

Developer Dashboard

Docker Hub image: [manifestation/developer-dashboard](#)

Short description

A developer-focused CLI and local dashboard for project context, workspaces, and everyday tools.

Full description

Developer Dashboard (command: `d2`) is a terminal-first workbench that brings project context and common development workflows into one consistent place. It helps developers move between repositories, services, and tasks without rebuilding their setup from shell history or a pile of one-off scripts.

Why developers use it

Project paths, useful links, local status checks, service commands, and workspace entry points often end up scattered across notes, aliases, and scripts. Developer Dashboard makes those tools discoverable from one command-line interface and an optional local web dashboard. The result is less context switching, faster task hand-offs, and a more repeatable way to work across projects.

What it brings to the workflow

- **Project context:** organize named paths and project-specific settings so frequently used locations are easy to find.
- **One command surface:** access dashboard pages, installed skill commands, status indicators, and configured collectors through `d2`.
- **Service workflows:** use integrated Docker Compose commands for the services configured in your environment.
- **Workspaces:** enter configured development workspaces, including container-based workspaces where set up.
- **Local dashboard:** bring project links and operational context into a browser view served by the local environment.
- **Layered configuration:** keep personal defaults and project-level customizations organized together.

Get the image and try the CLI

Pull the latest image, then check the installed command:

```
docker pull manifestation/developer-dashboard:latest
docker run --rm manifestation/developer-dashboard:latest version
```

To run a command against your current project directory, mount it into the container:

```
docker run --rm -v "$PWD:/workspace" -w /workspace \\\n  manifestation/developer-dashboard:latest --help
```

The image starts `d2` by default. Supply the project files and any runtime configuration your chosen workflow needs.

Docker Compose example

Save this as `compose.yaml` in a project directory:

```
services:
  d2:
    image: manifestation/developer-dashboard:latest
    working_dir: /workspace
    volumes:
      - ./workspace
    command: ["version"]
```

Run the example with:

```
docker compose run --rm d2
```

To run another command, override the Compose command, for example `docker compose run --rm d2 --help`.

Image details

- Linux container image for `linux/amd64` and `linux/arm64`.
- The image is a developer tool environment, not a hosted service.
- Apple Silicon can run the ARM64 Linux image through Docker Desktop's Linux virtual machine; containers do not use a macOS kernel.