

START [HERE](#)

Understanding AI harnesses.

The tools, context, permissions and controls around an AI model.

~/GUIDES/004

The model thinks about a task. The harness makes a workflow possible.



00 / INTRODUCTION

More than a model.

If you have used a coding agent, you have probably used a harness without thinking much about it.

A language model can propose text or actions. A harness is the software surrounding the model that supplies context, exposes tools, manages a session and coordinates authorised actions. The exact features differ between products. It is useful to know what the model decides and what the surrounding system actually does.

THE BASIC DISTINCTION

Model = generates decisions or responses. Harness = manages the working environment, tool calls and rules around those responses.

IN THIS GUIDE

- 01 Core components**
Model, agent, harness and environment.
- 02 The execution loop**
How a tool request becomes a result.
- 03 Context and memory**
What carries across a long task.
- 04 Permissions**
Where real safeguards belong.
- 05 Reliable workflows**
Tests, checkpoints and recovery.
- 06 Your checklist**
Questions to ask about a harness.

01 / THE COMPONENTS

Four different layers.

These terms are related, but they do not mean the same thing. Keeping them separate makes technical discussions much clearer.

01 / MODEL

The generator

A language model interprets supplied context and produces responses or requests for permitted actions. It does not automatically have access to your computer.

02 / AGENT

The goal-driven system

An agent uses instructions, a model and possibly tools to work towards a goal through multiple steps. Some agents are tightly supervised.

03 / HARNESS

The control layer

The harness prepares messages, manages state and the tool loop, routes actions and applies configured permission or approval rules.

04 / ENVIRONMENT

Where work happens

Files, command-line tools, browsers, sandboxes and connected services are the places in which authorised actions may run.

USEFUL MENTAL MODEL

The same model may behave very differently when paired with different tools, context and execution policies.

02 / WHAT HAPPENS

Follow a single request.

A normal agent session repeats a controlled exchange between the model and its surrounding software.

01

You provide the goal

The harness receives the task, relevant session history, tool descriptions and any system rules.

02

The model chooses a step

It returns an answer or requests a tool operation, such as reading a file or running a test.

03

The harness checks permission

The system decides whether the requested action is available, approved and allowed in the current environment.

04

A tool performs the work

The tool runs in the authorised environment and returns output, success, failure or an error.

05

The loop continues or stops

The result becomes fresh context. The agent proceeds, requests approval, reports a blocker or returns its work.

IMPORTANT

A model requesting an action is not the same as that action succeeding. Ask for evidence of what really ran.

03 / INFRASTRUCTURE

What the harness supplies.

Features vary, but these four responsibilities explain why a harness matters as much as a clever prompt.

CONTEXT

Instructions and material

Combines the current task with approved files, past steps, project guidance and a description of available tools.

TOOLS

Action adapters

Connects commands, file access, browsers, APIs and other operations to controlled functions that return results.

STATE

Session continuity

Tracks conversation history, tool responses and sometimes persistent checkpoints. Memory may be limited or require explicit files.

POLICY

Approvals and limits

Applies permissions, restrictions, execution budgets and rules for when human confirmation is required.

NOT MAGIC

A harness cannot make unavailable tools exist. Credentials, integration setup and secure permissions still matter.

Same model, new outcome.

Model quality matters, but an effective workflow also depends on what happens before and after a model response.

FRAGILE SETUP

“Just keep trying.”

A coding assistant has a broad instruction, no clear file scope, no test command, no stop limit and no recorded checkpoints. It may loop, drift away from requirements or claim that a task is finished.

BETTER HARNESS

Bounded work with evidence

The same task includes a project brief, restricted access, a list of checks, a retry limit and an agreed stopping point. Results are recorded, tests are run and unfinished work is reported.

THE LESSON

Reliability comes from the workflow around the model, not from confident answers or longer prompts alone.

05 / PRACTICAL LOOP

A website repair task.

Imagine you want an agent to investigate a broken contact form in a local development copy of your website.

A good harness helps translate the goal into observable operations while keeping production systems out of reach.

CONTEXT

Define the scope

Give the project folder, relevant form files, expected behaviour and a strict instruction not to touch the live site.

TOOL USE

Inspect and reproduce

Allow the agent to read source code, inspect logs and run local tests. It should report the actual failure before changing code.

VERIFICATION

Fix, test, document

Review the patch, rerun the failing test, check for regressions and record what changed. Manual browser testing may still be needed.

The harness coordinates access and evidence. The developer remains responsible for approving the completed work.

06 / LONG TASKS

Context does not last forever.

Long-running agents can lose track of details as the conversation grows or the working session changes.

START

Keep a project brief

Write down the goal, requirements, decisions, file locations and constraints outside the chat.

DURING

Checkpoint useful progress

After each phase, record completed changes, test outcomes, open problems and the next step.

TRANSFER

Prepare a fresh session

Summarise the current state and tell a new session exactly what to read before acting.

RECOVER

Use version control

Commit tested work in sensible stages so an error or interrupted session does not destroy the last good state.

PRACTICAL TIP

Ask: "Create a self-contained handoff brief with decisions, changes, failed tests and next actions."

07 / PERMISSIONS

Real controls beat promises.

Telling an AI to be careful is not equivalent to restricting its ability to cause harm.

FILES**Separate read from write**

Grant access only to folders that matter. Start read-only when a task is investigative.

ACTIONS**Approve sensitive changes**

Require confirmation before deploying, deleting, publishing, sending messages or spending money.

ISOLATION**Work in safe environments**

Use local clones, staging sites, containers or restricted workspaces when you can.

INPUTS**Treat retrieved text as data**

A webpage or repository file may contain malicious instructions. It should not override the real user request or permissions.

SECURITY PRINCIPLE

Enforcement must happen in the surrounding software and services, not only in the model's wording.

Set up a small workflow.

You do not need to build a custom platform to apply harness thinking to your existing coding tools.

01

Choose a narrow job

For example, inspect a staging website for broken internal links.

02

Give clear context

Supply the project folder, known commands and the exact definition of success.

03

Limit the tools

Allow reading and testing; do not give production deployment access.

04

Add checkpoints

Require the agent to record what it inspected and what failed after each phase.

05

Check, then expand

Review evidence yourself before allowing changes or giving the system a larger task.

RULE OF THUMB

Start with a reliable ten-minute workflow before trying a project that takes hours.

Audit the setup.

These questions help you understand what your chosen AI tool can actually do, and under which conditions.

01

Model and task

What is the model asked to do? What counts as success?

02

Context

Which files, history and instructions does it receive?

03

Tool access

Which tools are actually connected and permitted?

04

Execution environment

Is it working locally, in a sandbox, or on a live service?

05

Approval gates

Who authorises publishing, deletion or other high-impact actions?

06

Evidence

How can you see actions, errors, changed files and passed tests?

07

Recovery

Where are checkpoints and how do you stop or roll back?

KEEP THIS PAGE

If you cannot answer these questions, limit the task until the setup is clearer.

10 / KEEP LEARNING

Design the workflow.

A good harness makes an AI system useful, observable and easier to supervise.

You can use the same model for chat, research or coding and get very different results depending on its tools, context and controls. Learn the boundaries of your software, start small and build workflows that leave evidence behind.

REMEMBER THESE THREE THINGS

01 The harness is the surrounding system

It coordinates context, tool use, state and permitted actions.

02 Context needs maintenance

Checkpoints and handoffs are essential for longer work.

03 Controls need enforcement

Review evidence and keep permissions narrow, especially near live systems.

FURTHER READING / REFERENCE

Microsoft / Understanding agent harnesses



Anthropic / Effective long-running harnesses



Australian government / Agentic AI harnesses



Explore more at nathm.net