Human out of the loop

Human out of the loop

25 ways to let your AI keep working on its own, safely and robustly

Floris Knol

Floris Knol

ISBN 978-94-0395-095-2

First edition, 2026

Cover design: Floris Knol

© 2026 Floris Knol

Published via Bookmundo

All rights reserved. No part of this publication may be reproduced or made public without the prior written permission of the author.

AI tools were used in making this book.

Contents

Foreword 7

1. Why the human is often the brake 9

2. An email address for your agent 17

3. Logging in, logging out and keeping sessions 23

4. One-time codes 29

5. Secrets outside the chat 35

6. Connecting without a plugin: the API itself 41

7. Where there is no API: the browser as the connection 47

PART II. LIMITS

8. Your safety rules as hard limits 53

9. Measuring whether your agent can escape 59

10. A gate for what the agent sends 65

11. A log that records everything 71

12. Human check as a setting, not a habit 77

13. Preparing work so the agent does not disturb you 83

14. All the questions at once, then carry on 89

15. Keeping on running when your computer is off 95

16. Restarting, a nudge, resuming 101

17. Recovering after an outage at the provider 107

18. A queue and a board 113

PART IV. COST

19. Cheaper when it keeps grinding on 119

20. Caps per day and per agent 125

21. Fewer tokens 131

PART V. REMOTE CONTROL AND QUALITY

22. Steering from your phone 137

23. An overview of what is running 143

24. Notifications that matter 149

25. Tests your agent runs itself 155

26. From one-off job to fixed role 161

TO FINISH

27. An afternoon to get started 167

Wantzel Studio and more 175

Foreword

I let AI agents do real work every day. Anyone who does that keeps running into the same things: an account that wants a code by text message, a website with no connection to plug into, an agent that stands still because it is waiting for an answer from you. Each of these problems has an approach you can set up in an afternoon. This book puts them side by side, 25 in total.

Who is this book for?

For anyone who already works with an AI assistant such as ChatGPT and wants the work to carry on while you are not sitting next to it. You do not need to program, and there is no code in this book. Every chapter explains the idea and gives you a prompt that you hand to your AI agent, such as Claude Code. The agent builds it; you check that it works. You do need to dare to give an assignment to an AI agent and to check what comes back; where something asks more of you, the chapter says so. It is a book for tinkerers: for people who want to try it out, break it and get it whole again.

The company in this book

All the examples take place at Lindeboom Fietsen, an invented bicycle business that restores second-hand bikes and sells them online. The people, the agents and the events are made up. The problems are real: they come from my own work and from what others who work with AI run into.

Tools

I name concrete tools, such as Playwright, a coding assistant in the terminal, an authenticator app and a server. Versions and menus change. The idea behind them stays; look up the current instructions if a button has a different name.

How the book is built

The book has five parts: access, limits, keeping on running, cost, and remote control and quality. You can start with any part, but the introduction explains how each chapter works and what safe and robust means. Read that first.

Floris Knol

1. Why the human is often the brake

Marijke Lindeboom runs Lindeboom Fietsen, a bicycle business with a workshop where second-hand cargo bikes are restored and a web shop where they are sold. For a few months now she has worked with four AI agents: Wiel looks into the suppliers' portals, Spaak reads the customer mail, Ketting does the administration and Bel updates the shop at night. On the first day it went well. On the second day it started to chafe.

Wiel stood still because he was no longer logged in at a supplier. Spaak had a reply ready and was waiting for permission. Ketting asked for a code that had been sent to Marijke's phone. Bel had a question about a photo in the night and waited until she woke up. Marijke sat next to her agents all day, and that was exactly what she did not want.

This book is about that day, and about everything she did about it.

The human in the loop

You often hear that there has to be a human in the loop. To a large extent that is true. An agent that makes payments, sends mails

and deletes files without supervision will sooner or later do something you did not want.

But look at what the human in the loop does in practice. The human is the one who clicks "allow". They type in the code. They log in again. They answer the question the agent asked at three in the night, the next morning at eight. They are the link in the chain that is slowest, that sleeps, that is travelling, that is in a meeting. Every time the agent waits for him, the work stands still.

You see the result with almost everyone who tinkers with AI. You start enthusiastically, your agent does in an hour what took you an afternoon, and after a week you are sitting there all day, ready to help it along. You have an assistant you cannot let go of. The work goes faster than before, and it keeps you just as busy.

This book is about the step after that: taking the human out of the loop. You stay the boss, you keep setting the limits and you look afterwards at what happened. But you no longer sit in between. The agent keeps running, even when you sleep, sit on the train or talk to a customer.

What safe and robust means

The title has two words that matter. Each is worked out in its own way in this book.

Safe means the agent does only what you have allowed, even when nobody is watching. You get there by putting the limits outside the agent: in permissions, in folders, in a gate for everything that

goes outside. A rule in a prompt is a request. A rule in the environment is a limit. Safe also means you can check what happened: a log the agent cannot change, and a self-test that shows the limits hold.

Robust means it keeps working when something goes against you. A session that expires, a code that is asked for, a provider with an outage, a limit that is full, an agent that gets stuck without reporting it. A robust setup notices this itself, tries a next route and lets you know only when it matters.

The two belong together. An agent that runs on without you and has no limits is a risk. An agent with limits that falls silent at the first outage is an agent that asks for your attention again after all. The book covers both.

The company you read along with

The examples take place at Lindeboom Fietsen, Marijke's business. It is invented, and it stays that way throughout the book: the people and the events do not exist. The problems they run into do. A supplier with a login portal without a connection, a bookkeeping system that asks for a code by text message, customer mail that arrives while everyone is working: you recognise that from your own business, even if your business does something entirely different.

Besides Marijke there are Bram in the workshop, Yara in the office and Teun the bookkeeper, who comes by twice a month. Of the

agents you meet Wiel, Spaak, Ketting and Bel. Each has its own task; you only ever see one or two at a time.

How each chapter works

The book has 25 ways to take the human out of the loop, divided over five parts. Each way is a chapter with the same six sections, so you know where you are:

1. **The problem.** What you run into as a user, in a situation you recognise.
2. **Why it holds you back.** What lies under the problem, so that you also recognise it in another form.
3. **The approach.** How it works, in plain words, and the idea behind it.
4. **The hands-on block.** A prompt that you give to your AI agent, so that it builds the thing for you, with the limits it must keep to and how you check the result. Every prompt was given to an AI agent in a disposable folder before it went into the book.
5. **The pitfalls.** Where it goes wrong and how you notice.
6. **The checklist.** A short list to go through to see whether it is in place.

Every chapter ends with something you do the same day. You can read the chapters in order, and the order is intended: what comes first makes what follows easier. You can also start with the part

where you run into the most trouble. Where a chapter uses something from an earlier chapter, it says so.

One more agreement. Versions and menus change, so look up the current instructions if a tool has a different name. Prices from providers are not in here, because they change faster than a book. And where something works differently at the provider of your model, you look in its documentation; the idea behind it stays.

How you do it safely

An agent that works independently also makes mistakes independently. That is why the book holds on to a few rules that come back in every chapter:

- Start with read-only. Give an agent write permission only once you have seen it read.
- Give each agent the minimum permissions its task needs.
- Keep secrets outside the chat and outside the logs.
- Let everything that cannot be undone pass through a gate.
- Play back every setup first on a throwaway environment, and only then look at your real work.

That last one applies to the hands-on blocks in this book too. Where a block touches your own accounts, it says so at the block, with the trial version you can do safely.

Do it today: the waiting list

For the first chapter you need no tools, only a sheet of paper or an empty file and a day.

Put two columns on it. In the left one you write every time your agent waits for you: a permission, a code, a login screen, a question, an agent that stands still. In the right one you write what it needed to carry on by itself. Do that for one working day, for every agent you use.

At the end of the day you see two things. You see how often you are the brake, and that number is yours: no book can count it for you. And you see which waiting moments keep coming back. Those are your first candidates. If a waiting moment belongs to access, go to the first part. If you mostly wait for permission, start with the part about limits. If your agent stands still as soon as your laptop closes, look at keeping on running.

Keep the list. You use it at the end of each part to see what has disappeared.

To remember

- The human in the loop is often the slowest link: he sleeps, travels and has meetings.
- You stay the boss and you no longer sit in between: you set the limits and look afterwards.

- Safe is that the agent stays within your limits, even when nobody is watching.
- Robust is that it carries on when something goes against you, and warns you only when it matters.
- Every chapter has the same six sections and ends with something you do the same day.
- Start with the waiting list: write down for one day what your agent waits on you for.

2. An email address for your agent

The problem

Marijke lets Spaak read the customer mail for the webshop. To do that properly, Spaak needs an account with the shop's ticket system, an account with the carrier, and an account with the platform where the bikes are advertised. Marijke created all three with her own email address, because that was the one at hand.

A week later, what you would expect happens. The carrier sends a confirmation email, the platform asks for a new password, and the ticket system reports a suspicious sign-in. It all lands in Marijke's inbox, between the quotes and the messages from Bram. Spaak can't reach any of it, so Spaak stands still. Marijke finds the email, clicks the link and pastes it into the conversation. Then Spaak runs again, until the next email.

Why it holds you back

An account belongs to whoever's email address is attached to it. Every confirmation, password reset, warning and one-time code goes to that address. If you put your agent's account on your ad-

dress, you become your agent's postman. By definition, you sit in between.

There is something else. Your own inbox contains everything: your bank, your private messages, the mail from your accountant. If you let the agent read your inbox so it can handle confirmations itself, you give it access to all of that. An address of its own keeps the access small: the agent sees its own mail and nothing else.

The approach

You give every agent its own email address and put every new account on it. You do that in four steps.

Choose where the address lives. If you have your own domain, create an extra mailbox with your email provider, for example `spaak@` followed by the name of the company. If you don't have a domain, a free mailbox with a well-known provider works too, as long as you make a separate address for each agent. One mailbox per agent keeps Wiel's mail apart from Spaak's, and you see at once which agent has which accounts.

Create access that only reads and fetches. Most providers let you create an app password, a password of its own for programs. You use that for the agent, never your main password. If you ever need to withdraw the access, you revoke that one app password and the rest of the mailbox keeps working. Look for "app password" at your provider and read the current instructions, because menus