INTERNAL EXPLAINER · FOR FEEDBACK

The Jarvis Loop, explained

How one of Anthony's AI tools quietly does a piece of work, grades its own work with a strict second opinion, and keeps fixing it until it is genuinely good, with no babysitting.

Plain-English version Read time: ~6 min For: Anthony + team review

TL;DR

- **It is a way to get work done well without anyone watching over it.** One AI does the work, a second AI checks it like a strict editor, and they pass it back and forth until it is good.
- **You only touch it twice.** Once at the start, to approve the checklist of what "good" means. Once at the end, to sign off.
- **The checker is always a fresh, separate AI** that never saw how the first one was thinking, so it catches real mistakes instead of nodding along.
- **It now scores quality 1 to 10** and refuses to call anything "done" until it clears a bar (8 normally, 9 for serious stuff), not just "technically correct."
- **It never fakes success.** If it runs out of tries, it tells you plainly what is still broken and hands it back to you.

The big idea

Imagine you ask a smart assistant to write something or build something for you. Normally you get one draft back, and you have no idea if it is actually right. You have to read it

carefully, find the holes, send it back, wait again. You are the quality control, every time.

The Jarvis Loop removes you from that grind. It is an assembly line with a built-in inspector. Work gets made, the inspector grades it hard, anything wrong goes back to be remade, and only finished, quality-checked work reaches your desk. You set the standard at the start and you approve the result at the end. Everything in the middle runs on its own.

The name comes from "Jarvis," Anthony's nickname for his AI assistant, and "loop," because the work goes around and around until it passes.

The two roles: Maker and Checker

The whole thing runs on two characters. Think of a writer and an editor.

THE MAKER

Does the actual work

Writes the plan, builds the page, drafts the email, fixes the code. It is the worker. It never grades its own work, the same way a writer should not be the one who decides their draft is perfect.

THE CHECKER

Grades it, hard

A separate AI that reads only the finished work and the agreed checklist, then says pass or fail and points to exactly what is wrong. It is the strict editor who has never met the writer.

They do not collaborate. The Maker hands over finished work, the Checker judges it cold. If it fails, the notes go to a brand new Maker (not the original one), and the cycle repeats. That separation is the whole trick, and the next section explains why it matters so much.

Why the checker has to be a stranger

If you ask the same AI "did you do a good job?", it will almost always say yes. It is attached to its own work, just like a person re-reading their own essay misses their own typos. A checker that shares the writer's memory is not really checking, it is agreeing with itself.

So the Jarvis Loop spins up a **completely fresh checker every single round**. This new checker gets only four things:

- The goal (what we are trying to achieve)
- The checklist (what "good" looks like, agreed up front)
- The finished work it is grading
- The real source facts, so it can catch made-up numbers, not just sloppy writing

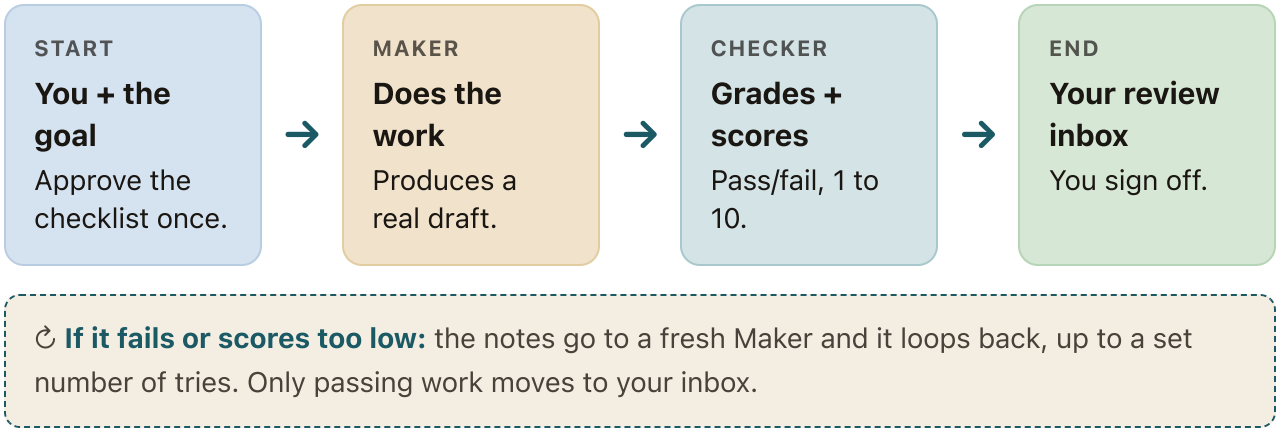
It never sees how the Maker was thinking. By default you get **two** checkers, not one, each hunting a different kind of mistake: one checks that it is correct and the facts are real, the other checks that it actually hits the goal and reads well. For important jobs it scales up to **three**. Two angles catch what a single reviewer would miss, which is the whole reason a lone checker is not enough.

The honesty rule

For every box it ticks as "passed," the checker has to quote the exact line that proves it. No quote means automatic fail. It is told to assume the work is wrong until the work proves it is right. This is what stops a lazy "looks good to me."

How a run actually goes, step by step

Here is the full lap, start to finish. Steps are numbered the way the tool numbers them (it starts at zero, because step zero is the setup before any work happens).



1

STEP 0

Agree on what "good" means, and gather the facts

Before any work starts, the tool writes a short checklist: the one-line goal, 3 to 8 specific things that must be true for it to count as done, and the hard rules (Anthony's preferences, brand, budget, no em-dashes). It also pulls in the real source material (the project card, the documents, notes from Anthony's "second brain") so both the Maker and the Checker are working from facts, not guesses.

This is the one moment you weigh in up front. You glance at the checklist and either say "yes" or tweak it. After that, it runs on its own.

2

STEP 1

The Maker does the work

A fresh Maker gets the goal, the checklist, and the facts, then produces a real draft or builds the real thing. If the job is code, it works in a safe isolated copy so nothing breaks, and it has to leave it in a working, runnable state.

3

STEP 2

The Checker grades it, hard

A fresh, separate Checker (or a panel for big jobs) reads the work and grades it against the checklist, quoting proof for every pass. For code, the real test runs first: if it does not build or the tests fail, that is an automatic fail before anyone even reads it. The Checker also gives a 1 to 10 quality score and lists exactly what would make it a 10.

4

STEP 3

Pass, or loop back

If it passes everything *and* clears the quality bar, it moves on. If not, the specific gaps get handed to a brand new Maker, and it goes back to Step 1 to fix them. This repeats until it passes or hits the cap (see "Where it stops").

5

STEP 4

It lands in your review inbox

The finished work goes to the "Ready to Review" column, never straight to "Done." You get a tight summary: the goal, the verdict, the quality score, the link, and the one thing (if any) that needs your call. You are the final human sign-off.

The quality bar: the part that was just added

Here is the upgrade that just went in. Before, the loop only asked "is this *correct*?" A boring-but-accurate answer could pass. That is a low bar.

Now the Checker also gives a **1 to 10 quality score**, and the work has to clear a minimum bar to count as done: **8 out of 10 for normal work, 9 for high-stakes work** (anything going to a customer, anything about money or policy, or anything Anthony flags). The Checker also writes down exactly what is missing to reach a 10, and those notes get fed back to the next Maker.

> **In one line**
>
> The loop used to stop at "correct." Now it keeps pushing until the work is "actually good." Correct is the floor; the quality bar is the new ceiling it climbs toward.

The tool also keeps a running log of the score across rounds, like "round 1 was a 6, round 2 was an 8." Over time that shows whether the loop is genuinely improving the work or just spinning.

Where it stops, and why it will never fake "done"

An endless loop is dangerous and expensive, so there are hard limits baked in from the start:

- **A tries cap.** It only gets so many rounds to fix things (usually 2). It cannot loop forever.
- **A spending cap.** There is a hard ceiling on how much AI usage one run can burn. Hit it, and the run stops.
- **The "streak" rule.** When the result depends on live data that can shift between checks, one passing grade is not enough. It has to pass cleanly two or three times in a row. One lucky pass does not count.

> **If it cannot get there**
>
> When the loop hits a cap and still has not passed, it does **not** pretend it succeeded. It writes "DID NOT PASS, here is exactly what is still open," attaches the best version it has, and sends

it to you anyway, because that needs a human decision, not a fake thumbs-up. Honesty over a clean-looking lie is the rule.

When to use it, and when not to

Good fit

- The task has a clear "done" you can write down as a checklist
- A plan, a report, a hiring doc, an email, a piece of code
- Work you would normally have to review yourself, line by line
- One-off projects (often a single project card) you want driven to a finished state

Not a fit

- Pure taste with no checkable standard ("make it feel cooler")
- A judgment call only a human should own (a hire, a big spend)
- Anything where you cannot describe what "good" looks like

The simple test: if you can list what a great result must include, it can be looped. If "good" is only a gut feeling, turn that feeling into a checklist first, then loop it. A loop with nothing to check against is just an AI agreeing with itself.

The dials you can turn

A few settings control how strict and how thorough a run is. You rarely touch them, but here is what they mean in plain terms.

DIAL	DEFAULT	WHAT IT DOES
Tries cap	2	How many times the Maker gets to fix things before it stops.
Quality bar	8 (9 for serious)	The 1 to 10 score the work must hit to count as done.
Panel size	2, or 3 for serious	How many separate checkers grade it, each from a different angle.

DIAL	DEFAULT	WHAT IT DOES
Streak	2 (3 for serious)	How many clean passes in a row it needs when the data can shift.
Spend ceiling	capped	The hard limit on AI usage for one run, so it can never run away.

"Serious" (the tool calls it high-stakes) automatically kicks in for anything customer-facing, anything involving money or policy, or anything Anthony flags. It raises the bar and adds more checkers.

How it fits with Anthony's other tools

These are its **neighbors, not its ingredients**. One lap of the loop is only ever two things: a Maker and a Checker. The Jarvis Loop does **not** run all four of these. The others sit before it, after it, or are baked into the checker. Here is exactly where each one lives:

`/deep` BEFORE A separate thinking tool. You use it up front, when the right approach is not obvious, to figure out the plan and the checklist. The loop borrows that careful-thinking style at setup, but it does not call /deep every lap.

`jarvis-loop` THE ENGINE The loop itself. Takes the goal and the checklist and drives them to a finished, quality-checked result. This is the page you are reading.

`rateandimprove` INSIDE The only one that actually runs inside the loop. Its "score it 1 to 10 and push toward a 10" grading lens was just built into the Checker. That is the upgrade described above.

`/loop` AFTER An optional wrapper that runs once the work is finished. It re-checks the result on a schedule to catch anything that drifts as the inputs change later. Separate from a single run.

Big picture: this is Anthony's testbed for "self-driving" AI work. Once it has proven itself on a handful of real projects, the plan is to turn it into a simpler team version so anyone can point an AI at a task and trust the result.

Feedback welcome

This is a working tool, not a finished one. If a step is confusing, if the quality bar feels too low or too high, or if you can think of a job it should or should not be trusted with, tell Anthony. That feedback is the point of this page.
