🦠 `adrscan` – Rust CLI Design

✨ Binaries

Binary	Description
`adrscan`	Native binary via `cargo install adrscan`
`adrscan-wasm`	WebAssembly binary with Node.js bindings via `wasm-bindgen` for npm usage

```
# Native install
cargo install adrscan

# Node.js-compatible WASM
npx create-adr-module
```

🦏 Commands

All commands output **JSON by default**, with optional `--format markdown` or `--format table`.

`adrscan init`

Initializes ADR workspace.

```
adrscan init
```

Creates:

- `./adr/`
- `adr-template.md`
- `adr/ADR-0000-Initial-State.md`

`adrscan inventory`

Scans repository and creates a content hash snapshot.

```
adrscan inventory --output .adrscan/inventory-latest.json
```

`adrscan diff`

Compares current repo state to last inventory snapshot.

```
adrscan diff --baseline .adrscan/inventory-latest.json
```

`adrscan propose`

Drafts a new ADR summarizing detected changes.

```
adrscan propose --diff .adrscan/diff-latest.json
```

`adrscan index`

Regenerates `adr/index.md` from parsed ADRs.

```
adrscan index --format markdown
```

`adrscan validate`

Lints ADR files for metadata, formatting, and semantic consistency.

```
adrscan validate
```

Checks for:

- Missing metadata
- Duplicate IDs
- Invalid links
- Non-conforming structure

Returns JSON array of issues.

🐥ADR Metadata Schema (YAML Frontmatter or JSON)

```yaml
id: "ADR-0005"
title: "Use SQLite for Local Persistence"
status: "proposed"  # ["proposed", "accepted", "rejected", "deprecated"]
date: "2025-07-17"
deciders:
  - "Trevor Bowman"
  - "AI Assistant"
supersedes: ["ADR-0002"]
tags: ["persistence", "database", "local"]
```

ADR documents can use:

- YAML frontmatter (`---` fenced block)
- Or top-level JSON block in comments

GitHub Action Integration

`.github/workflows/adrscan.yml`

```yaml
name: ADR Drift & Lint

on:
  push:
    paths:
      - '**/*.rs'
      - '**/*.md'
      - '!adr/index.md'
  schedule:
    - cron: '0 3 * * 1'  # Weekly scan

jobs:
  adrscan:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: Install adrscan
        run: cargo install adrscan

      - name: Run inventory and diff
        run: |
          adrscan inventory --output .adrscan/inventory.json
```

```
          adrscan diff --baseline .adrscan/inventory.json > .adrscan/diff.json

      - name: Propose ADR if changes found
        run: |
          adrscan propose --diff .adrscan/diff.json

      - name: Lint ADRs
        run: |
          adrscan validate > .adrscan/lint-results.json

      - name: Upload report
        uses: actions/upload-artifact@v4
        with:
          name: adrscan-results
          path: .adrscan/
```

🍀 Directory Structure (Post-init)

```
repo/
├── adr/
│   ├── ADR-0000-Initial-State.md
│   └── index.md
├── adr-template.md
└── .adrscan/
    ├── inventory-<timestamp>.json
    ├── diff-<timestamp>.json
    └── lint-results.json
```

🌳 Functional Requirements

- Initialize ADR environment with boilerplate and templates
- Generate and store an inventory hash of the repository content
- Compare current state to previous snapshot to detect drift
- Generate delta-based ADR proposals from detected drift
- Build and update an `index.md` referencing all known ADRs
- Lint and validate ADR files for metadata completeness and formatting
- Output all command results in JSON by default
- Provide Node.js-compatible WASM bindings for CI/CD use
- Integrate seamlessly into GitHub Actions workflows
- Support markdown and table output formatting via flags

🦝 Non-Functional Requirements

- Built using safe, idiomatic Rust with `clap`, `serde`, `walkdir`, `ignore`
- Produce deterministic, reproducible outputs from scans and diffs
- Support CI/CD with unit, integration, and regression test coverage
- Maintain a small binary size and minimal memory footprint for CLI and WASM
- Support multi-platform operation (Linux, macOS, Windows)
- Cross-compile cleanly to `wasm32-unknown-unknown` with minimal external dependencies
- Follow SemVer for release versioning and changelogs
- Emit clear, actionable errors for invalid input or repository state
- Enable offline operation and caching for repeated scans
- Use SPDX identifiers and include license metadata in distributed packages

---

🦅 Documentation

🐞 Self-Training Guide

````
# Self-Training: adrscan

## Objective
Learn how to use `adrscan` to manage and automate Architectural Decision
Records.

## Step-by-Step

1. **Install the CLI**
   ```bash
   cargo install adrscan
````

1. **Initialize ADR Workspace**

```
adrscan init
```

This creates the `adr/` folder and a base template.

1. **Run a Repository Inventory**

```
adrscan inventory --output .adrscan/inventory.json
```

1. **Detect Drift**
```

```
adrscan diff --baseline .adrscan/inventory.json
```

1. **Propose a New ADR**

```
adrscan propose --diff .adrscan/diff.json
```

1. **Rebuild ADR Index**

```
adrscan index
```

1. **Validate ADRs**

```
adrscan validate
```

🐥User Guide (Markdown)

````
# User Guide: adrscan

## Overview
`adrscan` helps track architectural decisions and detect drift in software
projects.

## Available Commands
- `adrscan init` – Bootstrap ADR structure
- `adrscan inventory` – Hash and record file inventory
- `adrscan diff` – Compare current repo to last snapshot
- `adrscan propose` – Draft ADRs for identified changes
- `adrscan index` – Regenerate `adr/index.md`
- `adrscan validate` – Lint ADRs for consistency

## Metadata Standard
Use frontmatter YAML:
```yaml
id: ADR-XXXX
title: Reasoned Decision
status: proposed
date: YYYY-MM-DD
deciders:
  - "Name"
supersedes: []
tags: []
````
````

Output Formats

- Default: `--format json`
- Optional: `--format markdown`, `--format table`

CI Integration

See `.github/workflows/adrscan.yml` for example.

```
---

## 🌍 Future Roadmap

| Feature                     |
Description                                                                  |
| --------------------------- |
------------------------------------------------------------------------- |
| 🦜 External ADR Linking     | Reference upstream ADR repositories via Git
submodules or `adrscan link` |
| 🦂 SBOM Integration          | Annotate ADRs with package/component provenance
using SPDX or CycloneDX  |
|    AI Assistance            | Integrate LLMs to assist in drafting or
evaluating ADR decisions         |
| 🪴 Drift Autoclassification | Classify diffs as doc/code/config/test to
prioritize proposals           |
| 🪴 Test Impact Tracing      | Cross-reference ADR decisions with affected test
cases                   |
|    Time-based Drift Analysis | Detect stale ADRs based on repo evolution over
time                       |
| 🦘 ADR Search Indexing      | Enable tag- and text-based querying of existing
ADRs                      |
| 🕳️ Markdown-to-JSON Bridge  | Extract and convert markdown ADRs into
normalized JSON docs for export   |
```