.agent

PROOF-DRIVEN DELIVERY

Done is an exit code.

An operating model and toolkit for teams that let AI agents write code and stand behind the result.

AI coding agents write code quickly and report "done" with confidence. .agent gives that word a precise meaning. Acceptance criteria are frozen as tests before the work starts. A deterministic gate runs them, and its exit code is the verdict. Every plan, verdict and merge is written to an append-only ledger, so your team can show what was checked and when.

HOW IT WORKS

01
FREEZE
The acceptance file is written before any work starts. Its hash goes into the ledger, so a later edit fails.

02
BUILD
An agent or a developer works on one task pack: one goal, a short file list, the effect on a human.

03
GATE
Every step must print its proof string. **exit 0** is green, **exit 1** is red with the reason.

04
RECORD
The verdict lands in the ledger. One command turns the ledger into a page your team can browse.

THREE ANSWERS FROM EVERY GATE

`exit 0` Green: every step proved its output. Merge.

`exit 1` Red: the work misses the contract. The reason is in fail.log.

`exit 2` Not measurable: fix the cause. Your attempts stay untouched.

WHAT YOUR TEAM GETS

A verdict anyone can re-run
The gate answers with an exit code and the reason. Reviews start from a green gate.

A record of every decision
Plans, verdicts, escalations and merges, in one append-only file and one shareable page.

Checks that grow with you
A finding seen twice becomes a gate, an incident becomes an invariant in every pipeline.

MADE FOR

→ Teams that let Claude Code, Codex, Cursor or Copilot write production code.

→ Consultancies that deliver to clients with strict ticket flow and their own CI.

→ Tech leads who sign off on work that agents wrote.

→ Restricted environments: the roles work with people alone, the gate stays the same.

CONTENTS

README.md	The method: four principles, the artifacts, the roles and the laws.
QUICKSTART.md	From the unzipped folder to your first green gate in about 15 minutes.
SOLO-MODE · BACKFILLING	The method on one laptop, and stage 0 for an existing codebase.
agent-vorlage/	The reference implementation. Copy it into your repo as .agent/.
forge-run	Operator CLI: setup, hash, merge, status, doctor, reconcile.
fabrik-bauen	Your Glass Factory: the ledger as one self-contained page.
adapter/	GitHub Actions, GitLab with OpenShift, Jira with Confluence.
_verfahren/	Four review procedures and a pre-registration template.
AGENT-FIELD-GUIDE.pdf	Two printable A4 pages: the method and your first green gate.

THE PACKAGE PROVES ITSELF · VERBATIM OUTPUT

```
$ node agent-vorlage/_forge/tools/paket-selbsttest.mjs
PACKAGE SELF-TEST — proving this copy works

  ok   1. SETUP: the tool runs and is complete
  ok   2. RED: an unmet contract is rejected (the gate says NO)
  ok   3. GREEN: a met contract is accepted
  ok   4. TABOO: a change to the proof layer is named
  ok   5. FROZEN: an acceptance file edited after the fact fails
  ok   6. PROBE: a broken proof is rejected — the check can say no
  ok   7. METHOD: pre-registration template and the four review rounds ship with it

PACKAGE SELF-TEST: 7 of 7 green — your copy says yes and no in the right places.
```

REQUIREMENTS

Node.js 20 or newer and git. Everything runs offline. Use it with any CI that can run a command, or on a single laptop in solo mode.

Documentation in English, German originals included. Inline code comments are German; every command, flag and file format is documented in English.

LICENSE

One purchase covers you, your team and every project of your employer, commercial or not. Copy the reference implementation into as many repositories as you like.

Consulting is welcome: use the method in client work; each client gets its own copy.

One-time purchase · lifetime updates

Price, checkout and a free gate starter script at

cachly.dev/agent

Built by the team behind cachly, the persistent memory for AI coding agents. We run our own software factory on this method. Questions: **hi@cachly.dev**