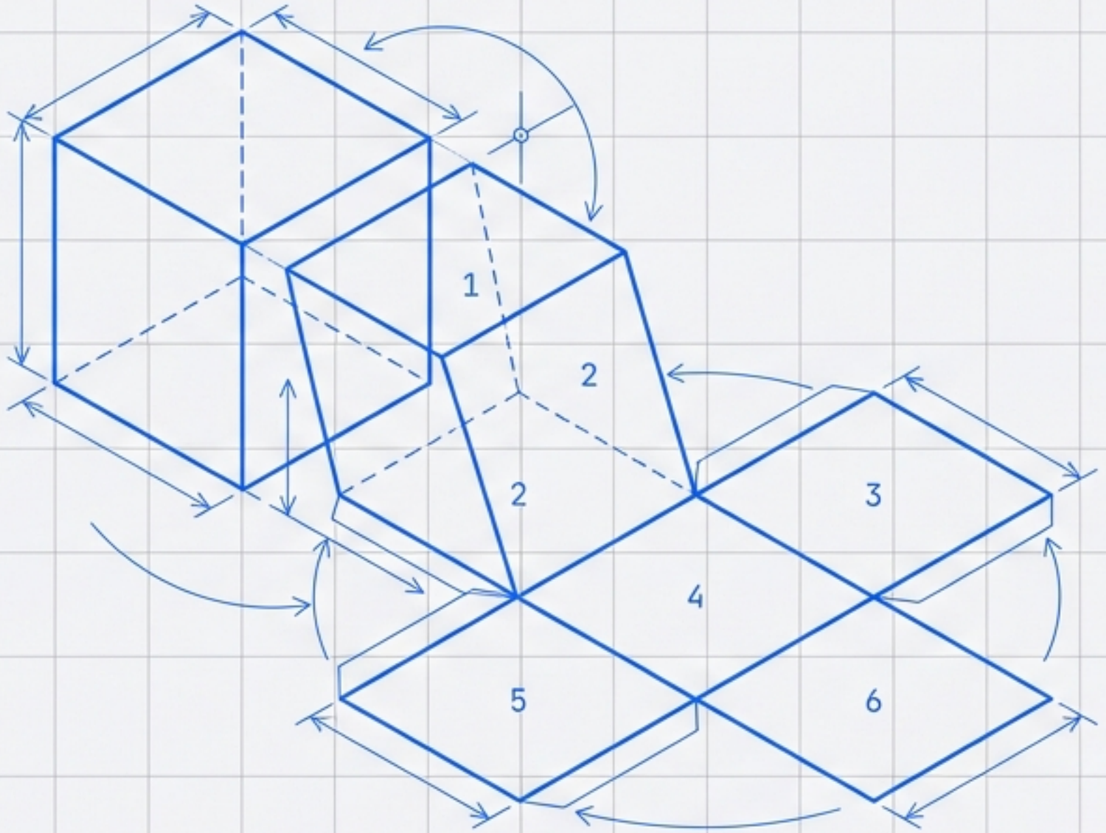
DRAFT: CONTEXT-DRIVEN DEVELOPMENT

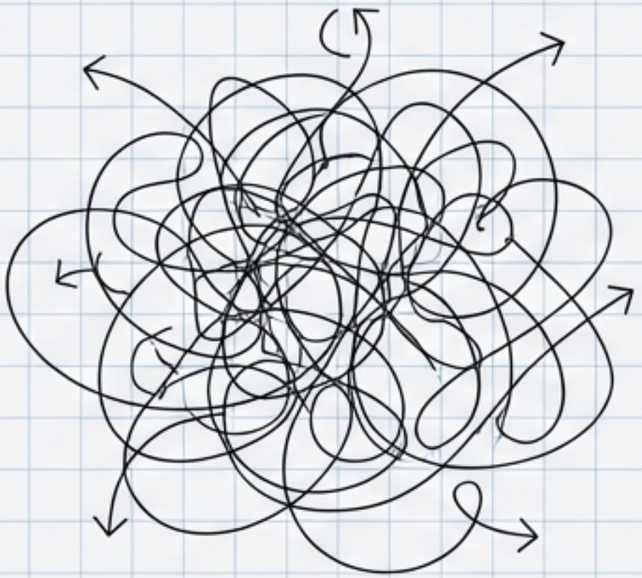
Measure twice, code once.



A methodology and plugin for Claude Code and Cursor.

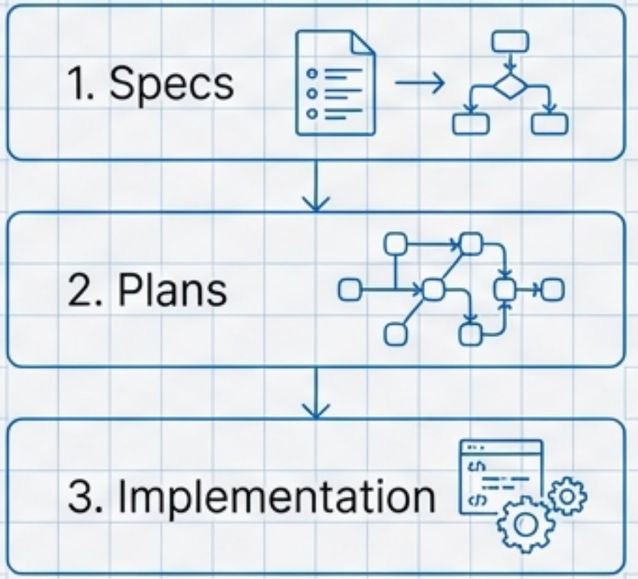
The Philosophy of Context-Driven Development

The Old Way: Unstructured Chaos



AI coding often suffers from loss of context and 'hallucinated' logic. Without a plan, the output is unpredictable.

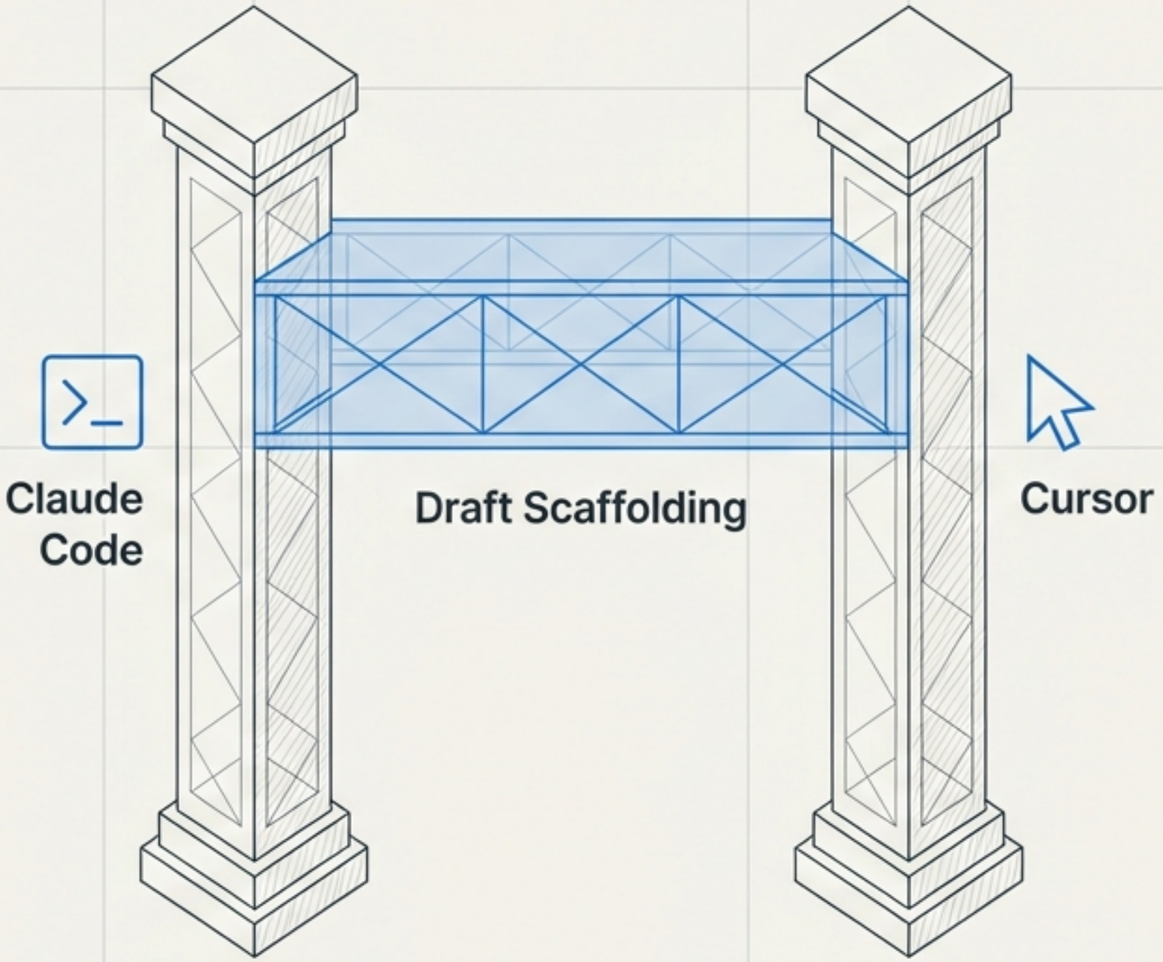
The Draft Way: Structured Engineering



Draft imposes structure. We define requirements (Specs), break them down (Plans), and only then execute structured workflows for features and fixes.

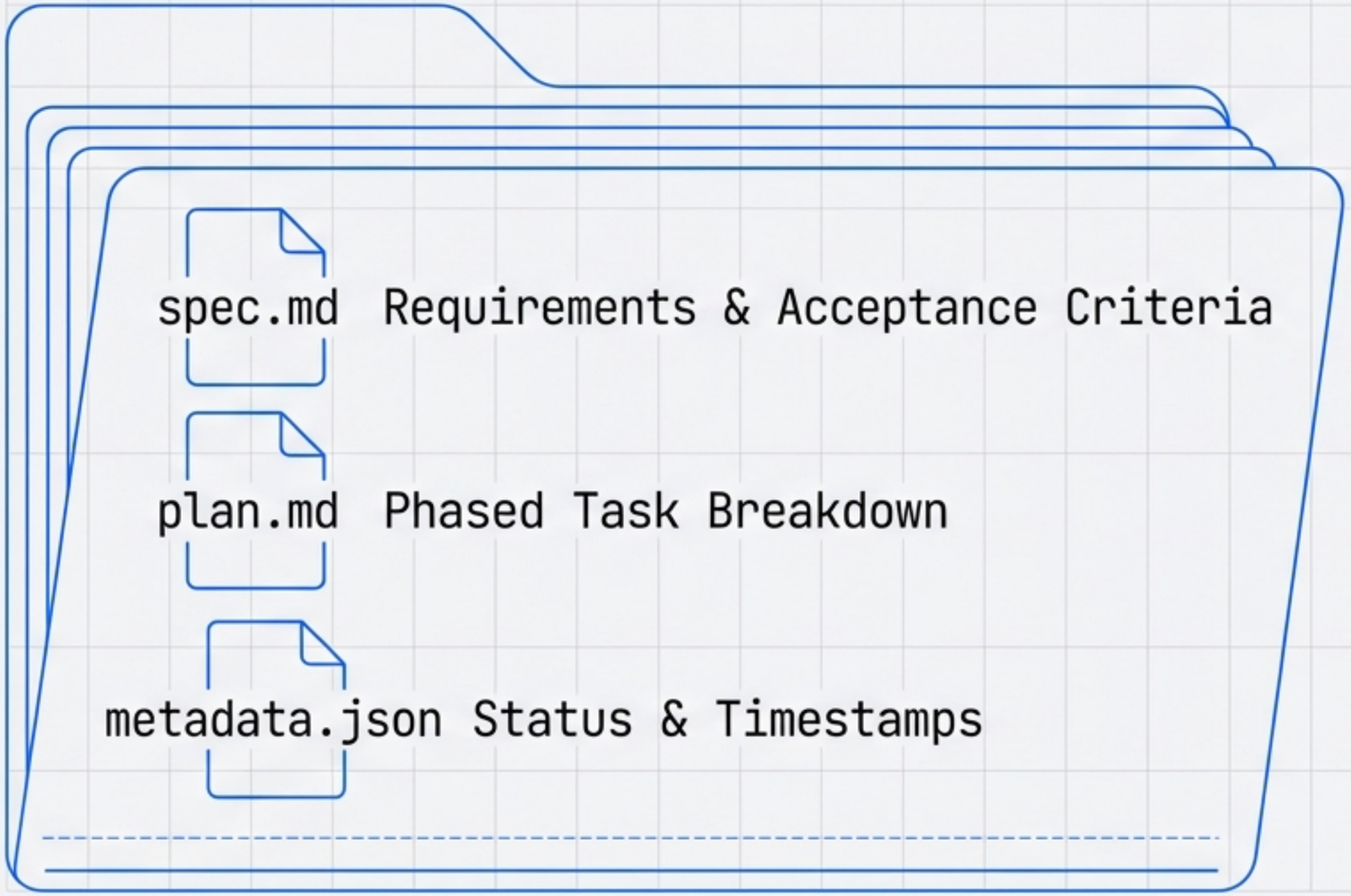
The Scaffolding for Intelligent Coding

Draft is not a standalone IDE. It is a plugin that brings Context-Driven Development (CDD) to your existing workflow, acting as the structural scaffolding for powerful AI models.



Core Concept: The "Track"

A Track is the fundamental unit of work—whether it's a feature, a bug fix, or a refactor.



Status Legend

[ ] Pending/New

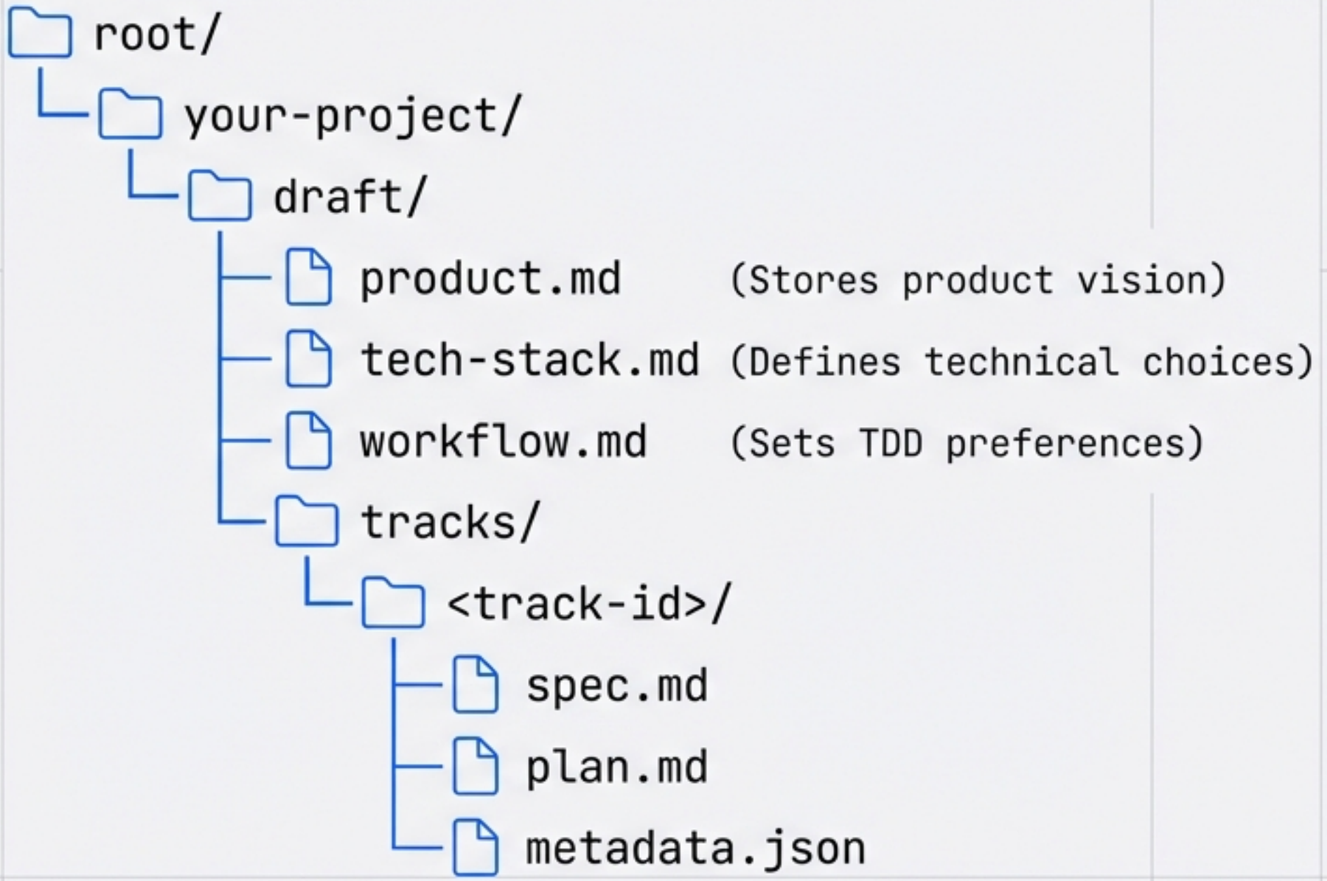
[~] In Progress

[x] Completed

[!] Blocked

The Project 'Brain'

Externalizing context into a persistent structure.



> “By creating this structure, the AI always knows WHAT it is building and HOW to build it.”

Laying the Foundation: Installation

```
$ curl -fsSL
https://raw.githubusercontent.com/
mayurpise/draft/main/install.sh | bash
```

```
$ claude --plugin-dir
~/.claude/plugins/draft
```

Pro Tip:

Create an alias for convenience.

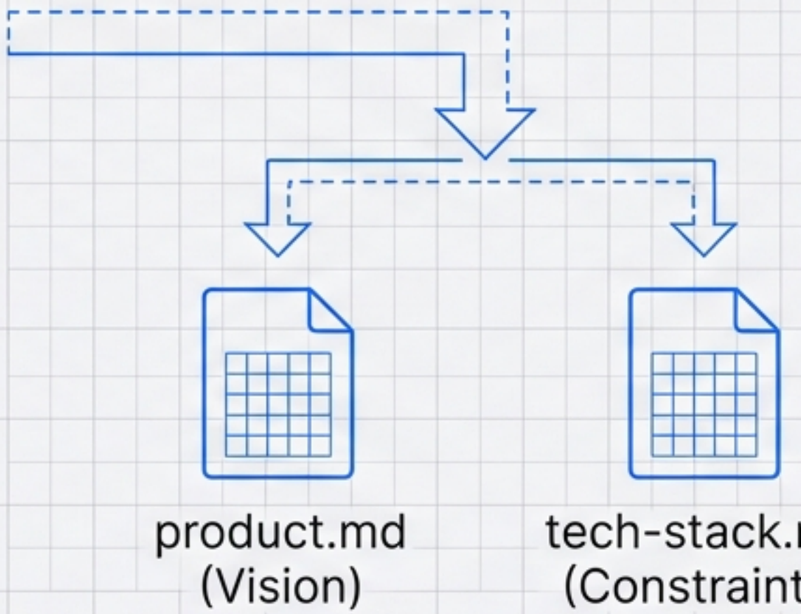
```
alias claude-draft='claude
--plugin-dir
~/.claude/plugins/draft'
```

Verify by running /draft to see command overview.

Initialization

/draft:init

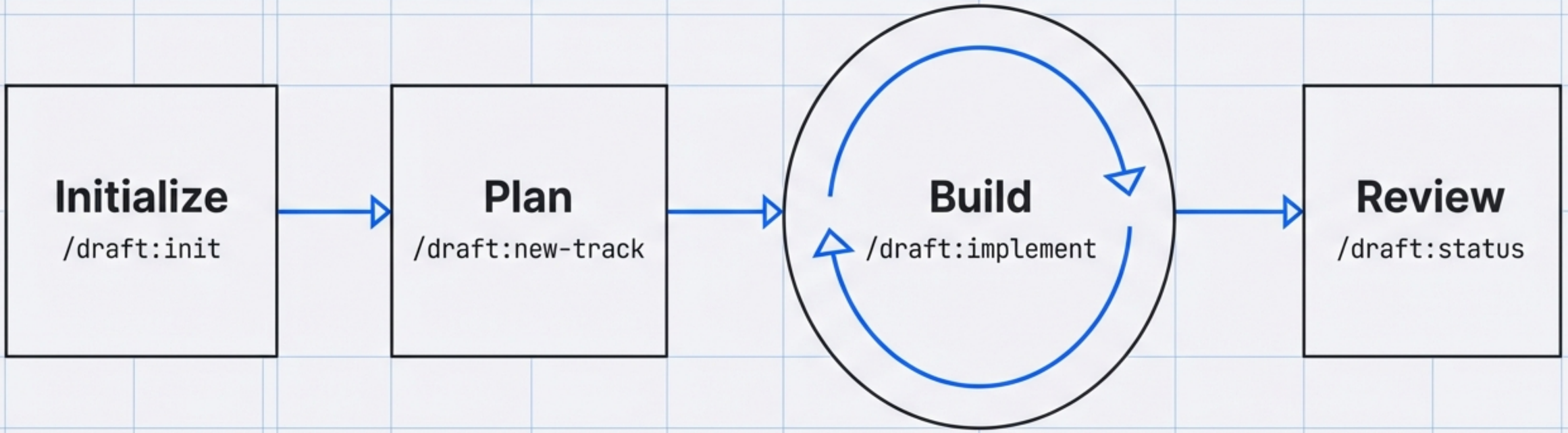
This command is run once per project to establish the context files.



Generates the **draft/** directory and prompts the user to populate the core definitions.

Ensures every subsequent line of code aligns with the project vision.

The Construction Workflow



A structured loop moving from requirements to verified code.

Step 1: Drafting the Plan

Thinking before acting.

```
> /draft:new-track ‘Add user authentication’
```

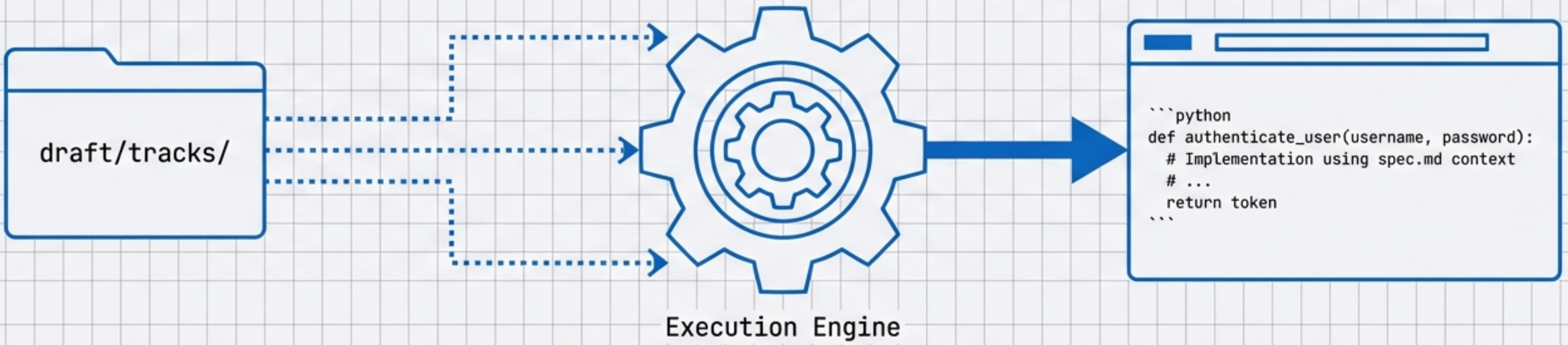
spec.md	plan.md
— Requirements derived from prompt — Context pulled from product.md	Phase 1: Database Schema Phase 2: API Endpoints Phase 3: Frontend Integration

The AI is forced to ‘measure twice’ by explicitly writing out the plan.

Step 2: Implementation

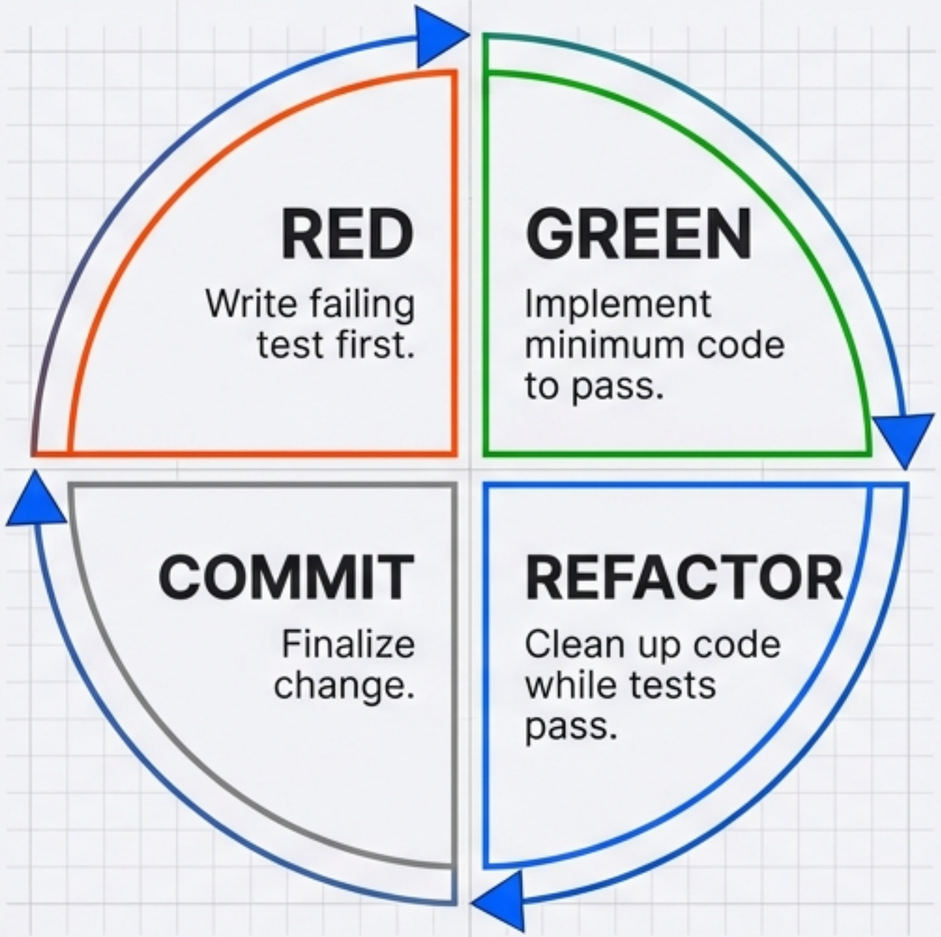
```
/draft:implement
```

1. Reads current plan.md
2. Selects next pending task
3. Executes using context from spec.md



The TDD Cycle

Enforcing Test-Driven Development via workflow.md



Prevents spaghetti code by ensuring every feature is verifiable.

Site Management & Safety

Progress Check

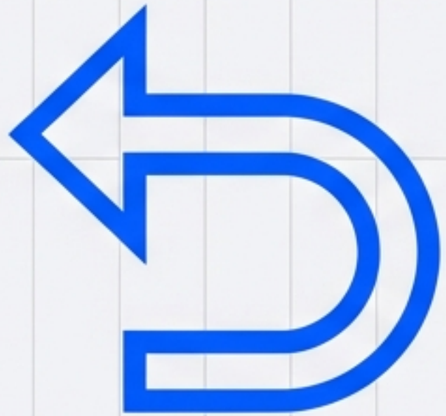
```
/draft:status
```

[x]	Task 1: Complete
[~]	Task 2: In Progress
[ ]	Task 3: Pending

Displays overview from metadata.json.

Rollback Protocol

```
/draft:revert
```



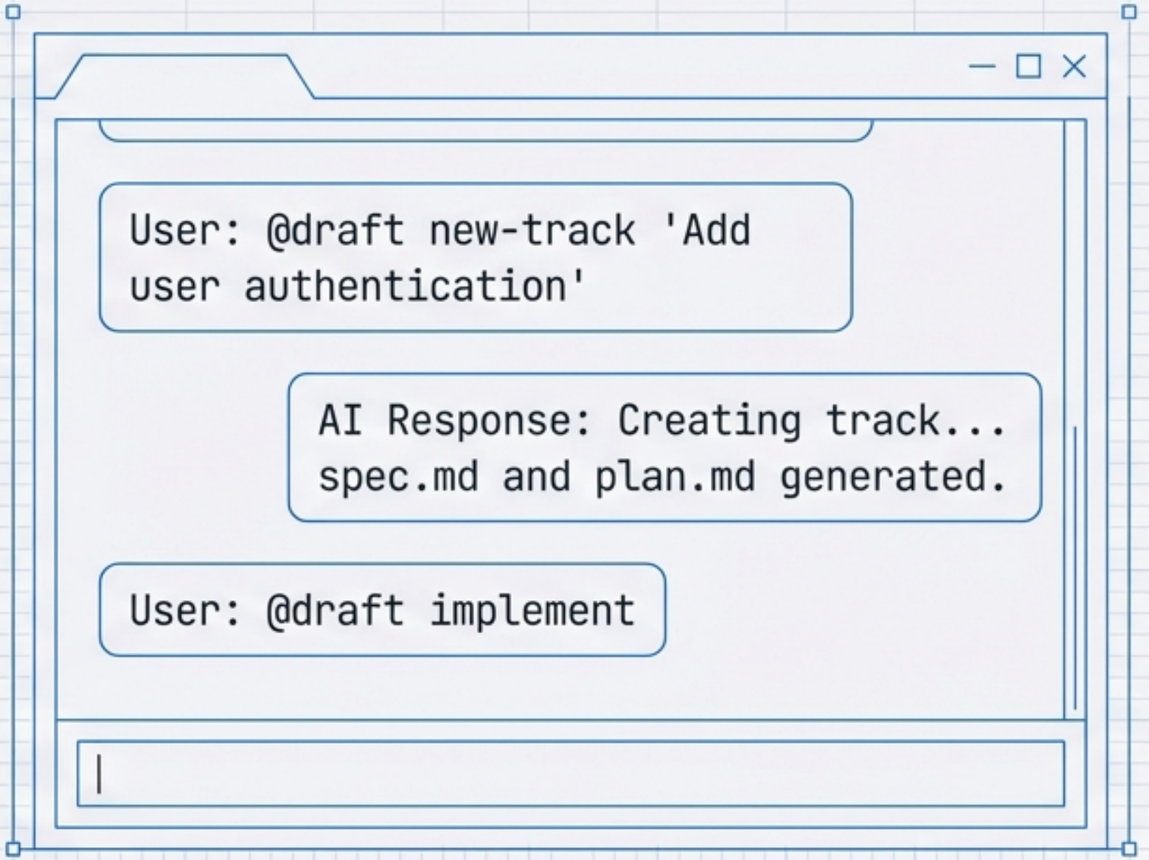
Git-aware rollback of tasks, phases, or entire tracks.

Allows you to undo changes safely if a track goes off the rails.

Integration: Cursor

Draft workflows within the Cursor IDE.

1. Copy rule file:
   `cp ../cursor/.cursorrules ~/project/`
2. Use `@draft` in chat.



Command Reference

Function	Command	Description
Setup	`/draft:init`	Initialize project context.
Planning	`/draft:new-track`	Create feature/bug track.
Execution	`/draft:implement`	Execute tasks with TDD.
Oversight	`/draft:status`	Display progress.
Safety	`/draft:revert`	Git-aware rollback.
Help	`/draft`	Show overview.

Start Drafting

Stop guessing. Start engineering.

```
git clone https://github.com/mayurpise/draft
```

License: Apache 2.0