-lite build completed, took 7.78s
info @uiFabric/pr-deploy-lite test completed, took 4.72s
info dev-tests test completed, took 2.12s
info
info Took a total of 104 12.76s to complete
Done in 143.04s.
/home/USER-00013101-/workspace/pinstudio
```


Cached Build

```
$ lage build test lint --grouped --verbose
```

[illegible]

Scoped Build

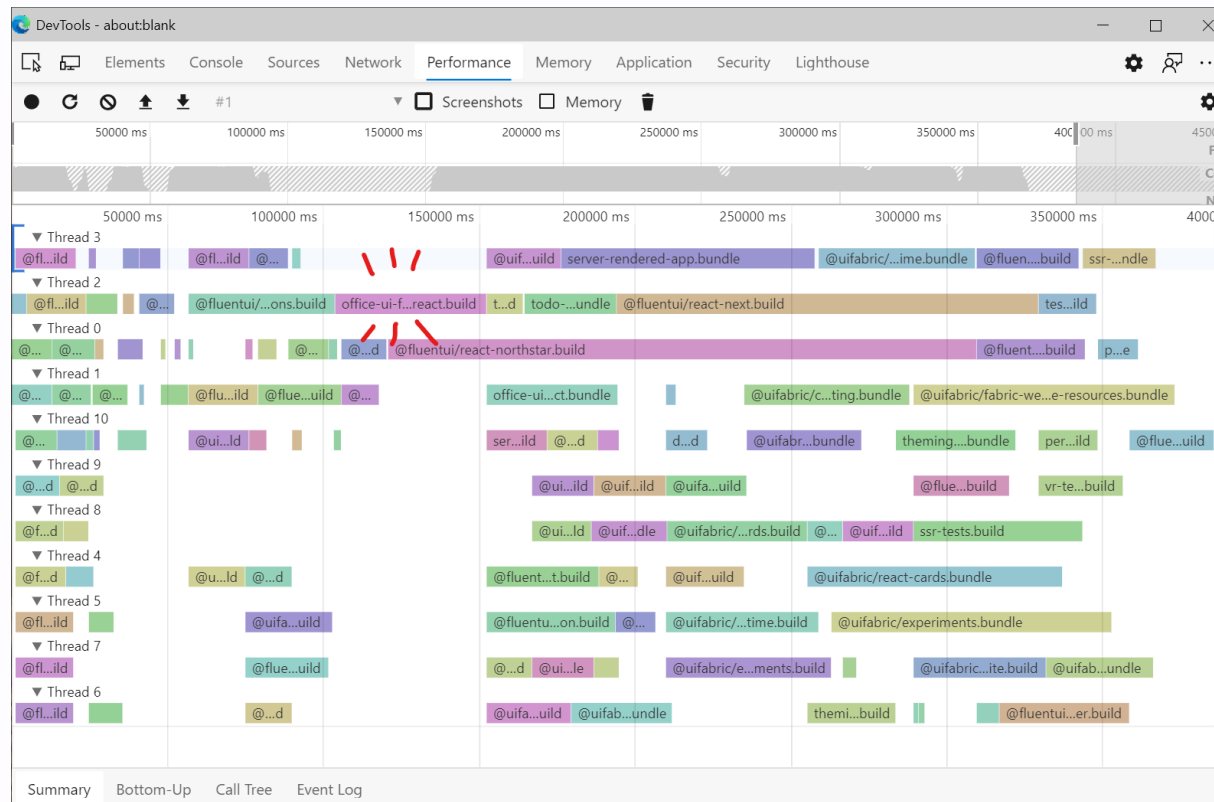
```
$ lage build test lint --grouped --verbose --scope @fluentui/web-components
```

```
~/workspace/fluentui$ npm run build test lint --verbose --no-cache --scope @fluentui/web-components
npm run v1.22.4
$ lage build test lint --verbose --no-cache --scope @fluentui/web-components
info lage test runner - let's make it
verb @fluentui/web-components build start
verb @fluentui/web-components build Running /home/ken/.npm/_nodedir/node/v12.18.1/bin/node run build --
verb @fluentui/web-components build with --script-path The node binary used for scripts is /home/ken/.npm/_nodedir/node/v12.18.1/bin/node itself. Use the --script-path option to include a
verb @fluentui/web-components build path for the node binary you are executing with.
verb @fluentui/web-components build @ @fluentui/web-components@0.1.2 build /home/ken/workspace/fluentui/packages/web-components
verb @fluentui/web-components build $ npm --script-path ../../node_modules/.bin/node run build
verb @fluentui/web-components build $ npm --script-path ../../node_modules/.bin/node run build
verb @fluentui/web-components build $ npm --script-path ../../node_modules/.bin/node run build
```



Profiling

```
$ lage build test lint --profile
```



How does it work?

<https://microsoft.github.io/lage/guide/levels.html>

How try it at home?

<https://microsoft.github.io/lage/guide/getting-started.html>

1. npm scripts (build, test, lint) are at package level
2. `npx lage init`
 - creates a `lage.config.js` - configure it
 - adds `lage` as a dep
3. `yarn lage build` or `npm run lage build`

Future

- `lage` is a great solution for a **single machine**, not distributed builds
- For MSFT is `buildxl`
 - `lage` needs to spit out `dscrip` or json config for `buildxl`
- Alternative: also investigate `bazel`
 - `lage` can potentially spit out `WORKSPACE` & `BUILD`

More info

Github:

<https://github.com/microsoft/lage>

Documentation:

<https://microsoft.github.io/lage/>

Complex Configuration:

<https://github.com/microsoft/fluentui/blob/master/lage.config.js>