# Yarn 3 Migration

Unravelling the knots of despair

# Preface

- This talk is mostly frontend but the docker bit might be interesting if you're not working frontend.
- Yarn is an alternative package manager for Node. It needed to be upgraded from "classic" (v1) to "berry" (v3) on all node/express repos in HMCTS.
- Upgrade urgency was due to the PlatOps deadline hitting last month.

# Introduction / Postmortem

- Did the Yarn v3 migration for P&I (CaTH) and helped a few people within cross cutting with their own migrations.
- Moved to ET, immediately made a massive error in my second week ”**helpfully”** upgrading the ccd-definition-processor repo to v3.
- Spent the next two weeks fixing my own mess with Yarn for about 10 teams.

# Danny’s body count

Blocked the following dev teams in a major way:

- Adoption
- Divorce
- Special Tribunals
- Civil General Apps
- Financial Remediation
- Private Law
- SSCS

# Danny’s body count (part 2)

- Blocked two separate releases (only one of which was saveable)
- Spent ~2 weeks on damage control.
- Yarn pipeline deadline helpfully hit on Apr 26th, prompting more teams to ask for help with migration.
- Did I mention I have a newborn baby at home?

# Lessons Learned

- Before upgrading a repo, make sure you check nobody is using it as a docker dependency!
- Bad disclosure of urgency – my vague warnings were not clear enough.
- Through this painful episode, I was able to find lots of Yarn v3-specific complexities – i.e., the rest of the talk

# ‘Simple’ approach for migrating to Yarn v3

- Follow Yarn’s own migration guide: https://yarnpkg.com/getting-started/migration
- Although advice suggests moving to the ”plug-n-play” (pnp) module linker, that can be very complex for existing projects.
- Best approach is to stick with the node_modules linker for a first pass, then raise a ticket to switch to pnp resolution later.
- Don’t expect anything to work exactly as it did before. There were lots of changes (both documented and undocumented).
- Lots of gotchas to be found even in the simple migration strategy.

# Primary issues found within HMCTS repos

- `Dockerfile` and `.dockerignore` changes required.
- Pointing to `:latest` for specific layers of your `Dockerfile` is a bad practice if you don't directly own the dependency. Point at a specific image ref that you know works. (https://vsupalov.com/docker-latest-tag/)
- Some old commands no longer work or work differently. Some examples: yarn audit, yarn install, yarn global
- Git submodules and the `yarn --cwd` command causing `AmbiguousSyntaxError`.
- `Package.json` files are parsed differently.
- File structure changes (e.g. `.yarnrc` -> `.yarnrc.yml`)

# Docker Integration – Debug Tips

- Generally, ”Docker Build” stage in the pipeline is equivalent to running: `docker build .`
- If the pipeline fails at the Helm Upgrade stage, this is the equivalent of running: `docker run <image id>`
- If either of these commands fail locally, they will likely fail on the pipeline. Save some public funds by trying stuff out locally before pushing.
- There are additional complexities with working directories and permissions that I’m happy to help with if requested.

# Docker Integration - Corepack

- Docker doesn’t inherently know you are using yarn v3, and the HMCTS base images (currently) default to v1 unless you specifically tell them in your Dockerfile:

```
USER root
RUN corepack enable
```

- Pretty much every team in HMCTS is still using yarn v1 in their Dockerfiles. 51 unarchived Node repos within HMCTS still require this change, even though they’ve migrated to v3 already.

# .dockerignore whitelisting

- Dockerignore whitelisting (best practice!) makes it difficult to debug what's going wrong but can be inferred sometimes.
- E.g., if `TypeError` appears after migration, you might have forgotten to whitelist `tsconfig.json`
- Oft-forgotten files & dirs include:
  `.yarnrc.yml`
  `.yarn/releases`
  `.yarn/plugins`

# .dockerignore blacklisting

- This is generally easier to understand during debugging. A lot of teams just do something like this:
  `COPY --from=base . .`
  This means you can for the most part just add the same stuff to `.dockerignore` that you did for `.gitignore`.

# Dockerfile Issues

- Lots of projects haven't yet migrated their docker images over to the Azure Container Registry. If the `FROM` stages of your Dockerfile do not contain `hmctspublic.azurecr.io` they need to be updated.
- Remember, your Dockerfile is the blueprint for what gets deployed – keep it updated!

# Package.json quirks

- Yarn will no longer be able read any non-standard sections of your package.json (e.g. “config”)
- This means if you use `$npm_package_config` you will need to use a new method to read config into your scripts (e.g. .env files).

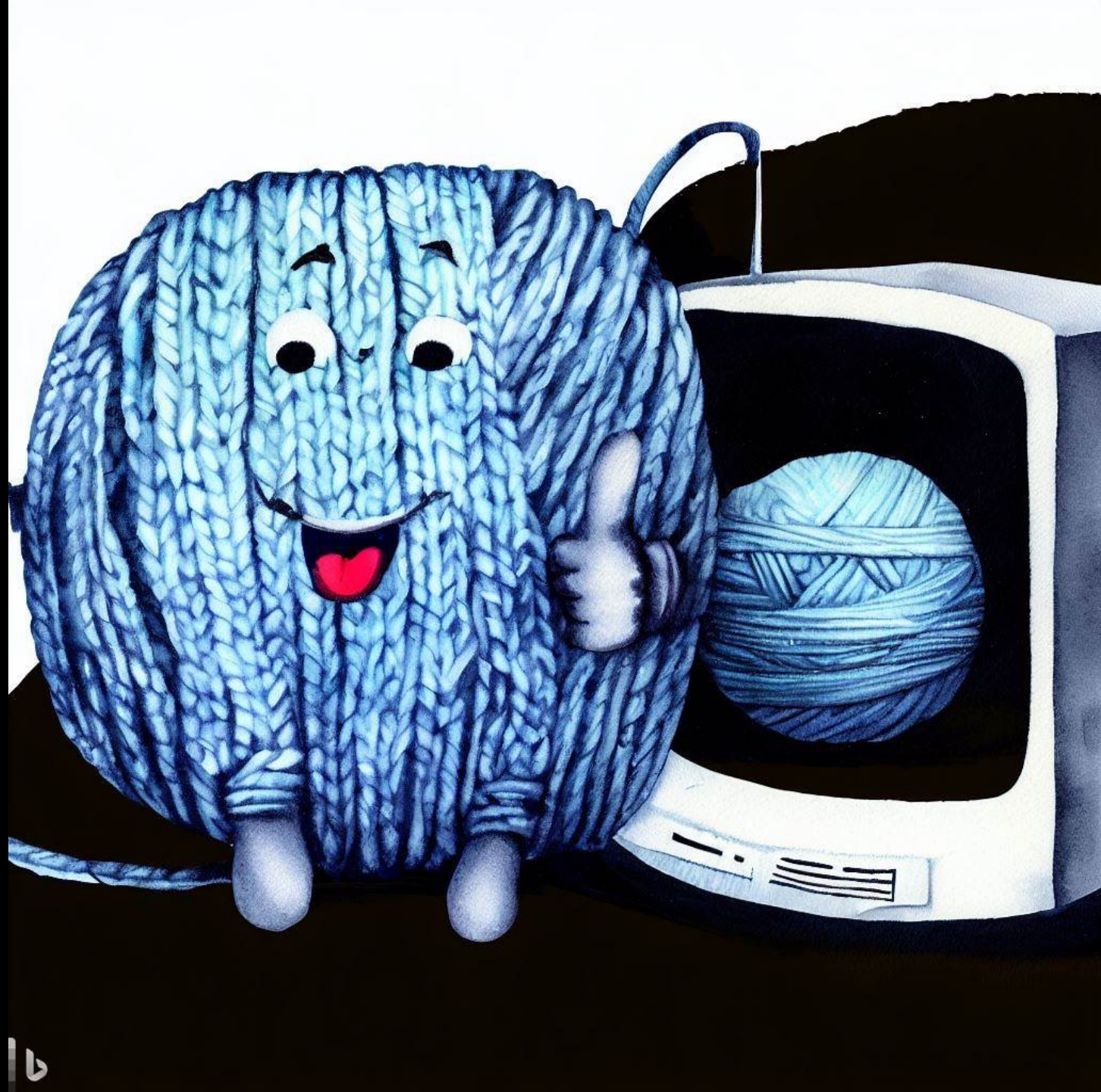
# Package.json quirks

- You can no longer easily use `yarn --cwd <script>` for git submodules. Don’t worry, this is easily solvable by simply using shell commands. For example: `cd <dir> && yarn <script>`

# Is it even worth it to upgrade?

# Yes!

- `yarn upgrade-interactive` is **fantastic** for upgrading many dependencies at once.
- Pnp linking makes an enormous difference to both storage and network requirements on projects (~5x smaller than node_modules).
- It has pretty colours!



# Questions?

- For yarn-related queries, contact me on danny.furnivall@justice.gov.uk
- Slides available at: https://github.com/hmcts/yarn-v3-presentation
- I don’t have six fingers.