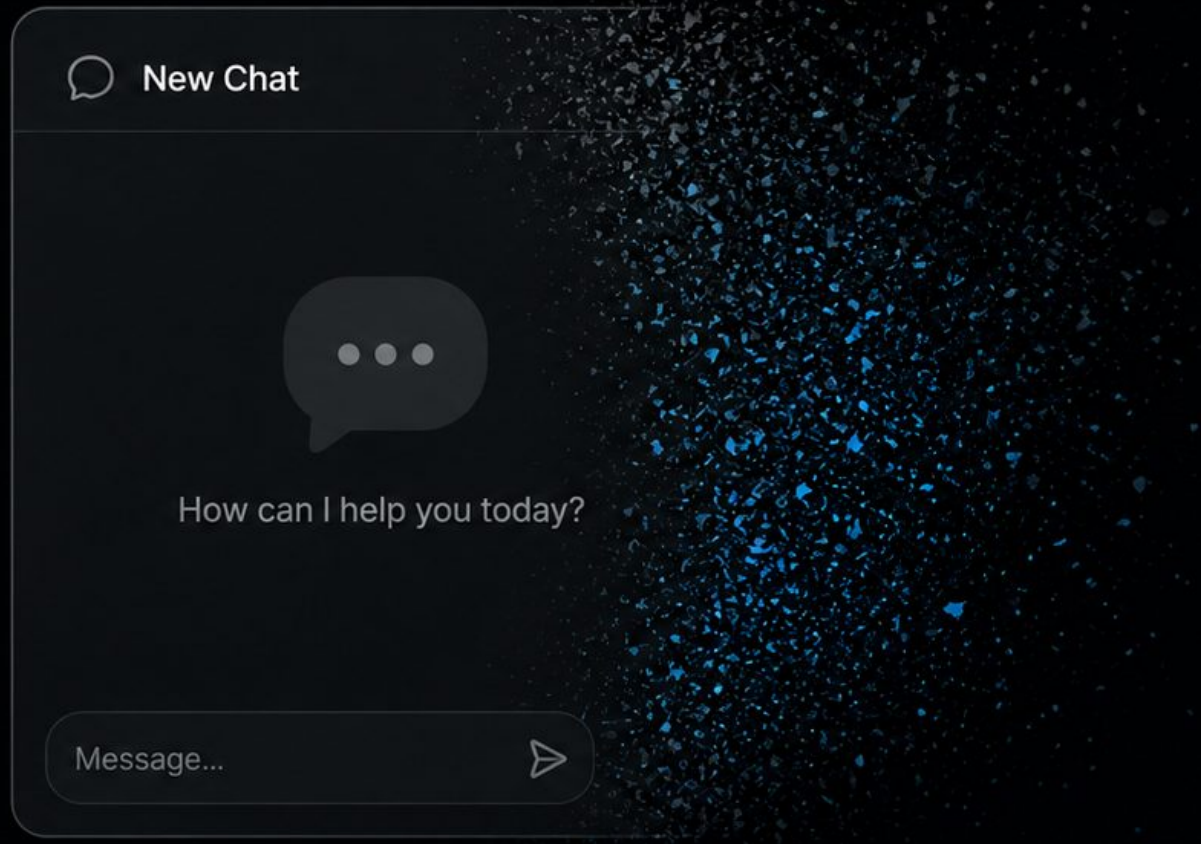
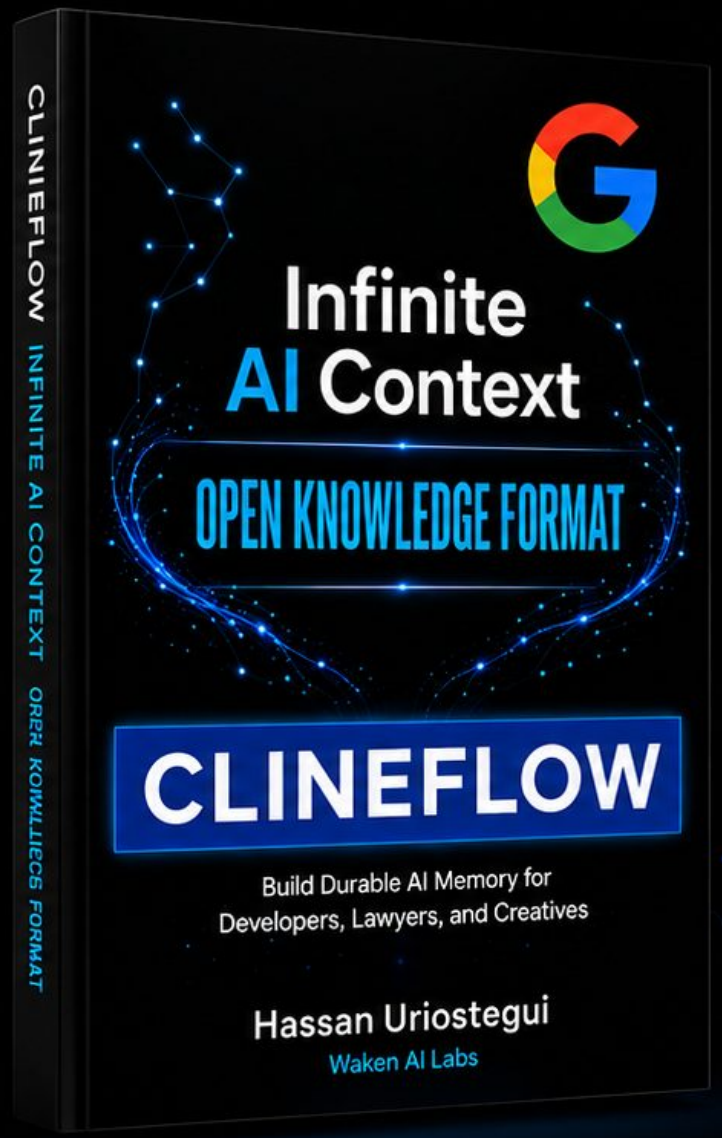
YOUR AGENT'S CONTEXT IS TEMPORARY.

Every new chat, window, or session starts with a blank slate.



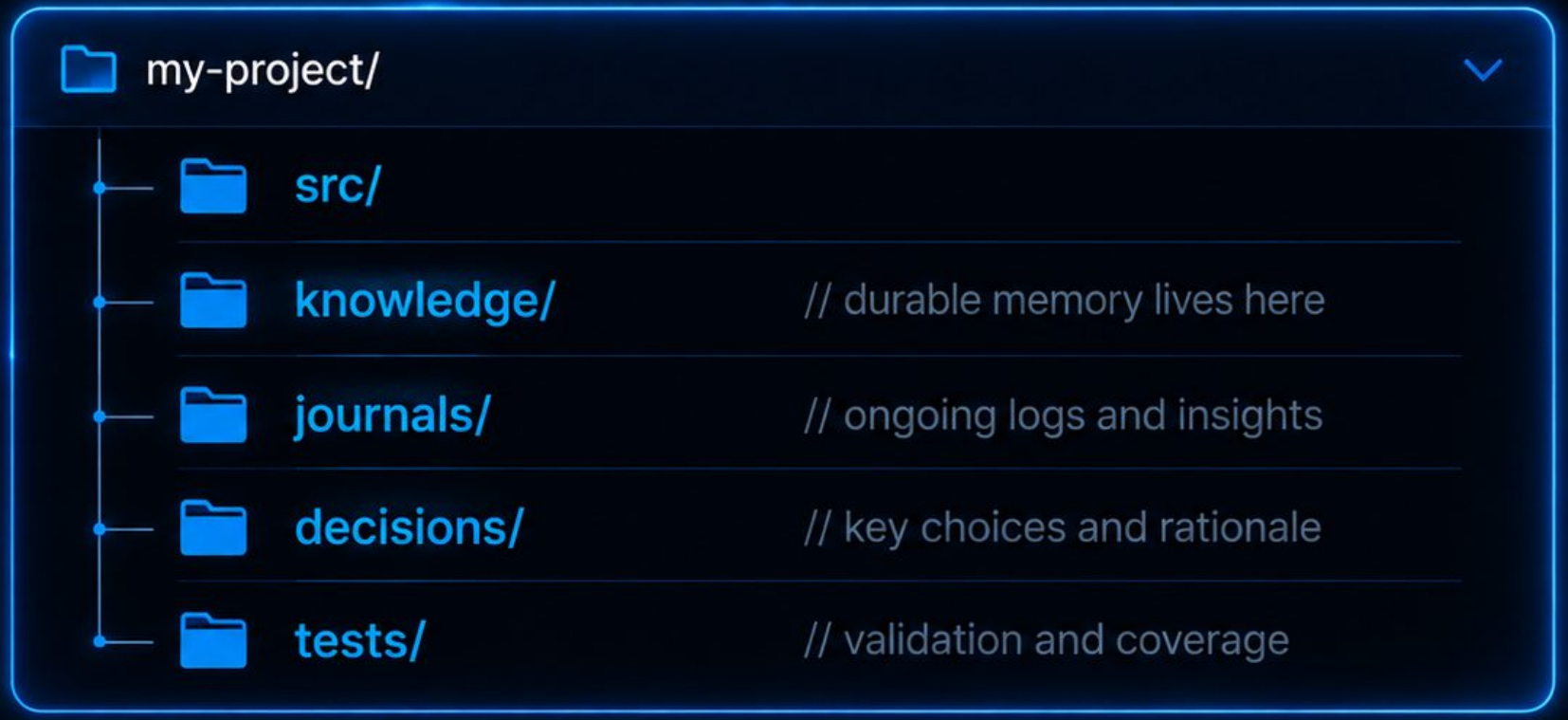


II
The Solution

www.clineflow.com

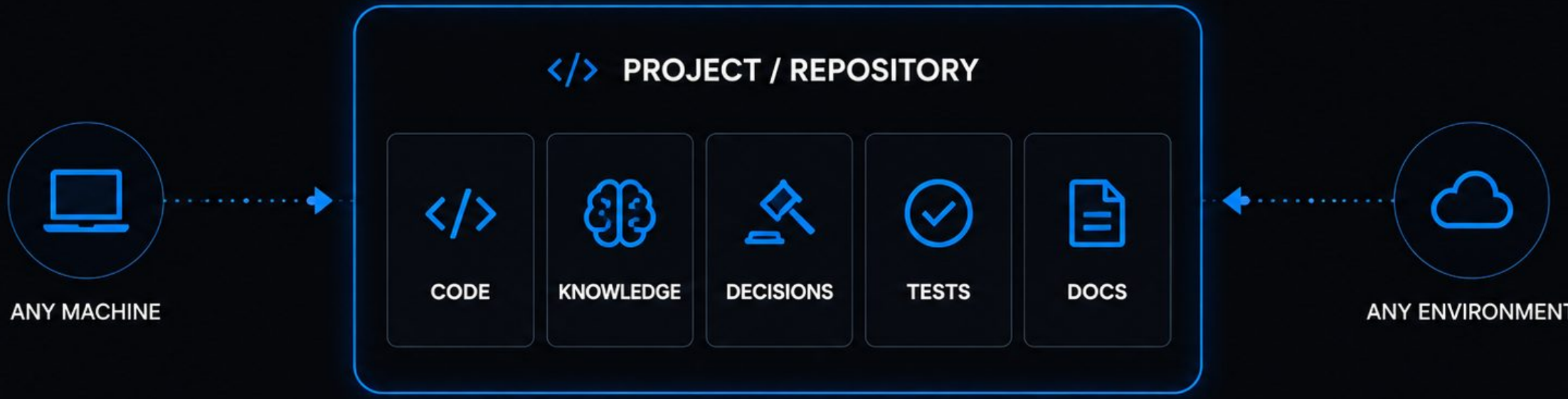
YOUR PROJECT'S CONTEXT SHOULDN'T BE.

ClineFlow writes durable memory into your project as structured, open, readable files.



CONTEXT LIVES WITH THE PROJECT.

MOVE THE PROJECT. MOVE THE KNOWLEDGE.



GIT CARRIES THE MEMORY.

Commit it. Push it. The memory travels with the project.

LOCAL MACHINE

You create context, notes, and artifacts.

GIT / REPOSITORY

Git captures the memory with the project.

REMOTE REPOSITORY

The memory is stored and travels anywhere.

ANY COMPATIBLE AGENT CAN PICK UP THE PROJECT.

Open the project. Read the knowledge. Continue with full context.

SO CAN ANOTHER DEVELOPER.

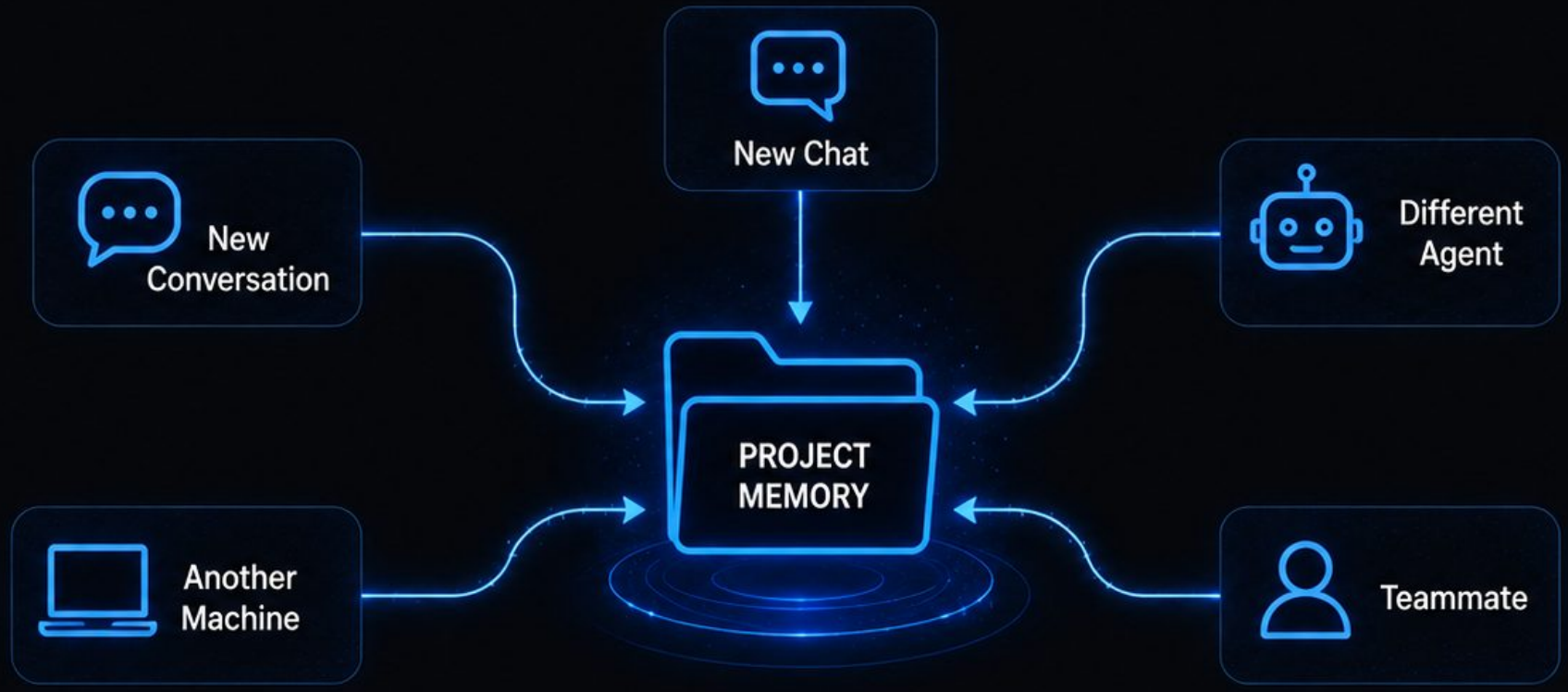
Your coworker clones the repo and gets code plus durable context together.

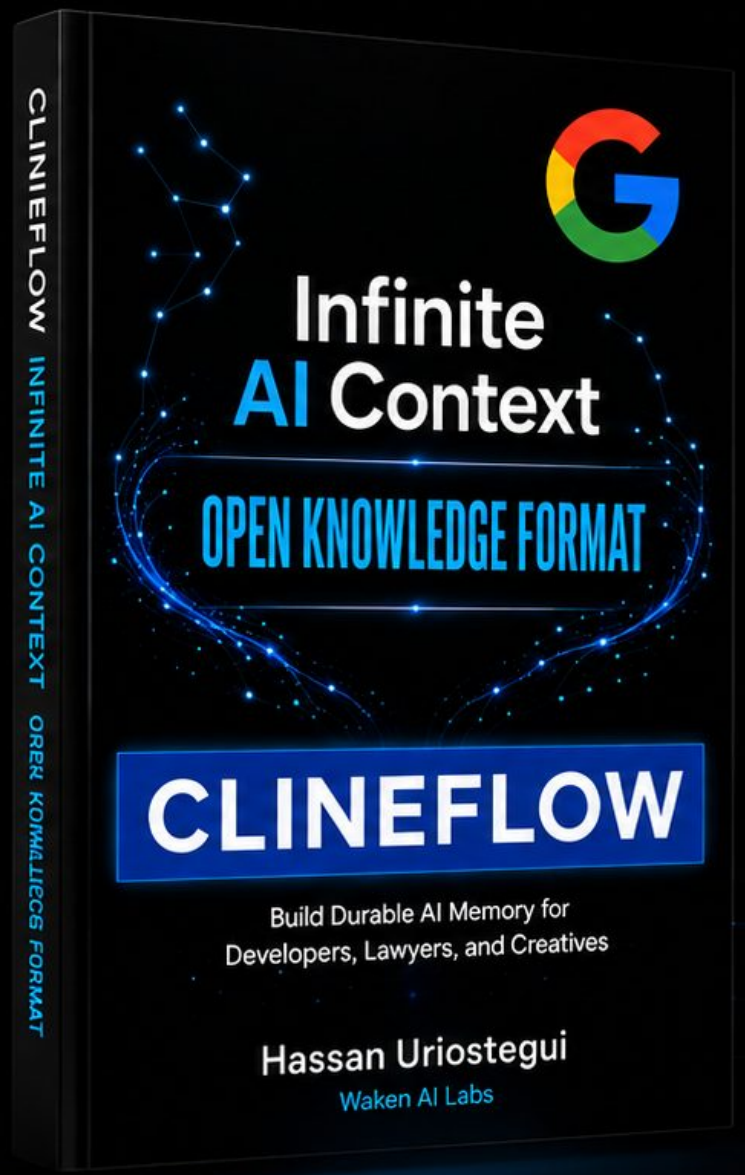
YOU
Originate the project with durable context.

REPOSITORY
Code + durable context live together.

COWORKER
Clones the repo and gets full context.

ONE PROJECT. MANY CHATS. MANY AGENTS. ONE MEMORY.



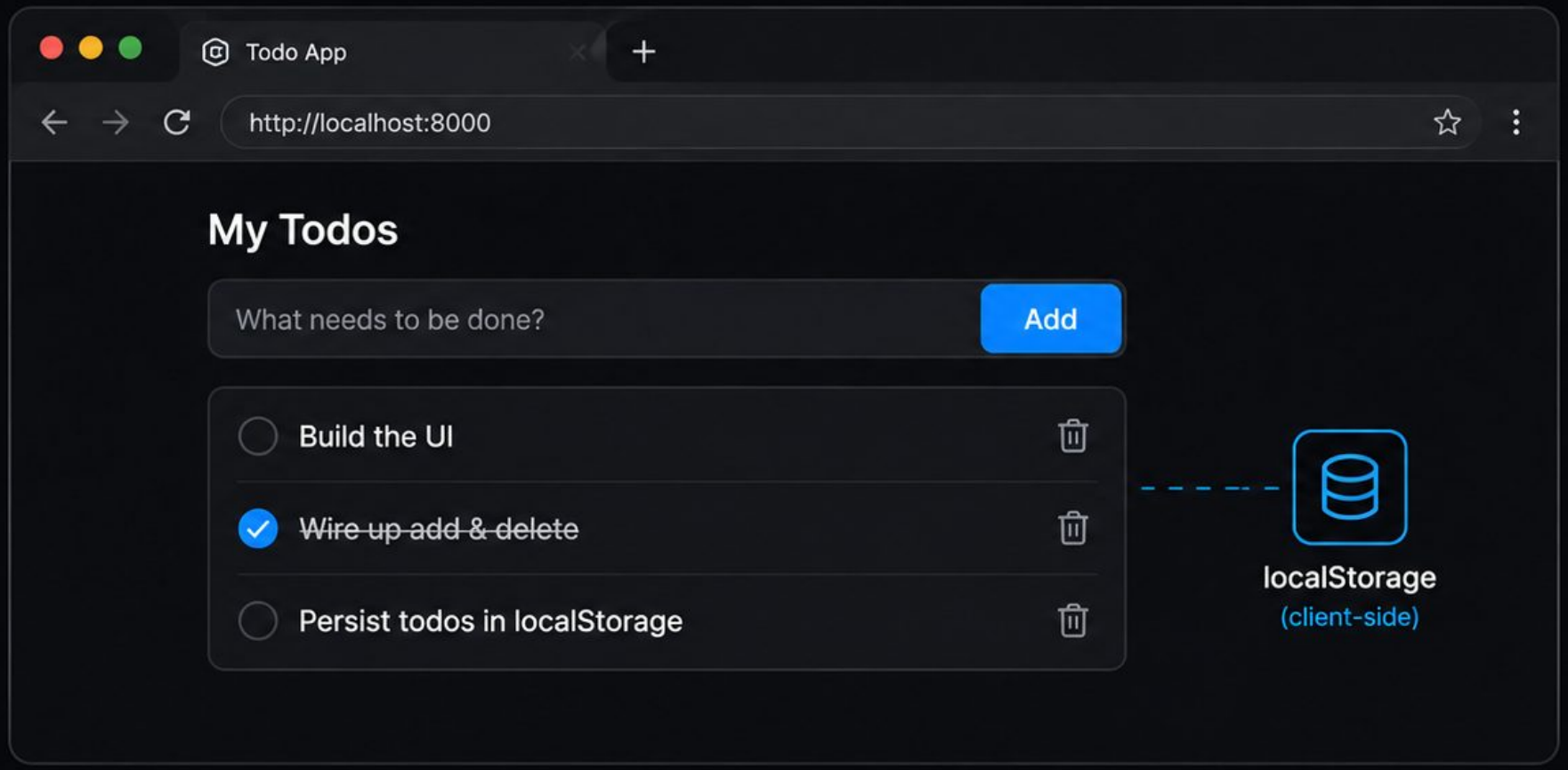


IV
The Project

www.clineflow.com

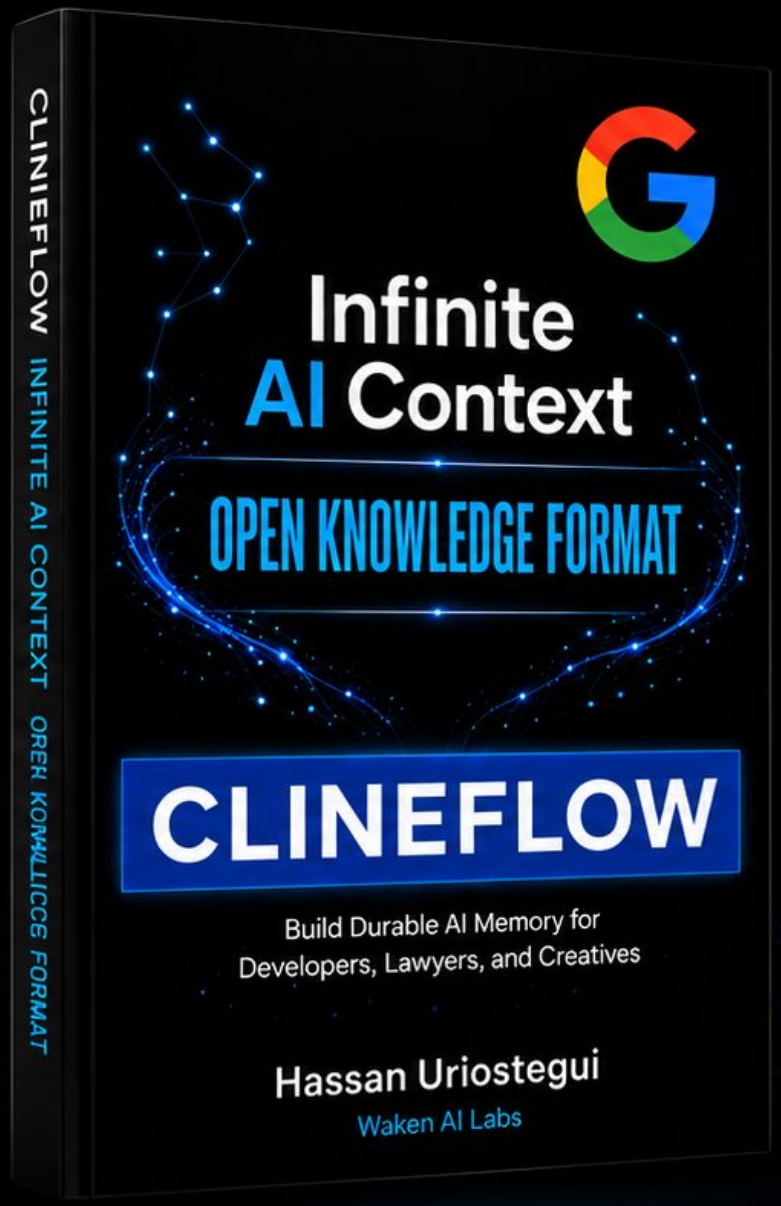
TODAY: ONE TINY TODO APP.

Plain HTML. CSS. JavaScript. No backend.



THE (INTENTIONALLY) VAGUE REQUIREMENT.

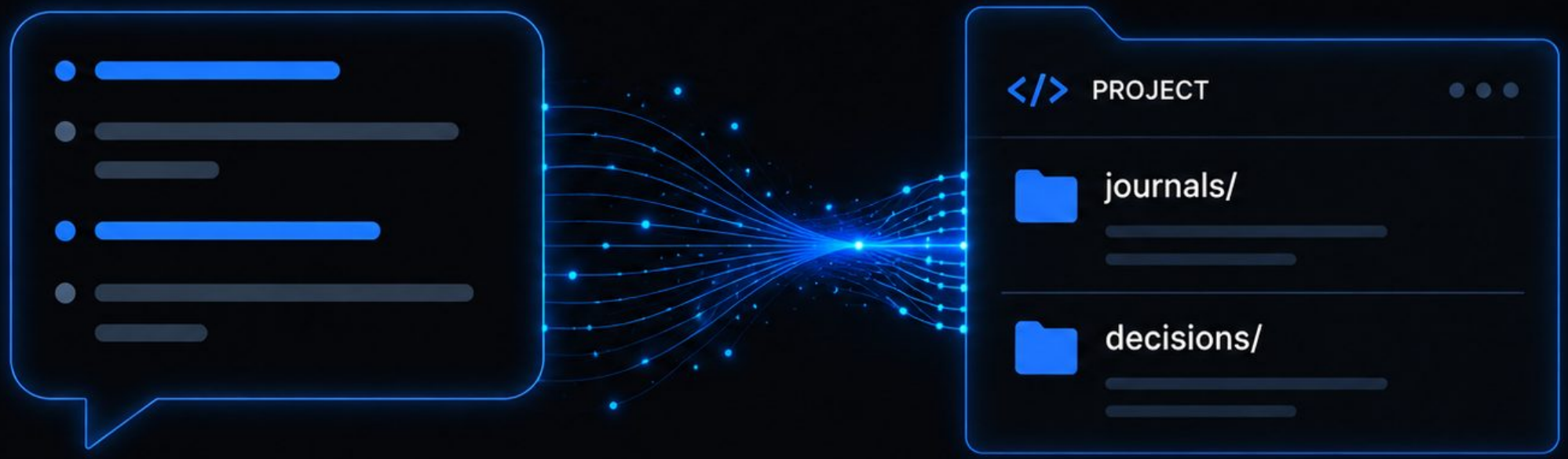
That's it. Let's see what happens.



V Grounding

www.clineflow.com

PLAN IN THE CONVERSATION. DECIDE IN THE PROJECT.



DEMO

DEMO 1 · INSTALL + GROUND THE FIRST SPEC

GOAL Show that the first execution is not product code — it is durable project context.

ON-SCREEN ACTIONS

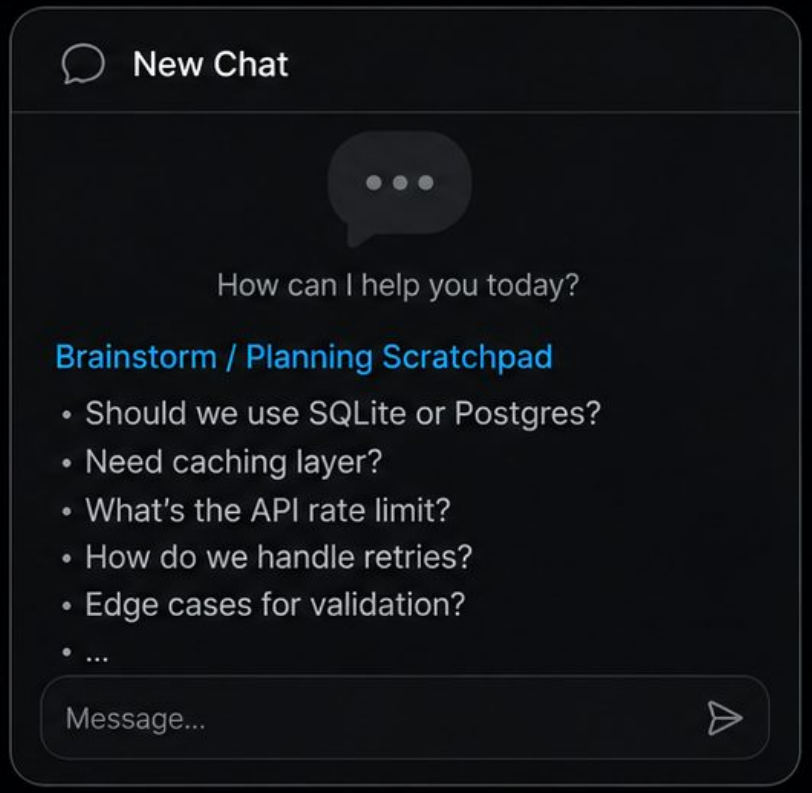
1. Start in the clean Todo project in Codex.
2. Install ClineFlow from the repository.
3. Give the deliberately vague Todo requirement.
4. Ask the agent to create/update the active journal and identify assumptions, edge cases, non-goals, and missing decisions.
5. Review the journal on screen. Correct anything you do not approve.
6. Do not implement the Todo app yet.

PROMPT TO TYPE

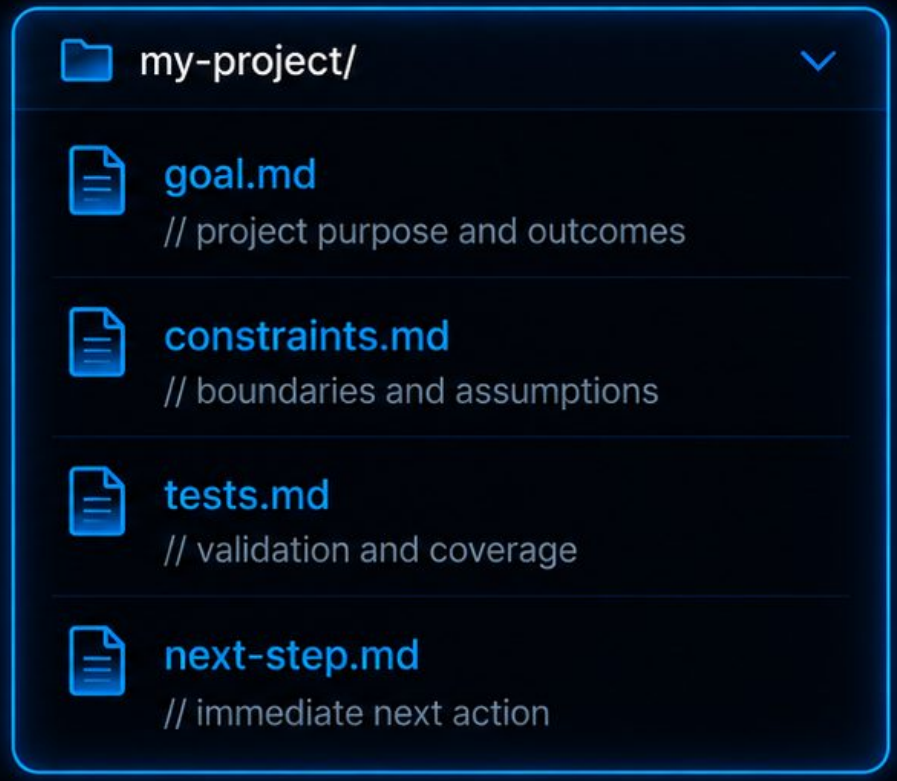
```
Please follow the installation guide at
https://github.com/hassanvfx/clineflow
and install ClineFlow in this project.

We are building a small Todo app in plain HTML, CSS and JavaScript using browser
localStorage.

Do not implement the app yet.

Start or update the active project journal and ground this requirement into a
product specification. Separate confirmed requirements, assumptions, missing
decisions, edge cases and explicit non-goals. State what you cannot infer instead
of inventing a requirement.
```

FROM SCRATCHPAD TO PROJECT MEMORY

Planning is temporary. Grounded decisions live in the project.

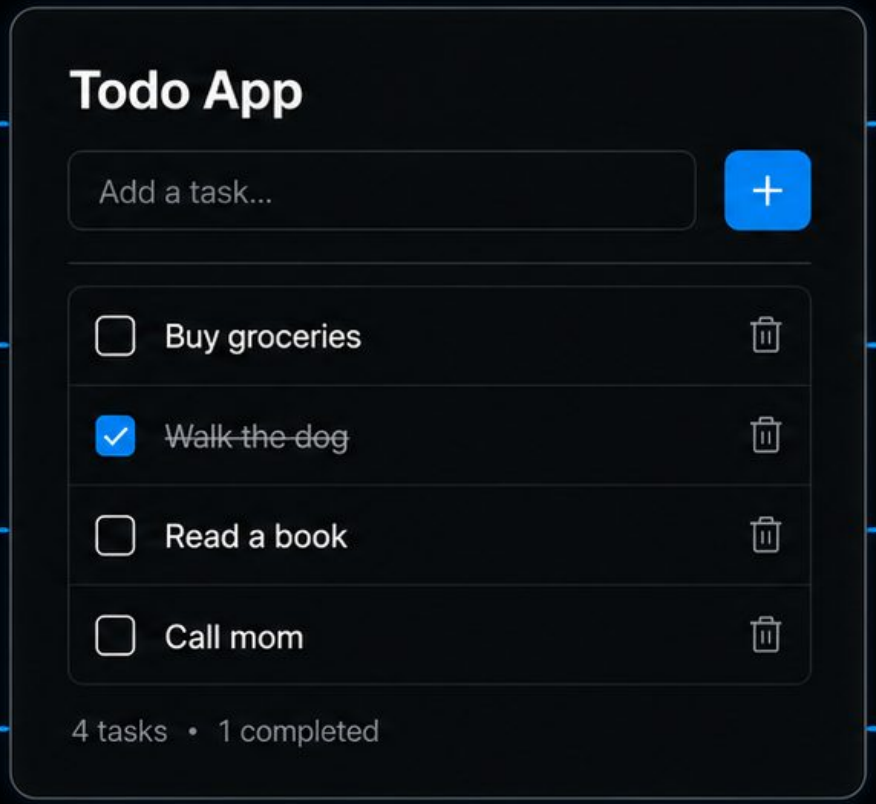


accepted decisions



WHAT DOES 'TODO APP' ACTUALLY MEAN?

1. **Blank tasks?** Can users add empty or whitespace tasks?
2. **Stable id?** Does each task have a unique, stable id?
3. **Toggle back?** Can a completed task be marked as active again?
4. **Delete one?** Can users delete a single task without affecting others?



5. **Malformed storage?** What if local storage is corrupted or invalid?
6. **No network?** Does it work fully offline?
7. **Task order?** Is the order preserved consistently?
8. **Storage key?** What is the exact key used in local storage?

GROUNDED TODO V1 CONTRACT

Task Model

- `id` string
- `title` string
- `completed` boolean

Local Boundary

- localStorage only
- no accounts
- no backend

Behavior

- add
- toggle
- delete
- reload

DEMO

DEMO 2 · REFINE THE CONTRACT + COMMIT

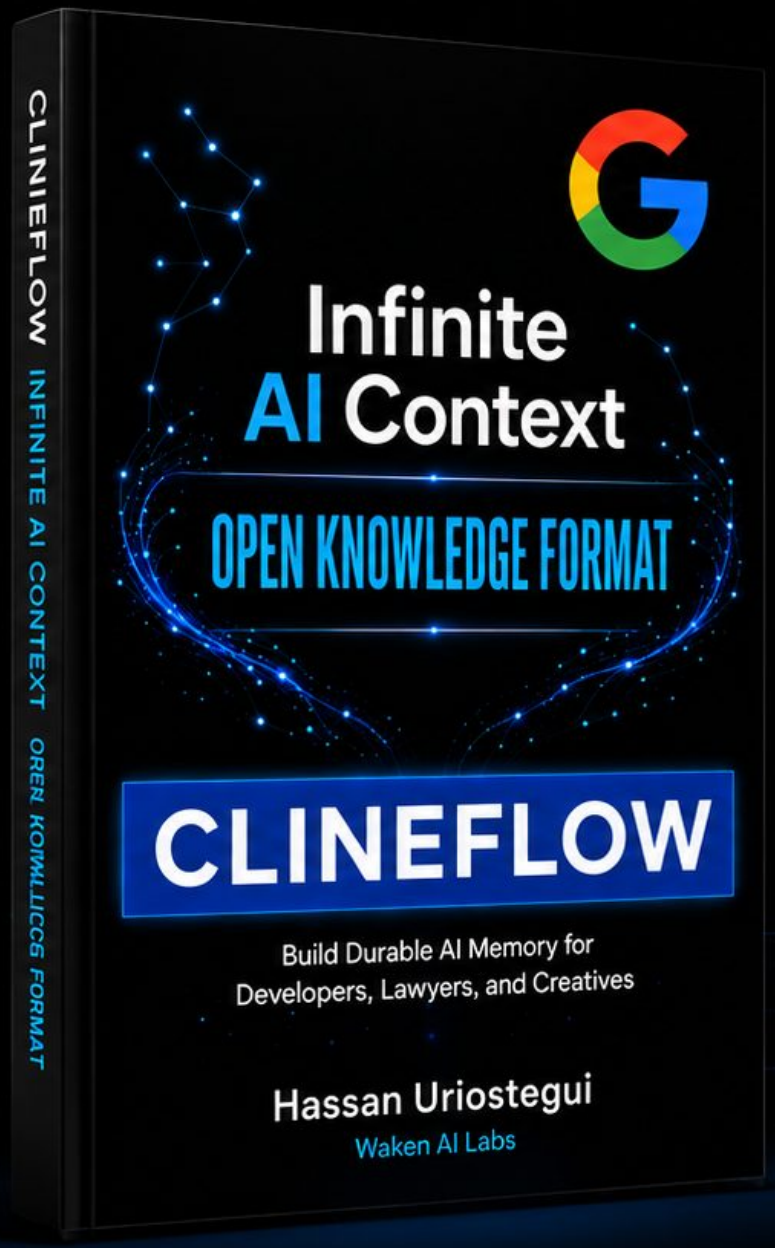
GOAL Turn the vague request into a bounded task model, persistence rule, and explicit checkpoint.

ON-SCREEN ACTIONS

1. Review stable id, non-empty title, completed boolean, create/toggle/delete, and localStorage-only boundary.
2. Confirm malformed or missing stored data must recover to an empty usable state.
3. Confirm no accounts, backend, telemetry, tags, due dates, sharing, or network task sync.
4. Ask the agent to update the journal with the accepted decisions.
5. Use the simple ClineFlow checkpoint command: “please commit”.
6. Show the resulting Git commit and journal handoff.

PROMPT TO TYPE

```
Update the active journal with the Todo v1 contract we approved.

The task model is:
- stable id
- non-empty title
- completed boolean

V1 supports create, toggle both ways, and delete one task at a time.
Task data stays only in this app's browser localStorage.
Missing or unusable stored data must recover to an empty usable list.
Do not add accounts, backend services, telemetry, due dates, tags, sharing, or
task-data network calls.

Identify any remaining ambiguity before implementation.

When the journal is correct, please commit.
```



VI Auto-Testing

www.clineflow.com

A SPEC NEEDS PROOF.

Define the evidence before the agent builds.

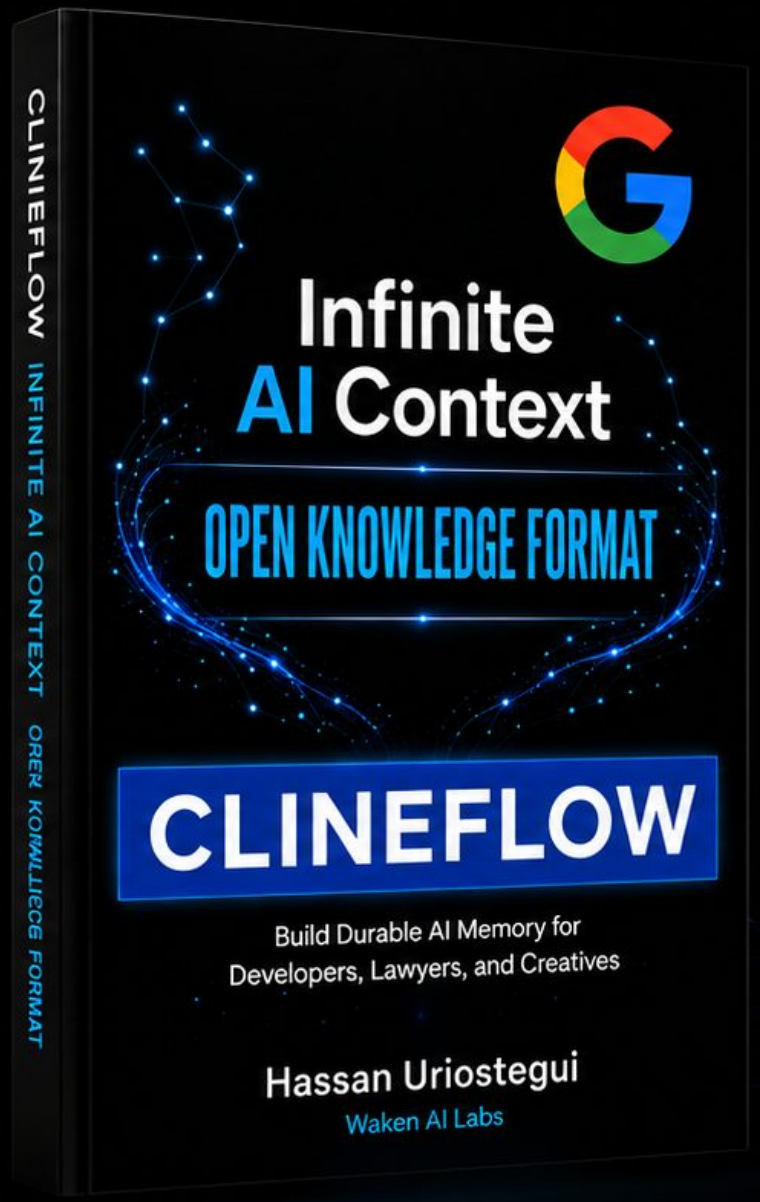
AUTOMATED BEHAVIORAL TESTS

End-to-end behavioral tests verify core functionality, state integrity, and resilience across scenarios.

```
$ bash
node --test
```

TEST	STATUS
create task	PASSED
reject blank input	PASSED
toggle task	PASSED
delete task	PASSED
save/load round trip	PASSED
missing storage recovery	PASSED
corrupted JSON recovery	PASSED

BROWSER ACCEPTANCE LOOP



VII
Goal & Success

www.clineflow.com

‘WORKING APP’ IS NOT A DEFINITION OF SUCCESS

Too vague

- looks good
- it works
- seems fine
- no errors
- works on my machine

Observable success

- add non-empty task
- toggle both ways
- delete independently
- survive reload
- recover from malformed storage

SPEC + SUCCESS = EXECUTION CONTRACT

Once the project knows what to build and how to prove it, the agent can execute.

DEMO

DEMO 3 · NEW CHAT + IMPLEMENT TO THE SUCCESS CONTRACT

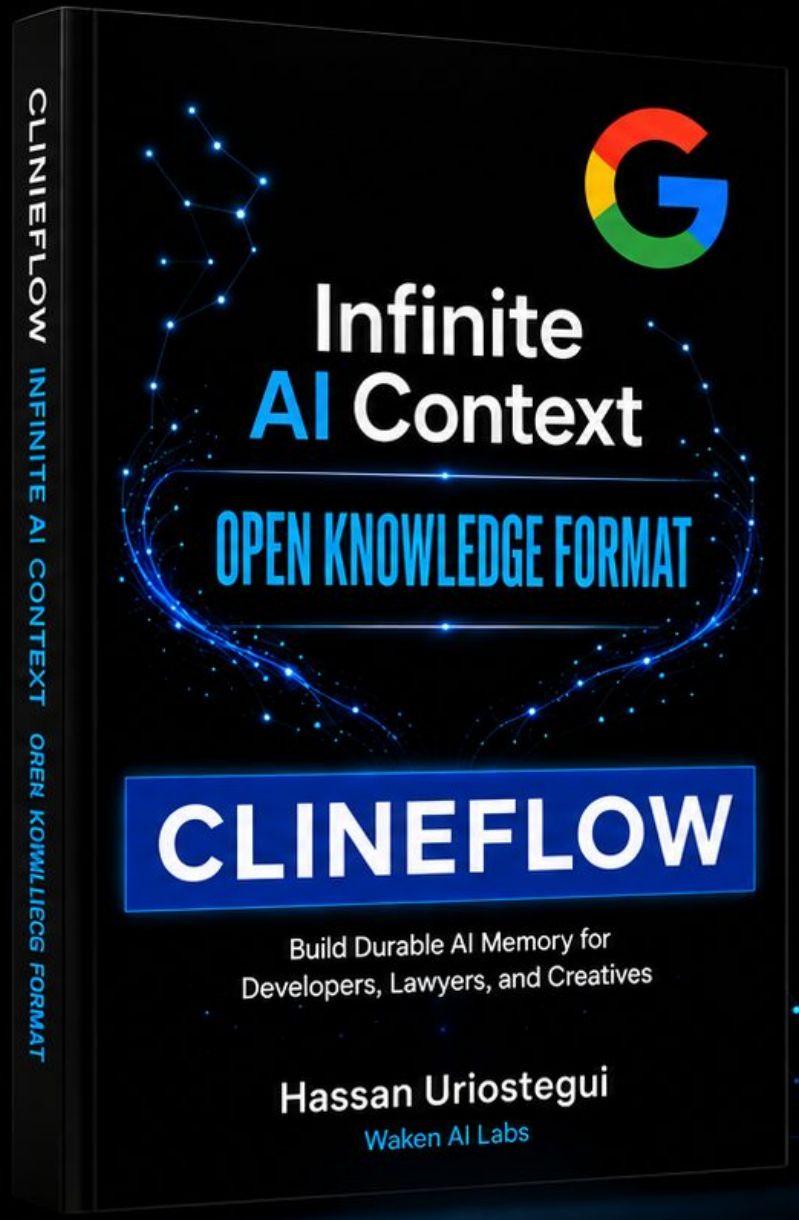
GOAL Prove the conversation is disposable once spec, tests, and success criteria are grounded.

ON-SCREEN ACTIONS

1. Close the original Codex chat and open a completely fresh one.
2. Ask only for the journal/spec summary first.
3. Authorize implementation only after it reconstructs the approved contract.
4. Run Node contract tests after each logical change.
5. Run Playwright browser acceptance in headed mode so the audience can see the app exercising the contract.
6. Allow failures to appear; let the agent repair until the agreed checks pass.
7. Finish with “please commit”.

PROMPT TO TYPE

```
Check the journal and summarize the approved Todo specification, test contract, and
definition of success. Do not change files yet.

After you summarize it correctly, implement the approved specification.

Use the journaled success criteria as the completion target.
Keep project memory current.
Run the Node contract tests and the browser acceptance tests.
Continue until the agreed checks pass or you encounter a product decision that is
not covered by the specification.

When the implementation and verification are complete, please commit.
```

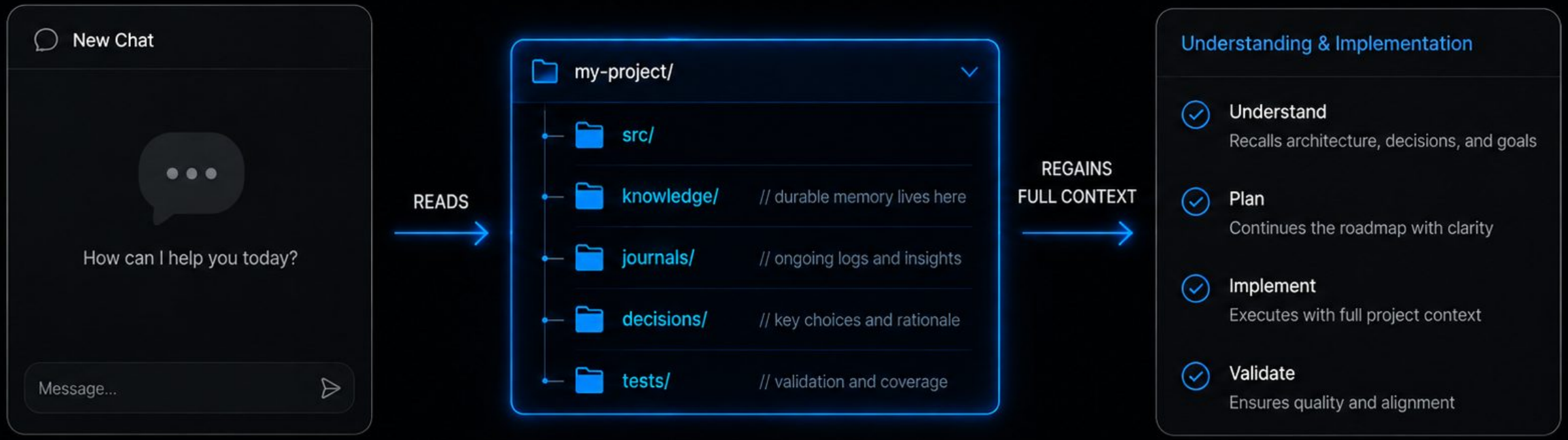


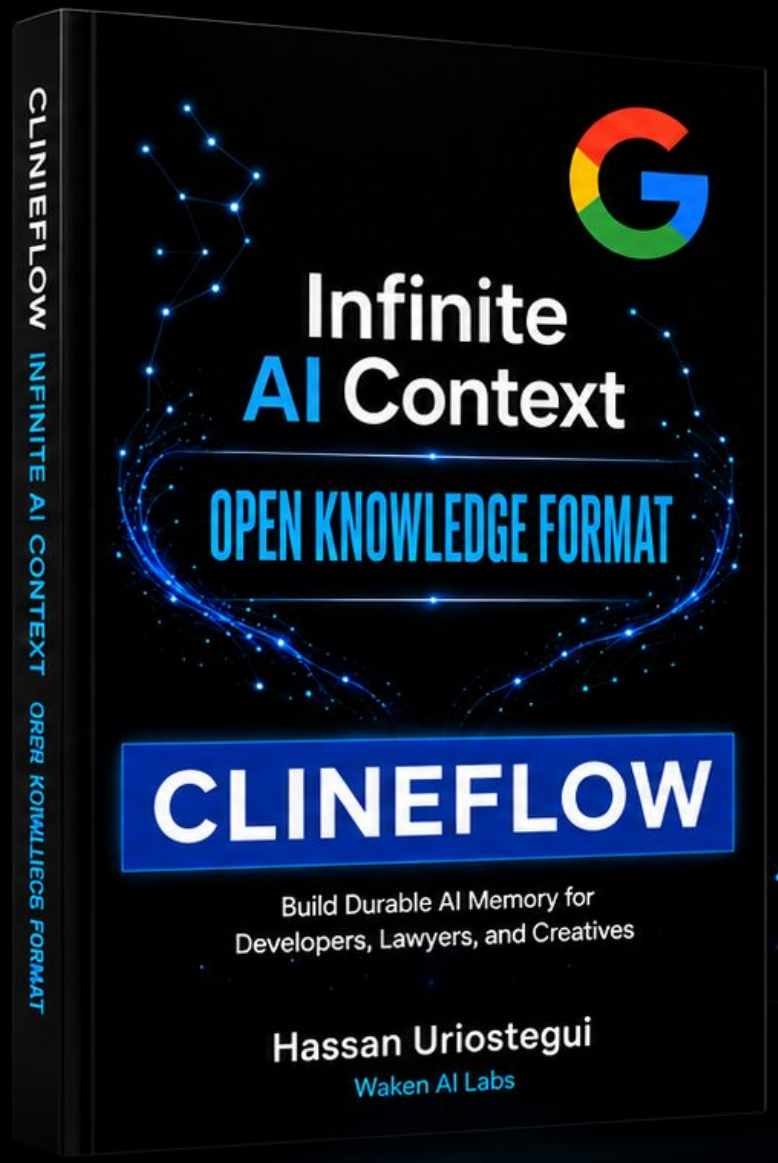
VIII Dev Loop

www.clineflow.com

NEW CHAT. SAME PROJECT.

The conversation is gone. The project still knows what to do.





IX

Ship to GitHub

www.clineflow.com

LOCAL PROJECT → GITHUB → PAGES

Static app. Clean deployment. Deployed proof.

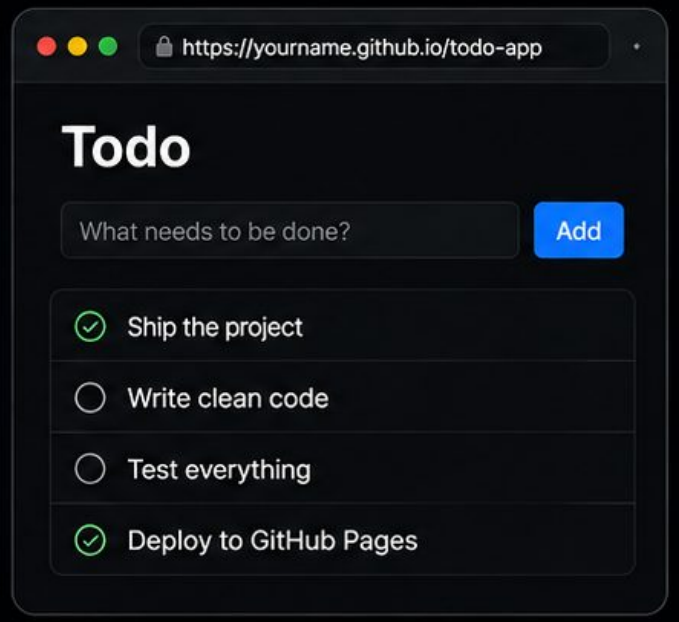
LOCAL PROJECT
Your code. Your machine.

→

GITHUB REPOSITORY
Push. Store. Version.

→

GITHUB PAGES
Build. Deploy. Go live.



DEMO

DEMO 4 · PUSH + GITHUB DEVICE FLOW + PAGES

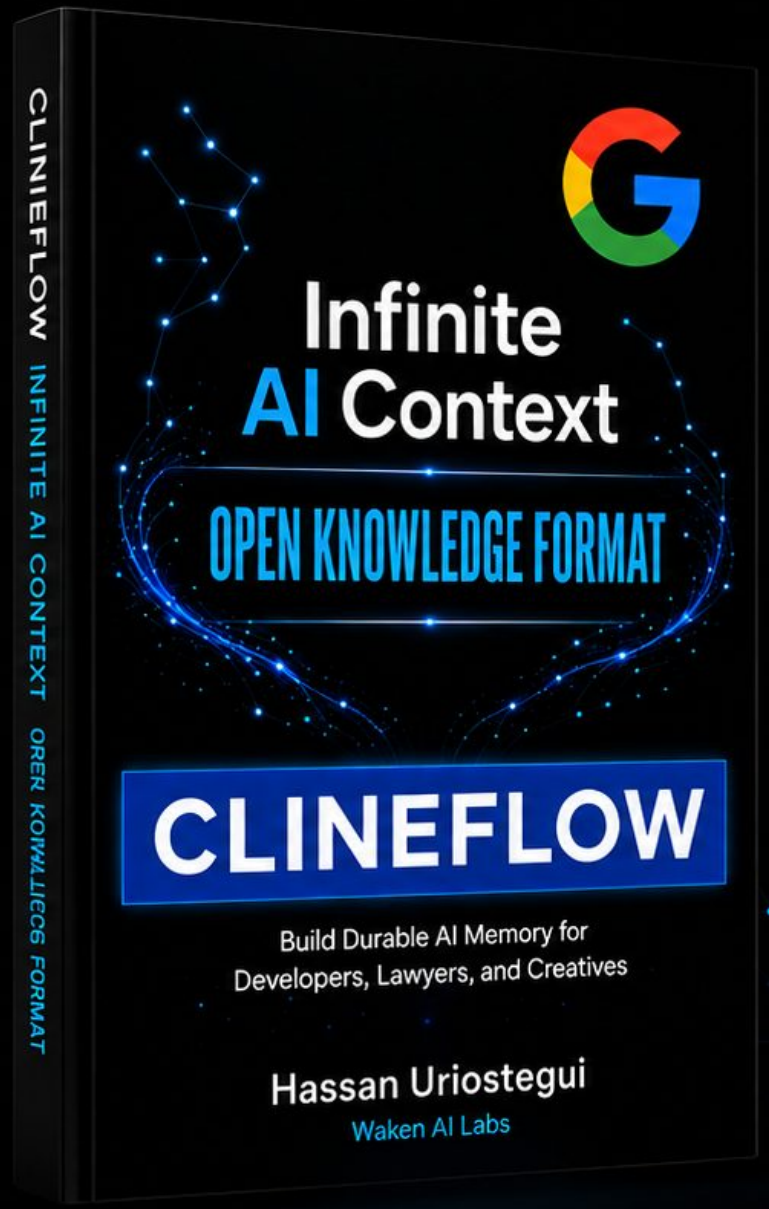
GOAL Turn the local proof into a public static deployment without adding application architecture.

ON-SCREEN ACTIONS

1. Create or connect the GitHub repository.
2. If GitHub CLI requests authentication, show the device-code/browser flow instead of hiding it.
3. Push the committed project.
4. Enable GitHub Pages from the repository branch/root appropriate to the static site.
5. Open the live Pages URL.
6. Add a task, complete it, reload, and show that localStorage persists on the deployed origin.
7. Optionally run the Playwright smoke suite against the Pages URL.

PROMPT TO TYPE

```
Prepare this repository for GitHub and a static GitHub Pages deployment.

Do not add a framework, backend, build service, or task-data network API.
Preserve the current plain HTML/CSS/JavaScript architecture.

Show me each command before any authentication or remote-changing step.
After the repository is pushed, tell me the exact Pages setting or command required
to publish the current static site.
```

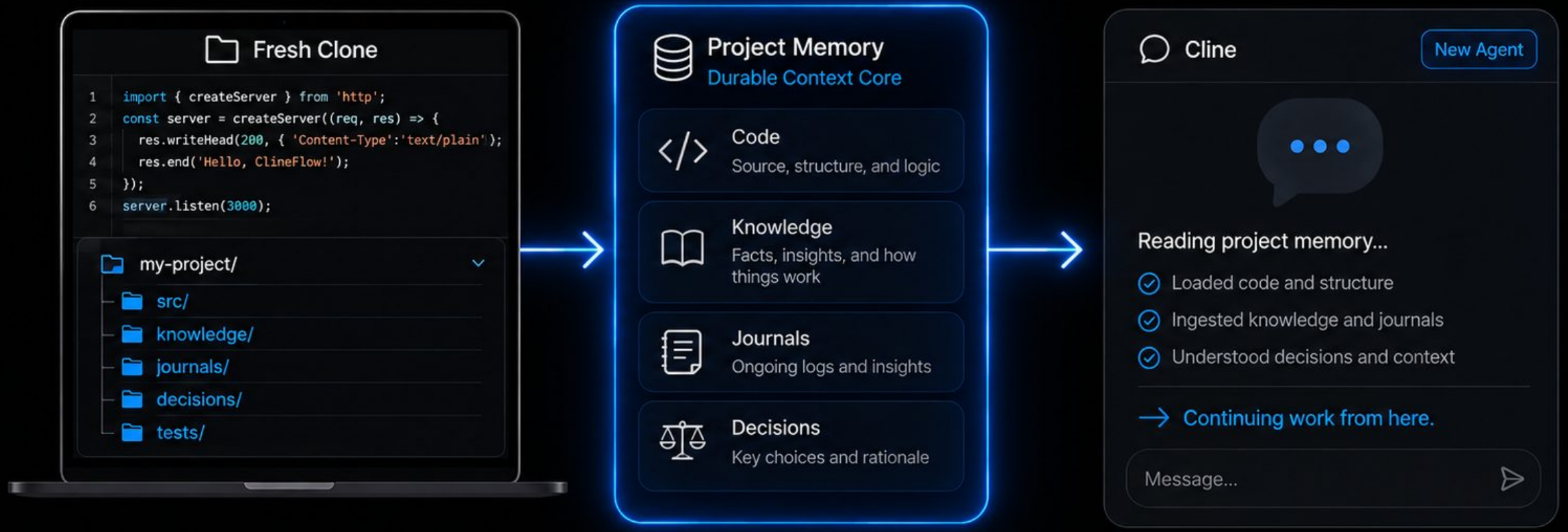


X Collaborate

www.clineflow.com

FRESH CLONE. DIFFERENT AGENT.

Clone the repo, open another copilot, and continue from the project memory.



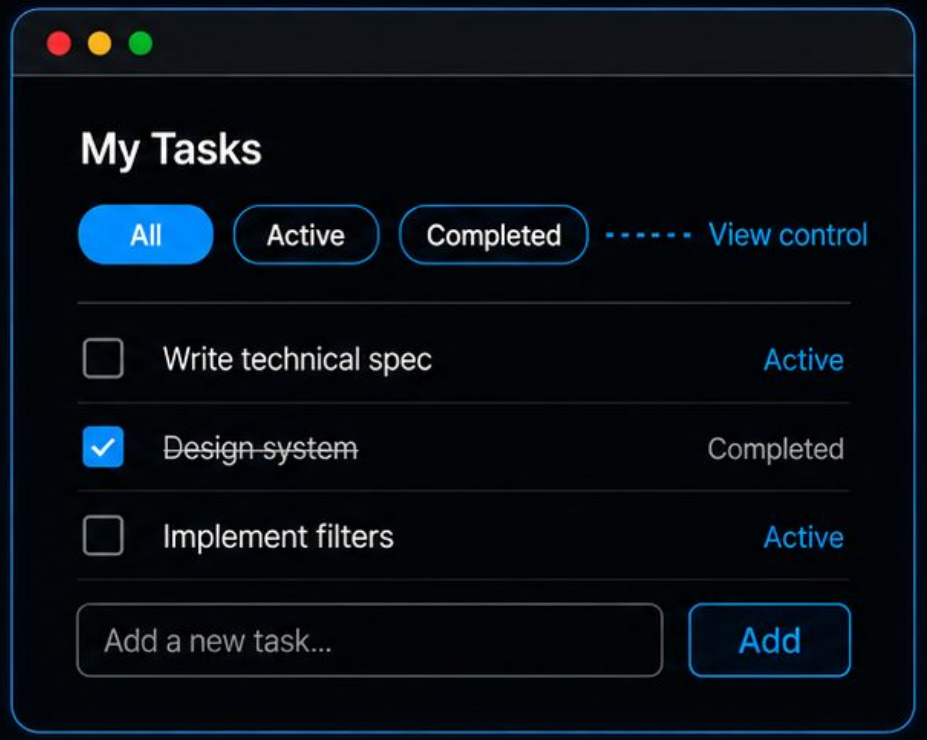
EXTEND WITHOUT BREAKING THE CONTRACT.

A new agent adds filters while preserving the original task model.

Persisted Task Model

```
{
  id: string,
  title: string,
  completed: boolean
}
```

unchanged



View State

current filter:
All / Active / Completed

temporary

Rules

- filters do not alter stored task data
- existing add / toggle / delete behavior still passes
- persistence remains localStorage only

DEMO

DEMO 5 · FRESH CLONE + DIFFERENT AGENT + FILTERS

GOAL Show Cline continuing from the same project memory in a fresh clone without inheriting the Codex chat.

ON-SCREEN ACTIONS

1. Clone the repository into a new directory.
2. Open it in VS Code with Cline.
3. Start a fresh Cline conversation and ask it to read the journal before editing.
4. Introduce All / Active / Completed filters.
5. Ground the extension first: filter selection is temporary view state and must not change the persisted task model.
6. Require the old tests to remain green plus new filter tests.
7. Implement, verify, and “please commit” on a feature branch.

PROMPT TO TYPE

```
Check the journal and explain the current Todo project before making changes.

I want to add All, Active, and Completed filters.

Ground this as an extension of the existing specification first.
Decide whether filter selection belongs in persisted task data.
Preserve the existing task model and persistence boundary unless the current
project context explicitly requires otherwise.

Define the new success criteria and regression checks before implementation.
Then implement the approved slice, run the existing and new tests, update the
journal, and please commit.
```

THE PR CARRIES CODE + CONTEXT.

A pull request should contain implementation, tests, and the reasoning behind the change.

- Code = UI + logic
- Tests = regression coverage
- Decision = filter state is temporary
- Journal = what changed and why

- All / Active / Completed filters
- task model unchanged
- tests passing
- ready to review

DEMO

DEMO 6 · PUSH THE FEATURE BRANCH + OPEN / MERGE THE PR

GOAL Make the collaboration artifact carry implementation, evidence, and reasoning together.

ON-SCREEN ACTIONS

1. Push the Cline feature branch.
2. Open a pull request.
3. Show that the diff includes implementation, tests, and journal/decision updates.
4. Call out the key decision: filter state is temporary; task persistence schema is unchanged.
5. Review the checks.
6. Merge the PR.

PROMPT TO TYPE

```
Push the current feature branch and prepare a pull request.

The PR description should summarize:
- the user-visible filter behavior,
- the decision that filter state is temporary view state,
- confirmation that the persisted task model is unchanged,
- tests added or updated,
- regression checks that passed,
- the journal/context files changed.

Do not merge until I review it.
```

BACK TO CODEX. PULL. NEW CHAT.

Work done by another agent arrives through Git, then a fresh chat reconstructs the project.

1

Merged PR

Another agent merged work into the repo.

2

```
$ git pull

Fetching new changes...
Already up to date.
Your branch is up to date
with 'origin/main'.
```

git pull

New changes arrive from Git into your original checkout.

3

New Codex Chat

Start a fresh chat. Codex reads the project state, not the old context.

4

Recovered Context

The project state restores the context the old chat lost.

The original chat did not remember this. The project did.

DEMO

DEMO 7 · BACK TO CODEX + PULL + FRESH SUMMARY

GOAL Complete the strongest continuity proof: Codex understands work performed by Cline in a chat it never saw.

ON-SCREEN ACTIONS

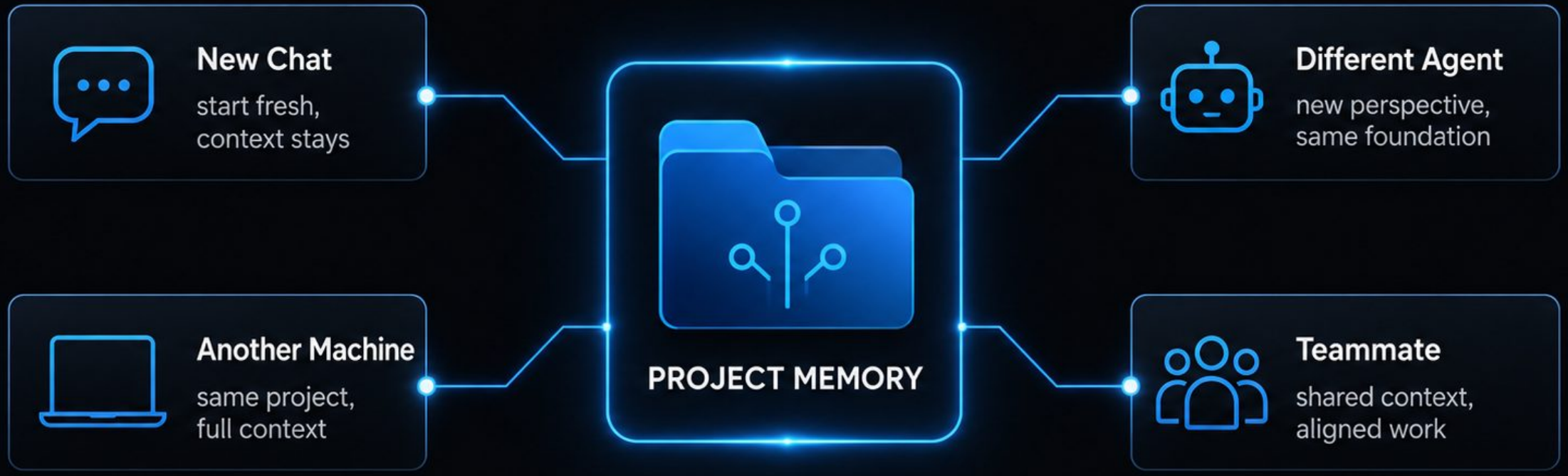
1. Return to the original Codex checkout.
2. Run git pull and receive the merged feature.
3. Open another completely fresh Codex chat.
4. Do not summarize Cline's work yourself.
5. Ask Codex to read the journal and explain the project history, current behavior, verification, and next safe step.
6. End on the fact that the models did not remember — the project did.

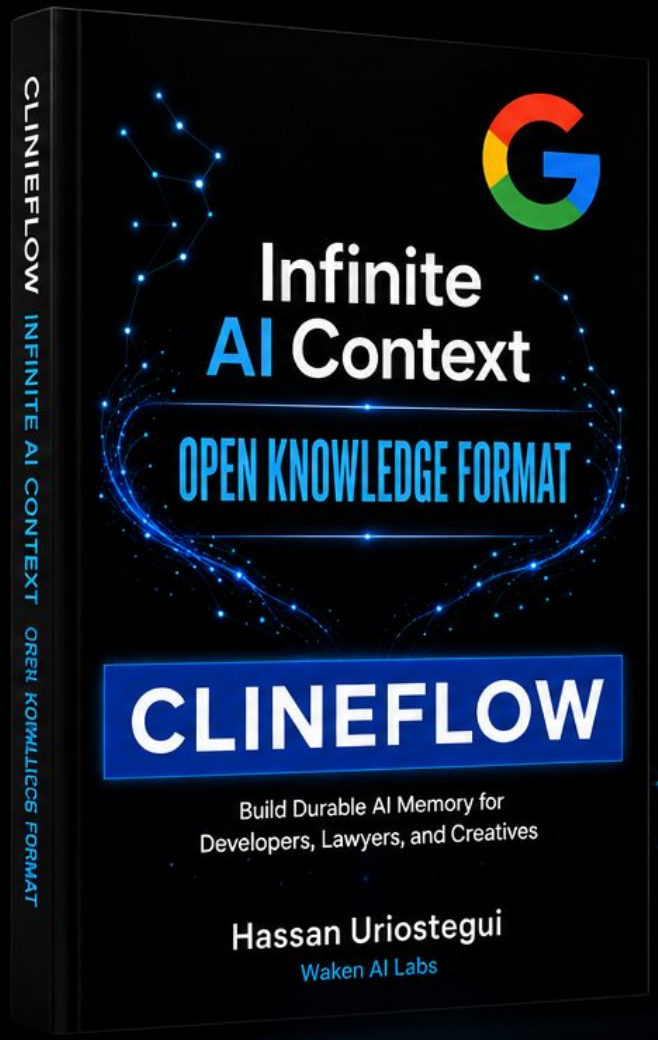
PROMPT TO TYPE

```
Check the journal and summarize the history of this project so far.

Tell me:
- what we originally set out to build,
- the approved Todo v1 contract,
- what has been implemented,
- what the other agent added,
- which decisions constrain the current design,
- what verification has passed,
- and the next reasonable improvement.

Do not make changes.
```

THE PROJECT REMEMBERED.

Chats end. Agents change. The project carries the context forward.





Infinite AI Context

www.clineflow.com

[Book] | [get the free eBook] | [install from Github]