Docker-out-of-Docker, and custom job images

Jobs run inside containers, but the jobs themselves need to *build* containers. The classic tangle is "Docker inside Docker." The cleaner approach used here is **Docker-out-of-Docker**: the runner mounts the host's Docker socket into the job container, so `docker build` inside a job actually runs against the host's Docker engine.

```
job container  --/var/run/docker.sock-->  host Docker engine (on GIT-Runner)
```

That avoids privileged nested-Docker entirely. The job image just needs a `docker` client on it.

For the job images themselves, I built **custom ones** (`ubuntu:24.04` / `ubuntu:26.04` bases plus Node, Git, the Docker CLI and buildx) and pushed them to the private registry. The reason: the stock minimal images lack the tools real jobs need, and rebuilding that toolchain at the start of every job is slow. A purpose-built job image makes pipelines start fast and behave predictably.

> **Lesson learned:** Docker-out-of-Docker beats Docker-in-Docker for a build runner — fewer privileges, no nested-daemon weirdness, and builds land in a cache you can reuse. And invest a little in a good job image; the time you spend baking tools in is repaid on every single pipeline run.

Revision #1
Created 2026-05-22 09:24:02 UTC by Eslam Shapsough
Updated 2026-05-22 09:33:02 UTC by Eslam Shapsough