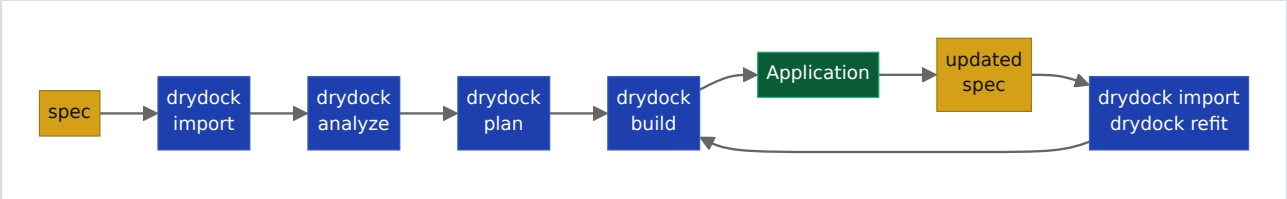
DRYDOCK TURNS SPECIFICATIONS INTO WORKING SOFTWARE.

Quick Start Guide

Ed Barlow www.webcloudstudio.com August 2026

This quick start guide builds a trivial application called ReadingList.



- `import` copies your specification material into a workspace.
- `analyze` decomposes the import/epic into stories.
- `plan` grooms blueprints, ac, and the build graph.
- `build` creates software testing each stp.
- `refit` diffs updated specs into tickets.
- The `quarterdeck` web interface provides control and observability.

Before you begin

1. Set up the `claude` or `codex` in your `PATH` (subscription API).
2. Install uv (easiest and its so fast.)

```
curl -LsSf https://astral.sh/uv/install.sh | sh       # macOS and Linux:
brew install uv                                        # macOS with Homebrew
winget install --id=astral-sh.uv -e                    # Windows with WinGet

# Windows PowerShell:
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"
```

3. Install Drydock (User Installation Guide).

```
# Recommended — isolated tool install:
uv tool install drydock-sdd

#  Alternative with `pipx`:
pipx install drydock-sdd

# Or if you want to manually setup your path:
python -m pip install drydock-sdd
```

`uv tool` and `pipx` install Drydock in a dedicated environment and put a command wrapper on your `PATH`; they are the right choice for an interactive CLI. `pip` installs

Drydock into whatever virtual environment is active and requires you to update your `PATH`.

4. Review your setup with

```
drydock --version
drydock config show                          # show your configuration
      # drydock config set drydock_workspace <Workspace for the drydock>
      # drydock config set drydock_build_directory <Your application parent directo
      # drydock is scoped to work only in these directories.
```

1. Create Target Workspace

Create workspace in `drydock_workspace/targets/<target>` and build target in `drydock_build_directory/<target>`.

```
# project ReadingList workspace in `drydock_workspace/targets/<target>`
# project ReadingList builds to `<drydock_build_directory>/ReadingList`
drydock init ReadingList             # Initialize project workspace
```

2. Import Your Specification

Create source specification material in a file (example: `reading-list.md`)

```
# Reading List

Build a web application that keeps a list of books to read.

The reader can add a book with a title and author, view the books in the order adde
and remove a book. An empty title or author is rejected with a clear error message

The application includes automated tests for each behavior.
```

Import the file you just created:

```
drydock import ReadingList ./reading-list.md
drydock score spec ReadingList                   # Optional
```

3. Analyze

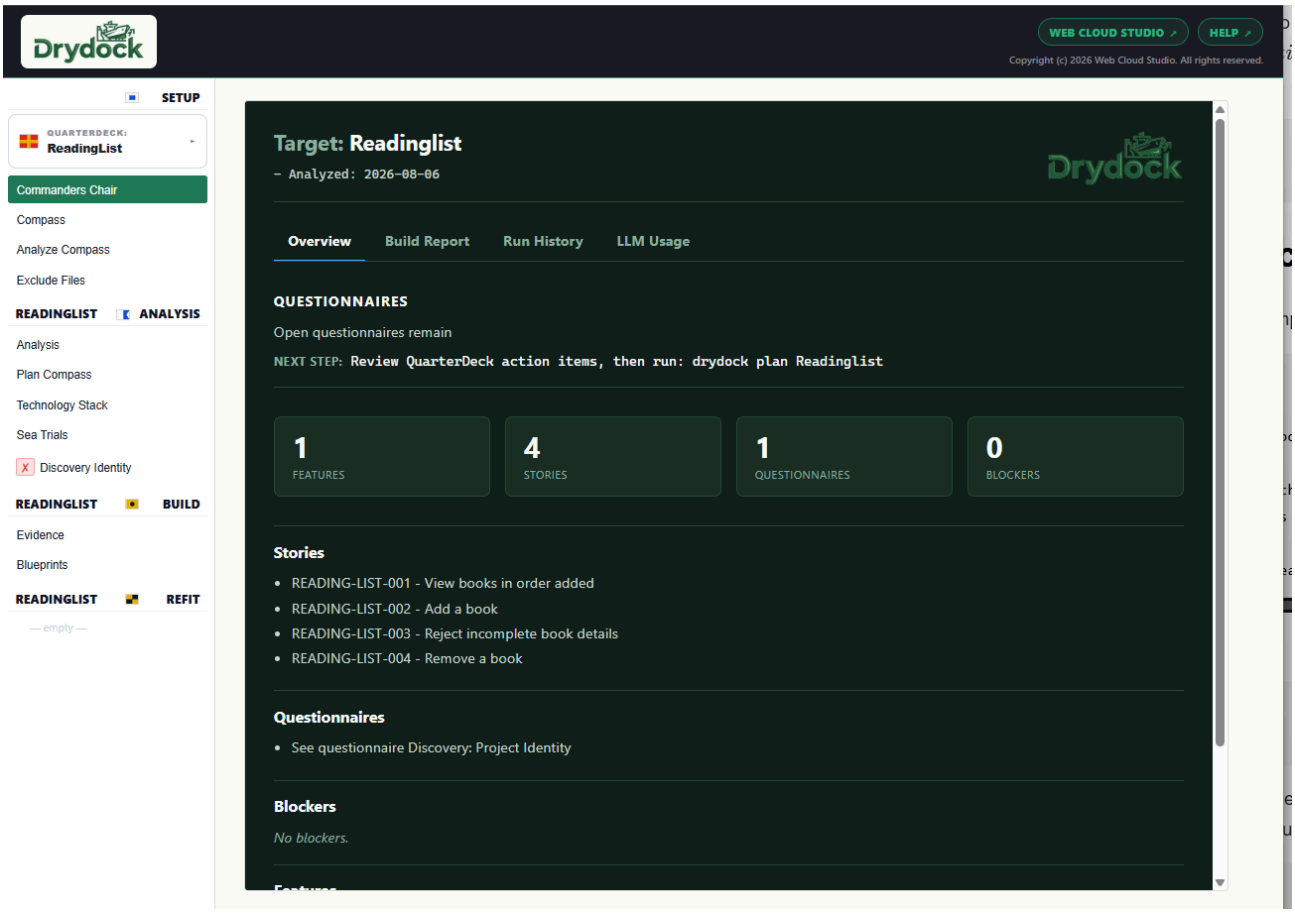
The analyze command performs an initial analysis of the source material. Analyze turns the source material into stories, acceptance criteria, questions, and blockers.

```
drydock analyze ReadingList
drydock run quarterdeck ReadingList
```

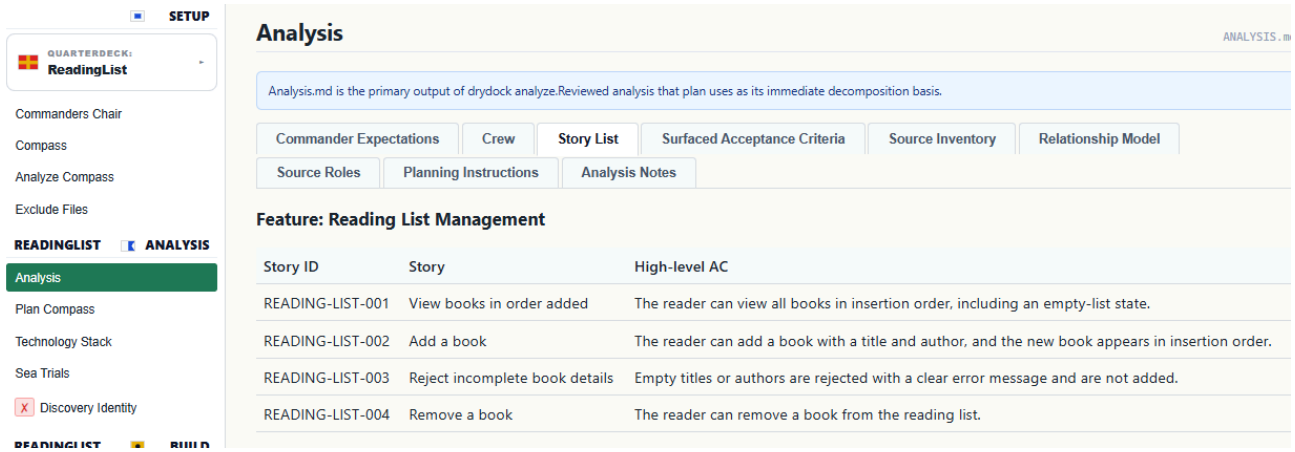
Navigate to the QuarterDeck web server at the default address, http://127.0.0.1:8080. Review the Target, then press `Ctrl+C` in the terminal to stop the server and continue.

QuarterDeck provides the following screens:

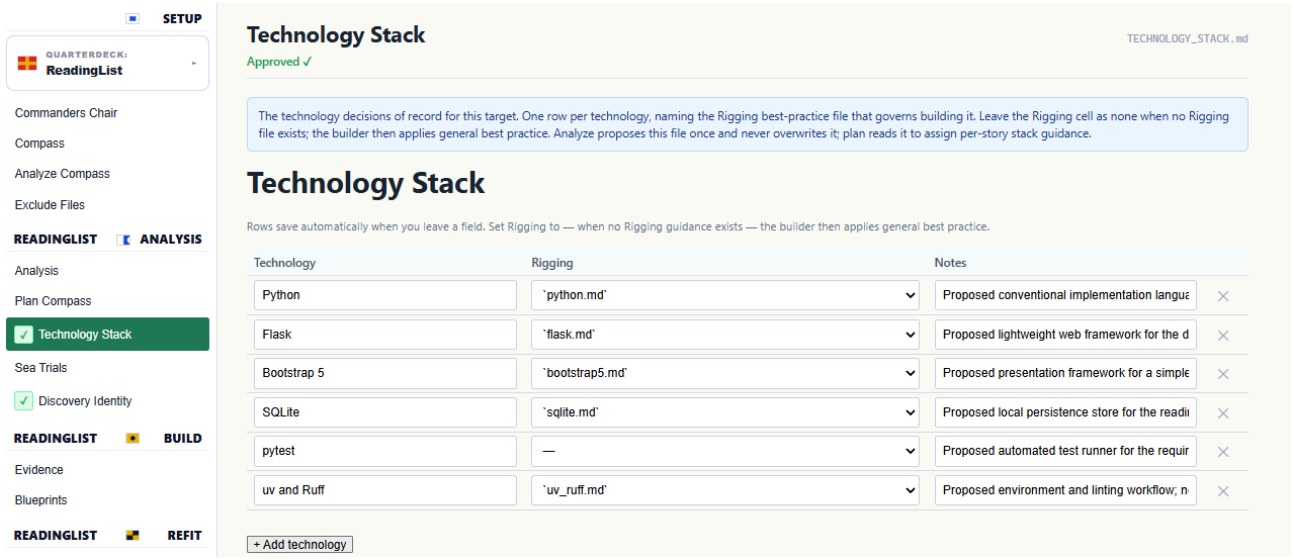
- Commander's Chair - Overview of status
- Compass - Your constitution
- Analysis - The stories and input analysis
- Sea Trials - Project acceptance criteria
- Blockers - **If this exists, you must answer**
- Questionnaires - Discovery identity and stack choices



The Commander's Chair summarizes the analyzed stories, questionnaires, and blockers.



The Analysis page shows the stories and high-level acceptance criteria created from the source material.



Review and approve the proposed technology stack in QuarterDeck before planning the build.

4. Plan

```
drydock plan ReadingList      # Create the Blueprint and Manifest.
```

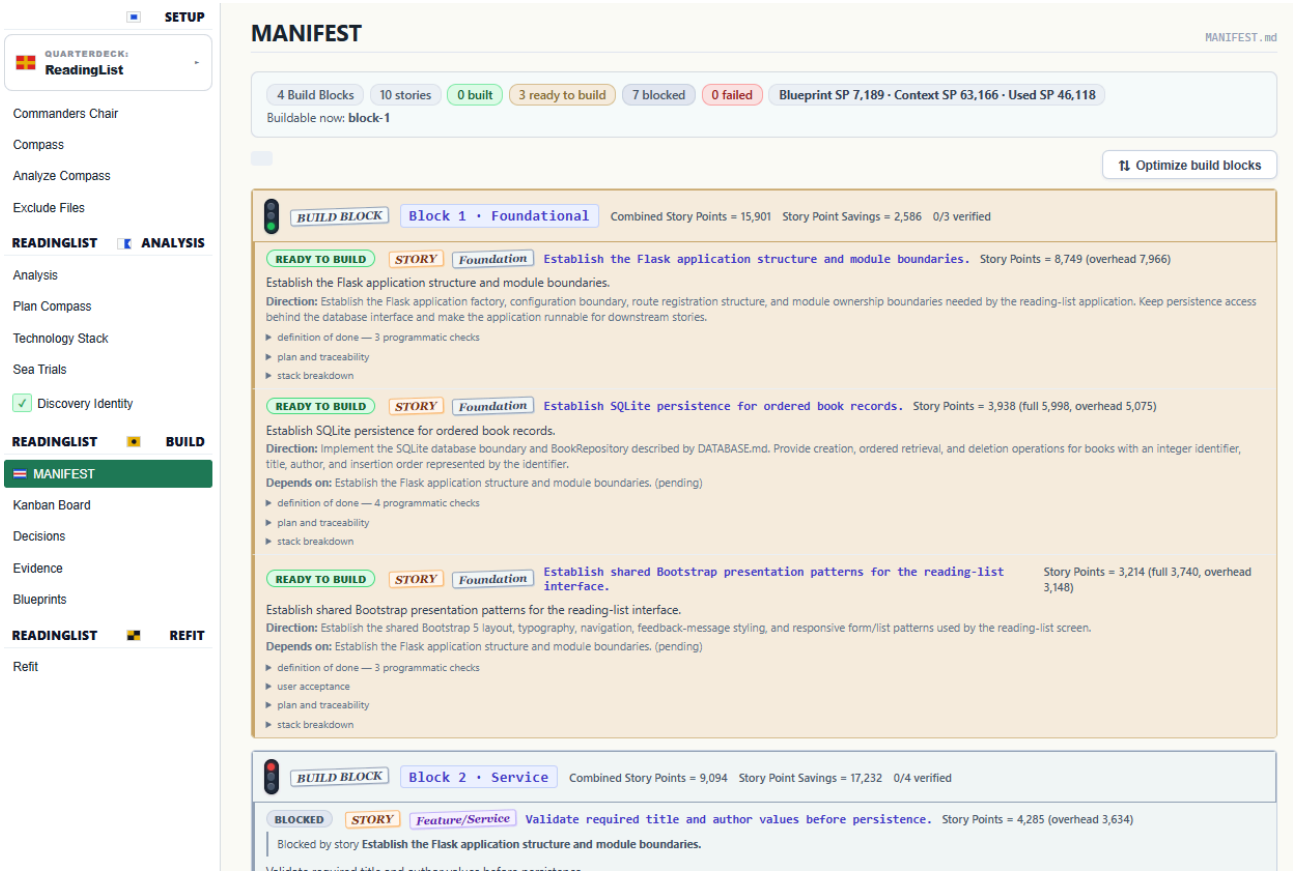
The plan or Agile Decomposition stage creates **Blueprint** files which are buildable specifications and the **Manifest** or dependency-aware build plan. The Blueprints contain test driven development assertions.

```
drydock run quarterdeck ReadingList
```

In QuarterDeck you have access to:

- The Manifest or Build Graph
- The Kanban Board
- Decisions - A controllable log of LLM decisions
- Blueprints - Your Blueprints

From QuarterDeck you can see the stages of the build and the details of each stage. Drydock breaks the build into Blocks of similar work: foundational work, persistence work, services, and service consumers such as UI screens.



The Manifest groups related stories into build blocks and shows which work is ready or blocked.

5. Build

```
drydock build ReadingList       # Build the app
```

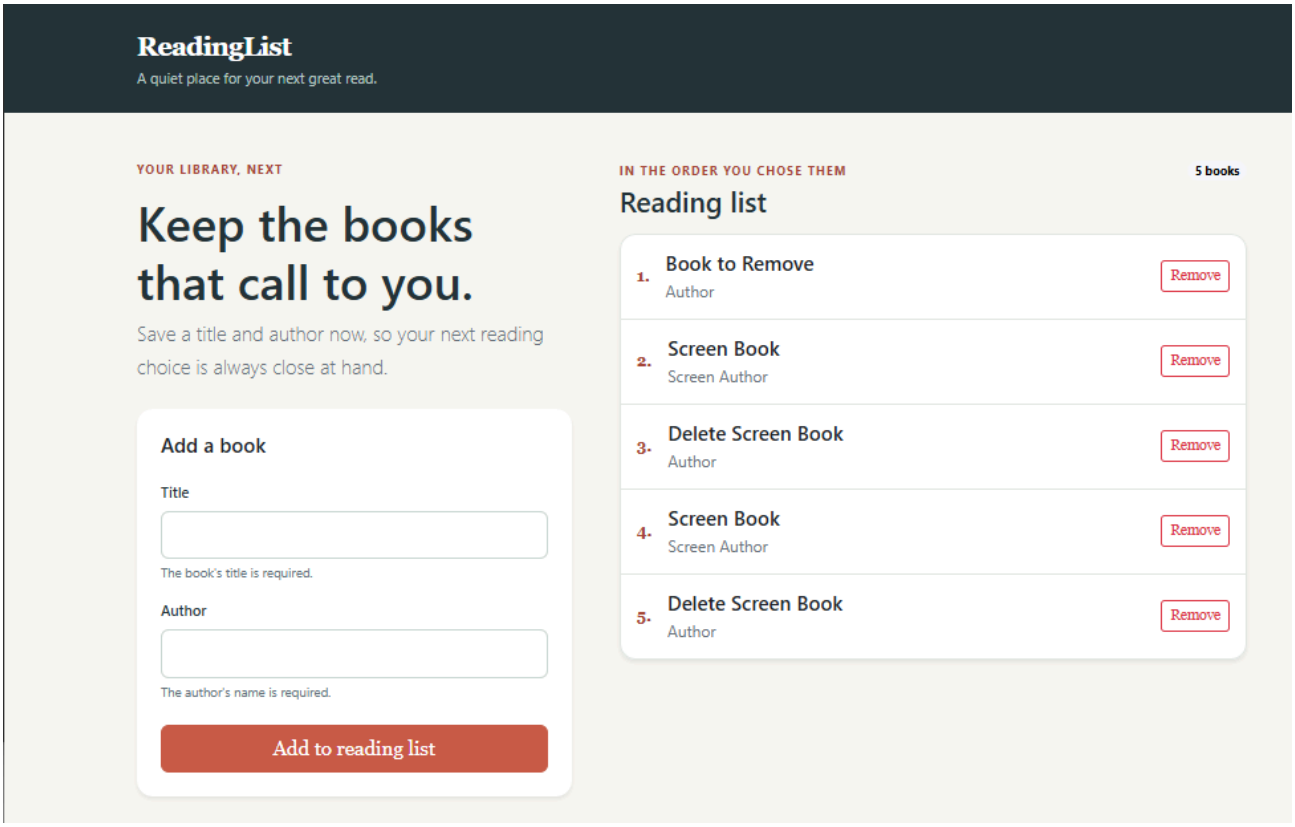
The created application is in `<drydock_build_directory>/ReadingList/`.

Run your project using the project instructions - explicit or set vi the selected technology stack. For ReadingList, run:

```
cd <drydock_build_directory>/ReadingList
uv run run.py
```

```
barlo@MSI:/mnt/c/Users/barlo/projects/ReadingList$ ls
README.md  __pycache__  app  bin  config.py  data  pyproject.toml  pytest.in
i  run.py  tests  uv.lock
barlo@MSI:/mnt/c/Users/barlo/projects/ReadingList$ uv run run.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deploym
ent. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5001
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 518-695-593
```

Follow the generated project instructions to start the application; this example runs it with `uv run run.py`.



The generated ReadingList application running after the build.

6. Refit - Changing the Application

Add the following line to `reading-list.md`:

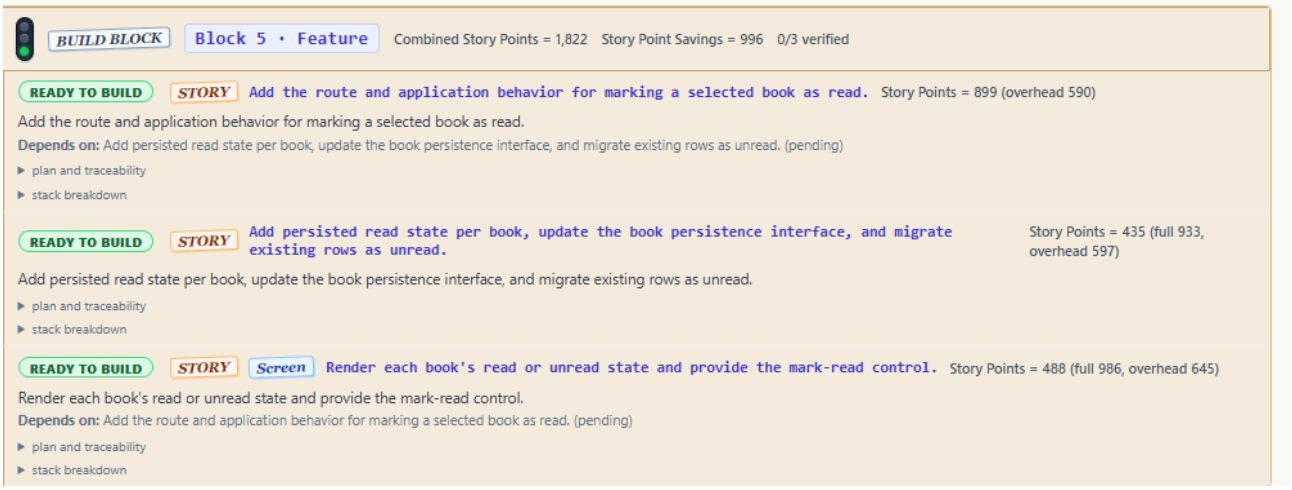
```
The reader can mark a book as read and view whether each book is unread or read.
```

Re-import the source, generate Refit Tickets, and run the incremental build:

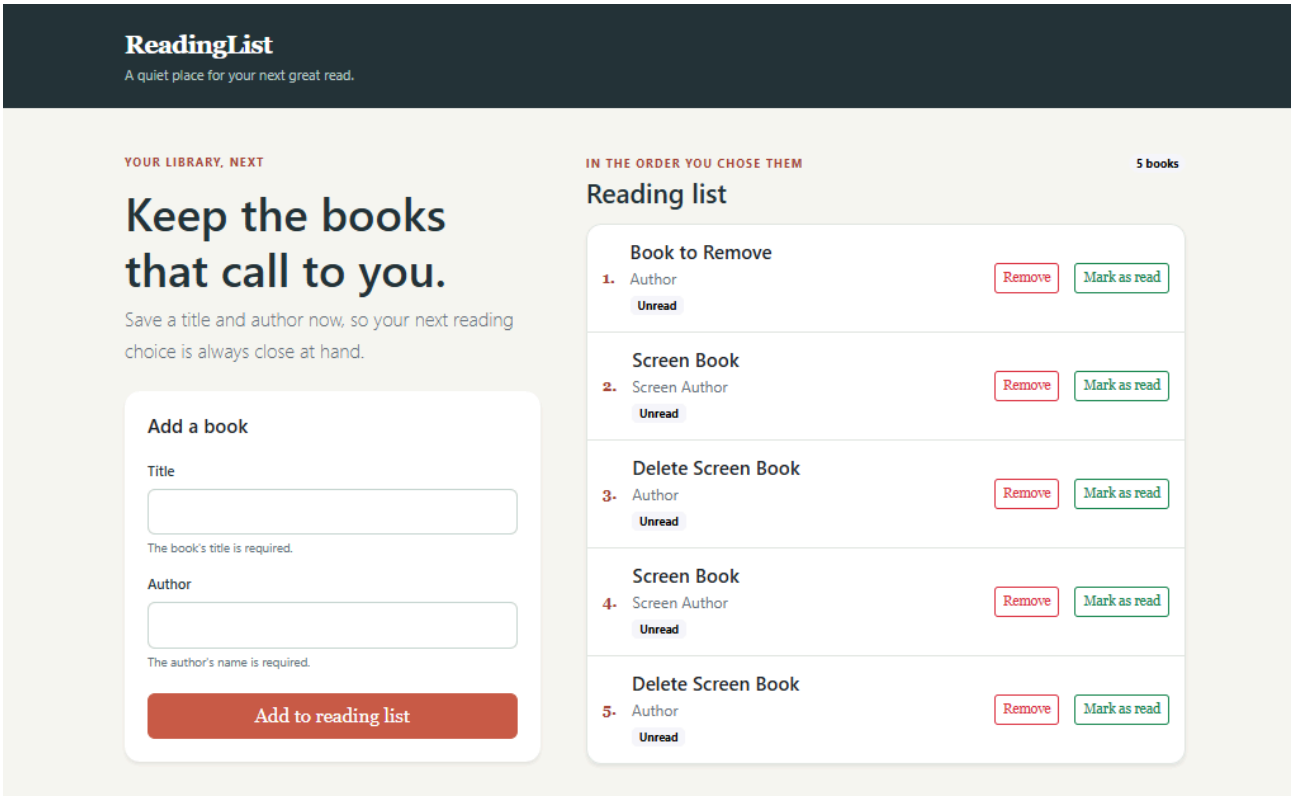
```
drydock import ReadingList --update  # Re-import your updated specs
drydock refit ReadingList --sources  # Create Refit Tickets, Update the Manifest
drydock build ReadingList            # Incremental Build
```

This is Drydock's development workflow, designed for high-velocity changes to specifications. `drydock refit` uses Git diff to identify source-material changes, maps source changes to Blueprints, and appends work to the build graph, enabling `drydock build` to run normally.

The post-production workflow uses a similar process but treats the internal Blueprints, rather than the user specifications, as canonical. Production change tickets map directly to Blueprints to minimize drift and make replanning mechanical without an LLM.



The updated Manifest with new stories.



The incrementally rebuilt application lets the reader mark books as read and see their status.

Links

- Project Home Page
- GitHub
- User Installation Guide
- Drydock Specification
