BUILD & AUTOMATE WITH AI

JOHANNESBURG · 15 OCT 2026 · 19:00

Parallel AI Coding Agents

Build Fast, Keep Control

Chaïm Zonnenberg

SOFTWARE ASSIST

About me

- Senior software architect
- 20+ years of .NET
- Enterprise background

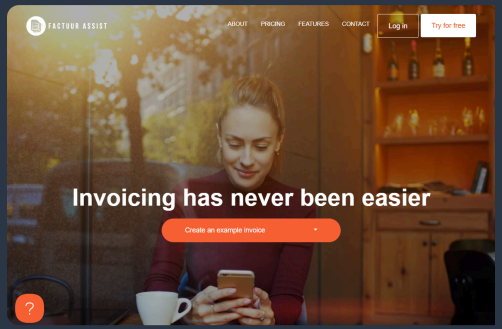
Financial services

Cloud applications

Identity & access management

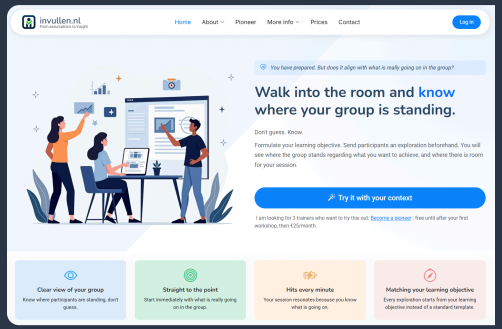


Software Assist and its products



Factuur-Assist

Online invoicing



Invullen.nl

Online questionnaires

Both built and maintained with AI agents, every day.

ALSO: ADVISING TEAMS ON AGENTIC ENGINEERING

SOFTWARE ASSIST

My stack

BACKEND .NET 10	FRONTEND Vue 3
CLOUD Azure · Kubernetes	PIPELINES Azure DevOps
SOURCE & ISSUES GitHub	AGENTS Claude Code · Codex



TYPE A NUMBER IN THE CHAT

How much experience do you have with AI coding agents?

0
Never tried an AI coding agent

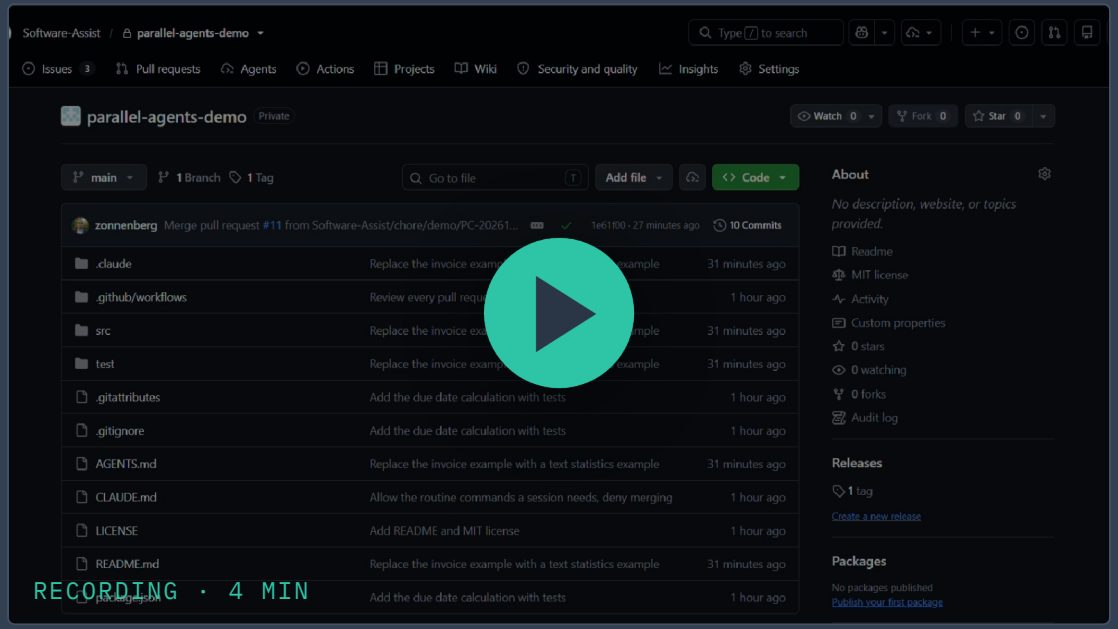
1
Tried one, now and then

2
Use one regularly, one session at a time

3
Run several sessions in parallel

Three sessions, side by side

Recorded, 4 minutes. Three worktrees, three agents, three pull requests.



RECORDING · 4 MIN

1 · worktree per issue

2 · agent per folder

3 · each session ends in its own PR

10–15 sessions in parallel, and
still secure and maintainable?

Why so many sessions?

- 10–15 sessions is not convenient, and not the goal
- Ideally: one session that answers right away
- It is not about speed but throughput: while I give one session my attention, as many others as possible keep working

TIP FOR ANY STACK Start with one session. Get coffee. Add a second when you notice you are waiting.



A SIMULTANEOUS CHESS EXHIBITION

Worktrees: a branch + a folder

~/source/factuurassist-all/	main checkout
├─ .claude/worktrees/btw-1345/	session 1 · branch `fix/portal/PC-0831-013-btw`
├─ .claude/worktrees/template-impact/	session 2 · branch `feat/portal/PC-0902-021-template-impact`
│ └─ Portal.Vue/node_modules	a junction to one shared copy
└─ … 18 worktrees today	one folder per session

TIP FOR ANY STACK One main checkout per model vendor.

My agents

ACTIVE ON OCTOBER 5TH, 14:07

■ agent □ subagent

14 agents and 2 subagents

PLAN

Product management
■ ×1
picks the issues

Epic design
■ + □□ ×1 +2

→

BUILD & TEST

Feature coding
■■■ ×3

Bug fix coding
■ ×1

→

REVIEW & MERGE

Review agents
■■■■ ×4
polled every 15 min

→

VALIDATE & ACCEPTANCE

Test agents
■■ ×2
acceptance

Security review agent
■ ×1

→

MEASURE

Measure agent
■ ×1
reads production logs every day, files bugs

↺ new bugs flow back to product management

Define the task before the coding session starts

1 · ISSUE #1513

Invoice editor: two-step due date selector, with origin and standards #1513

Closed

softwareassist-coder opened on Aug 31

Continuation of #1482 / #1484 . That placed the expiration date in the attributes block, with a blank input field for the term next to it. This issue replaces that field with a two-step selector, following the "4b Two-Step Popover" design.

Complete design: `docs/superpowers/specs/2026-08-31-vervaldatum-kiezer-design.md`

Why

The payment term is a four-tier setting (fixed `14` → account default → customer default → this invoice), but the portal only sees the flattened number. As a result, "14" does not indicate whether that is your default, the agreement with this customer, or something someone once typed in manually. The selector makes that origin visible, displays the resulting date for each preference, and provides a single place to adjust the defaults yourself.

What happens

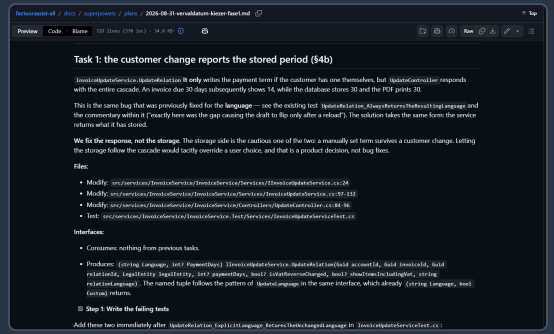
- **Step 1** — prefixes 8/14/30/60, supplemented with the account default, the customer default, and the current value (deduplicated, ascending). Each line shows the resulting date and where the value comes from.
- **Step 2** — Adjust the customer default and the account default. Do not touch the current invoice.
- **No more empty terms** in the editor. Existing empty terms are not filled, the API is not tightened, and the document path remains unchanged — see §4 of the draft.
- **The payment phrase loses its input field** and becomes running text again; the selector is the only editing location.

Adjacent error (found during the investigation)

When switching customers, `InvoiceUpdateService.UpdateRelation` the term is only recorded if the customer actually has one, whereas `UpdateController` the entire cascade reports back. An invoice that was set to 30 days subsequently shows 14, while the database stores 30 and the PDF prints 30. See §4b — the proposal is to fix the response, not the storage.

Why, evidence, phasing.

2 · PLAN



Task 1: the customer change reports the stored period (§4b)

`InvoiceUpdateService.UpdateRelation` **It only** writes the payment term if the customer has one themselves, but `UpdateController` responds with the entire cascade. An invoice due 30 days subsequently shows 14, while the database stores 30 and the PDF prints 30.

This is the same bug that was previously fixed for the **language** — see the existing test `UpdateRelation_AlwaysReturnsTheResultingLanguage` and the commentary within it ("exactly here was the gap causing the draft to flip only after a reload"). The solution takes the same form: the service returns what it has stored.

We fix the response, not the storage. The storage side is the cautious one of the two: a manually set term survives a customer change. Letting the storage follow the cascade would tacitly override a user choice, and that is a product decision, not bug fixes.

Files:

- Modify: `src/services/InvoiceService/InvoiceService/Services/IInvoiceUpdateService.cs:24`
- Modify: `src/services/InvoiceService/InvoiceService/Services/InvoiceUpdateService.cs:97-112`
- Modify: `src/services/InvoiceService/InvoiceService/Controllers/UpdateController.cs:84-96`
- Test: `src/services/InvoiceService/InvoiceService.Test/Services/InvoiceUpdateServiceTest.cs`

Interfaces:

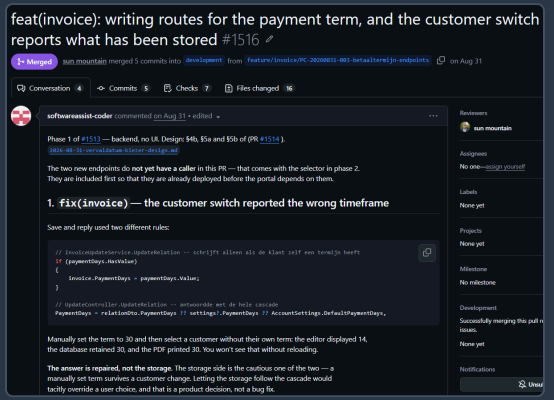
- Consumes: nothing from previous tasks.
- Produces: `(string Language, int? PaymentDays) IInvoiceUpdateService.UpdateRelation(Guid accountId, Guid invoiceId, Guid relationId, LegalEntity legalEntity, int? paymentDays, bool? isVatReverseCharged, bool? showItemsIncludingVat, string relationLanguage)`. The named tuple follows the pattern of `UpdateLanguage` in the same interface, which already `(string Language, bool Custom)` returns.

☐ **Step 1: Write the failing tests**

Add these two immediately after `UpdateRelation_ExplicitLanguage_ReturnsTheExchangedLanguage` in `InvoiceUpdateServiceTest.cs`:

Starts with “Step 1: write the failing tests”.

3 · PULL REQUEST #1516



feat(invoice): writing routes for the payment term, and the customer switch reports what has been stored #1516

Merged · sun mountain merged 5 commits into `development` from `feature/invoice/PC-20260831-003-betaaltermijn-endpoints` on Aug 31

Conversation 4 · Commits 5 · Checks 7 · Files changed 16

softwareassist-coder commented on Aug 31 • edited

Phase 1 of #1513 — backend, no UI. Design: §4b, §5a and §5b of (PR #1514).

`2026-08-31-vervaldatum-kiezer-design.md`

The two new endpoints do **not yet have a caller** in this PR — that comes with the selector in phase 2. They are included first so that they are already deployed before the portal depends on them.

1. `fix(invoice)` — the customer switch reported the wrong timeframe

Save and reply used two different rules:

```
// InvoiceUpdateService.UpdateRelation -- schrijft alleen als de klant zelf een termijn heeft
if (paymentDays.HasValue)
{
    invoice.PaymentDays = paymentDays.Value;
}

// UpdateController.UpdateRelation -- antwoordde met de hele cascade
PaymentDays = relationDto.PaymentDays ?? settings?.PaymentDays ?? AccountSettings.DefaultPaymentDays,
```

Manually set the term to 30 and then select a customer without their own term: the editor displayed 14, the database retained 30, and the PDF printed 30. You won't see that without reloading.

The answer is repaired, not the storage. The storage side is the cautious one of the two — a manually set term survives a customer change. Letting the storage follow the cascade would tacitly override a user choice, and that is a product decision, not a bug fix.

Reviewers: sun mountain · Assignees: No one—assign yourself · Labels: None yet · Projects: None yet · Milestone: No milestone · Development: Successfully merging this pull r… issues. · None yet · Notifications

Merged the same day. I approved it.

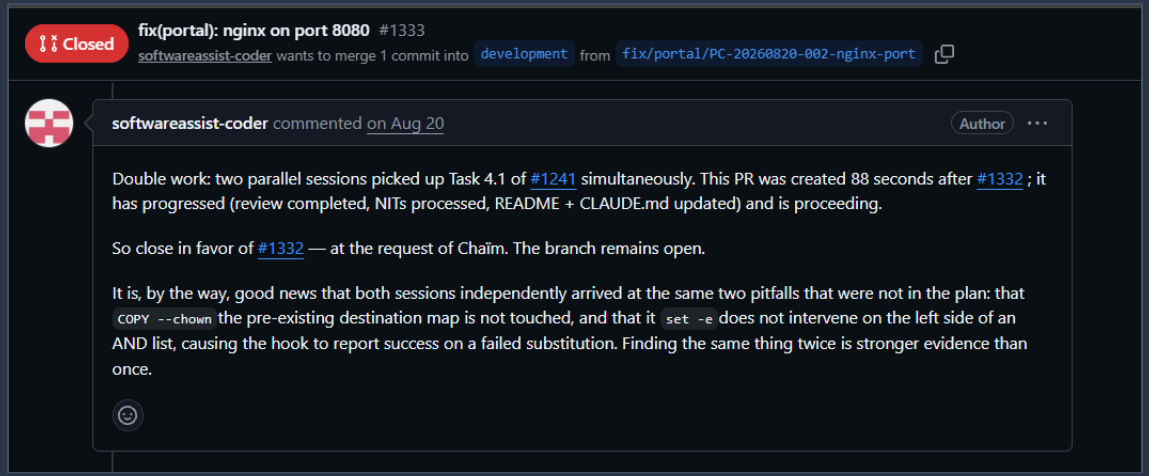
TIP FOR ANY STACK Acceptance criteria first, then the coding session.

The coordination layer is the Kanban board

- Every story has one visible phase, so no agent picks up work that is already taken
- Pull, not push: an agent only takes the next story when a column has room
- WIP limits on design, coding and review: the cap is on stories in a phase, not on agents

01 Backlog	02 Design 2 / 3	03 Coding & test 5 / 5	04 Review & verify 4 / 5	05 Ready for Acceptance test	06 Ready for Production Release
#1533 Dashboard: open invoices	#1502 Peppol impact analysis	#1345 VAT: btw-1345	#1769 Invoice PDF: logo position	#1513 Customer switch: editor 14, DB 30	#1516 PR for #1513, merged 15:35
#1536 Portal: dark mode	#1508 Template impact analysis	#1497 Invoice PDF: footer text	#1771 Payment link: expiry		
		#1521 Customer import: duplicates	#1774 Customer search: diacritics		
		#1524 Reminder e-mail: wording	#1776 VAT report: quarter totals		
		#1530 Export: date format			

Failure mode 1: two sessions, same task



Closed **fix(portal): nginx on port 8080** #1333

softwareassist-coder wants to merge 1 commit into `development` from `fix/portal/PC-20260820-002-nginx-port`

softwareassist-coder commented on Aug 20 — Author

Double work: two parallel sessions picked up Task 4.1 of #1241 simultaneously. This PR was created 88 seconds after #1332 ; it has progressed (review completed, NITs processed, README + CLAUDE.md updated) and is proceeding.

So close in favor of #1332 — at the request of Chaïm. The branch remains open.

It is, by the way, good news that both sessions independently arrived at the same two pitfalls that were not in the plan: that `COPY --chown` the pre-existing destination map is not touched, and that it `set -e` does not intervene on the left side of an AND list, causing the hook to report success on a failed substitution. Finding the same thing twice is stronger evidence than once.

- ✓ Check open pull requests and worktrees before starting
- ✓ The board says who has it
- ✓ Parallel sessions even picked the same sequence number for their branch ID

TIP FOR ANY STACK Two issues that look alike? Give them to one agent.

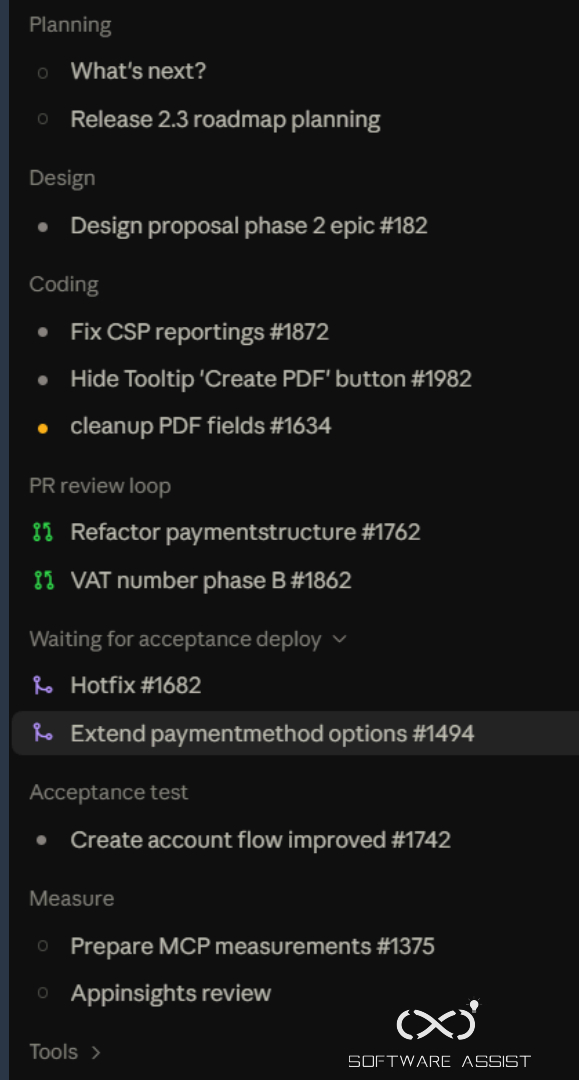
Failure mode 2: building on last month’s code

- While you plan, other sessions merge
- Its instructions come from that old code too

TIP FOR ANY STACK Fetch first; branch from origin, not your local main.

Open is not the same as working

- Measured over 30 days: 3 to 6 working, peak of 14
- The rest is waiting
- The bottleneck is my attention, not the agents



Design sessions to need me less often

- I approve design and plan; the session runs to a reviewed PR
- Bounded loop: every 15 minutes, five times at most
- Evidence in the PR description: one document, not the session



ME	ME	AGENT	AGENT + REVIEWER BOT	ME
Approve design	Approve plan	Build until PR	Review loop: every 15 min, max 5	Read PR evidence

Review: what I tried, and what I use now



Human review only	At 25 a day I became the queue.	DROPPED
Bot review on every PR	I read the summary, not the diff.	DEFAULT
Bounded loop	Every 15 minutes, five rounds at most.	DEFAULT
Cross-vendor review (Claude and Codex)	Both found the same blocker in round 3.	FOR DESIGN PRS

A bounded review loop, in practice

ROUND 1 · THE BOT REVIEWS

Merged **fix(invoice): MakeFinal no longer returns a plain 500 for an unnamed recipient** #1769

sun mountain merged 6 commits into `development` from `fix/invoice/PC-20260906-070-makefinal-without-definitive-pdf`

claude Bot commented on Sep 6

Claude Code Review

The chain in the PR description matches what I find in the code. Checked and confirmed:

- `GeneratorException` → 405 is indeed captured in `DocumentController.FinishInvoice` (`DocumentController.cs:80`); `SetStatusToDefinitiveAsync` has no other caller, so the change from `Guid` to `Guid?` on `GetFinalDocumentId()` is fully enclosed.
- `ValidateTo` provides `TO_INCOMPLETE` and `TO_NAME_INCOMPLETE` via a single `else if` chain, so those two can never appear together on an invoice — splitting `needsCustomer()` leaves no gap.
- `ValidateFrom` `OrganizationName` indeed makes a missing one a `AddWarning` (`ValidatorService.cs:107`), so the claim "the seller side is indeed reachable" under the new UBL guard is correct.
- `MapInvoice` calls `FindUnsupportedReason` before `FindMissingDataReasons` and returns there, so a consumer keeps `UBL_CONSUMER_CUSTOMER` as the only reason — exactly what the new test asserts.
- `BuildInvoice()` in `UblMapperTest` fills in both organization names, so the two new guards at the top `FindMissingDataReasons` do not break an existing one `Assert.Equal(new[] { ... }, Codes(invoice))` .
- The two new reason codes are not `DocumentGenerator.Ubl` consumed outside of anything and are tests — no translation table or `switch` one that silently gets a hole.
- `SingleOrDefault` cannot fall under the "more than one" category here: `BuildDocumentRequests` set exactly one PDF request per generation.

RECOMMENDED (1)

- **The validation codes arrive as `System.String[]` in the telemetry** — `src/services/InvoiceService/InvoiceService/Services/InvoiceManager.cs:251` — `{ValidationErrorCodes}` the `IReadOnlyList<string>` output receives `GetValidationErrorCodes()` (runtime a `string[]`). InvoiceService logs via `Microsoft.Extensions.Logging` + Application Insights (`ApplicationInsightsHelper.Initialize` , no Serilog), and it formats a parameter value with `ToString()` — both in the message and in `customDimensions` it then literally `System.String[]` appears . This causes exactly the information for which `GetValidationErrorCodes()` was added to be lost, and that log line is the only trace this rejection leaves behind. Fix: `string.Join(", ", generationResult.GetValidationErrorCodes())` , as `LoginService/Services/RegistrationProcessService.cs:40` and `FactuurAssist/Generic/Services/ReCaptchaService.cs:129` already do.

NIT (3)

- `editCustomer()` reads `invoiceStore.invoice.toRelationId` without `?.` , while `canEditCustomer` two lines above it `?.` does use — safe today because the `v-if` keeps the link away, but the two forms side by side read as a coincidence (

ROUND 4 · THE SESSION CLOSES THE CYCLE

Merged **fix(invoice): MakeFinal no longer returns a plain 500 for an unnamed recipient** #1769

sun mountain merged 6 commits into `development` from `fix/invoice/PC-20260906-070-makefinal-without-definitive-pdf`

softwareassist-coder commented last month — Author

Review cycle closed

Round 4 (after `06dd606`) yielded no BLOCKER, RECOMMENDED, or NIT. According to boundary 2 of the skill, `reviewfeedback-verwerken` a round without findings is the last one, so I stop the cycle here.

What has been processed over four rounds:

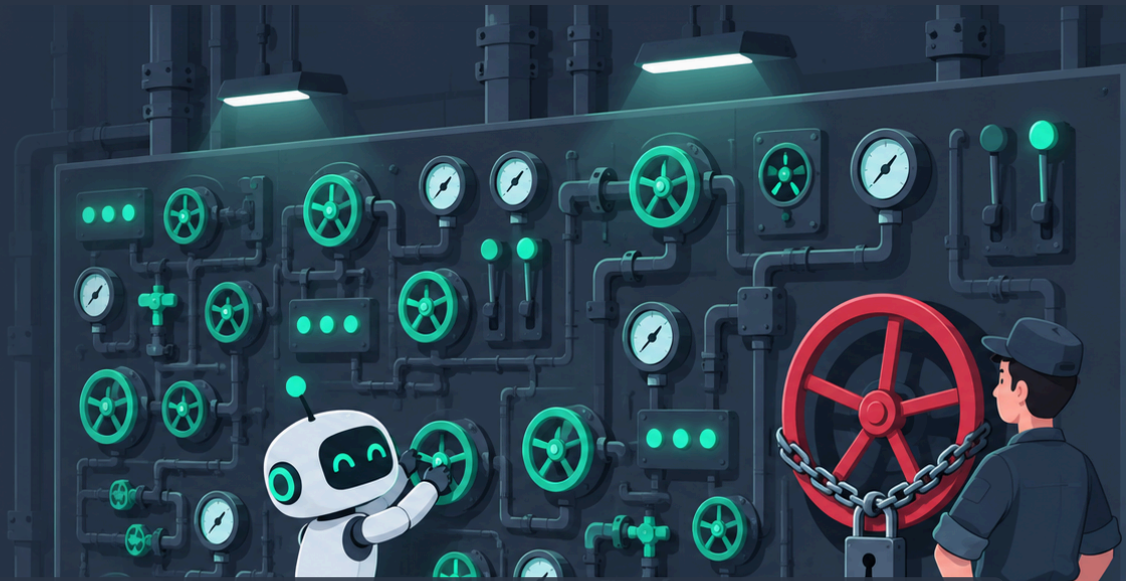
Round	Finding	Outcome
1	`System.String[]` in the telemetry (RECOMMENDED)	`string.Join` - incorporated
1	missing `?.` , BOM on five files (NIT)	incorporated
2	burnt invoice number unattended (RECOMMENDED)	processed, with `Times.Never` instead of the proposed `Assert.Empty` — measured that the latter is vacuum
2	dubbele fixture (RECOMMENDED)	`InvoiceScenarios.WithNamelessRecipient()`
2	`editCustomer` -duplication, two ID sources (NIT)	rejected with reason
3	own pinia setup instead of `withInvoiceStore` (RECOMMENDED)	incorporated
4	—	no

Not a single round brought a finding on code that a previous round had added, so the cycle did not empty on its own.

Ready to merge as soon as the ADO checks are green. After the merge, #1755 goes to `Ready for Acceptance test` the board — closing only happens after the acceptance test.

Let agents work undisturbed

- A permission prompt parks the session until I look
- Decide the routine up front: allow lists, sandbox, auto mode
- Interrupt only for destructive actions, production, security



RUNS ON ITS OWN

`build · test · lint · format`

`read code · search · edit in its worktree`

STOPS FOR ME

`delete · force-push · migrate data`

`production · secrets · security settings`

TIP FOR ANY STACK Deny by default; grant permissions per task.

Decided up front, in `settings.json`

- Allowed without asking: navigating, searching the code, web search
- Everyday git: add, branch, checkout, commit, fetch, merge, pull
- A hook checks every shell command, before and after it runs

settings.json

C: > Users > caezo > source > Invullen > worktrees > invullen-all > .claude > {} settings.json > ...

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "command",
            "command": "powershell.exe -ExecutionPolicy Bypass -NoProfile -File .claude/hooks/pre-bash.ps1"
          }
        ]
      }
    ],
    "PostToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "command",
            "command": "powershell.exe -ExecutionPolicy Bypass -NoProfile -File .claude/hooks/post-bash.ps1"
          }
        ]
      }
    ]
  },
  "permissions": {
    "allow": [
      "Bash(cd:*)",
      "Bash(find:*)",
      "WebSearch",
      "Bash(git add:*)",
      "Bash(git branch:*)",
      "Bash(git checkout:*)",
      "Bash(git commit:*)",
      "Bash(git fetch:*)",
      "Bash(git log:*)",
      "Bash(git merge:*)",
      "Bash(git pull:*)",
```

Limit my queue, not the number of sessions

- In July my rule was: at most 5 sessions
- What matters now: what waits for me
- Today: at most 3 focus epics, 10 issues around them



My attention runs out before the agents do

- Brainfry: early mornings take more sessions than late afternoons
- Pick stories by energy. Fresh? design; Tired? bug fixes
- Fewer context switches takes planning ahead: batch reviews, work per epic, pick up only when ready



One good move per board

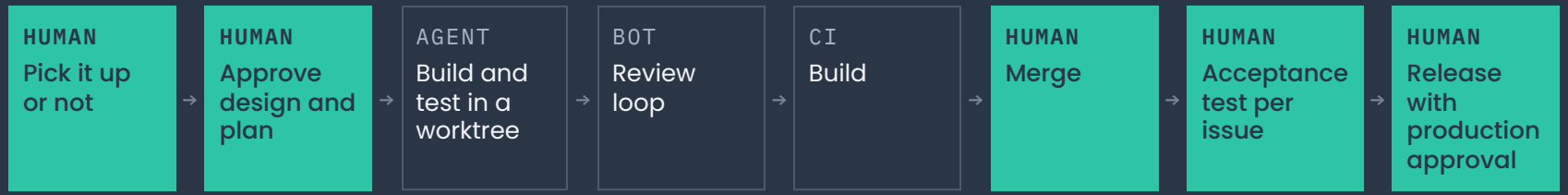
- A simultaneous chess exhibition: not all boards at once
- Every board must be in a state where it can wait
- Approvals turning into reflexes? Lower the WIP limits.



TIP FOR ANY STACK

Before a long task, let the other agents run ahead: that keeps throughput up.

Where I still look: five gates



THROUGHPUT, AUGUST 2026

770

pull requests merged in August

~34

merged per working day

937

of the latest 1,000 opened by the agent account

Where the testing happens

IN THE SESSION · LOCAL

The failing test comes first

→

ON EVERY PULL REQUEST · CI

Full suites: ~3,300 service tests, ~2,400 front-end tests

→

ON ACCEPTANCE · AFTER DEPLOY

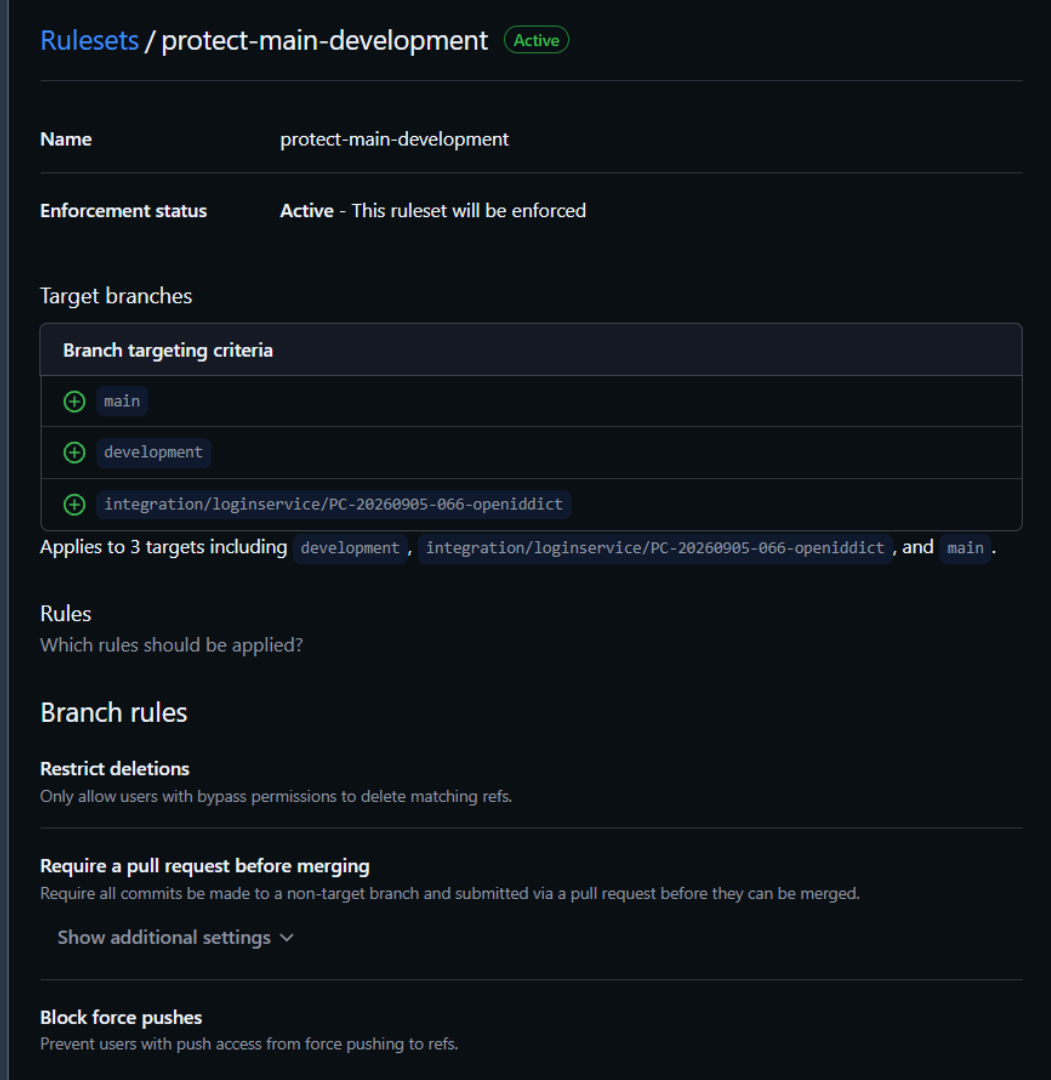
I log in; the agent never holds my credentials

VAT REGRESSION MATRIX, RUN ON ACCEPTANCE, 15 SEPTEMBER

67	61	2	4
scenarios	passed	product bugs found	blocked by missing test data

Enforced by GitHub

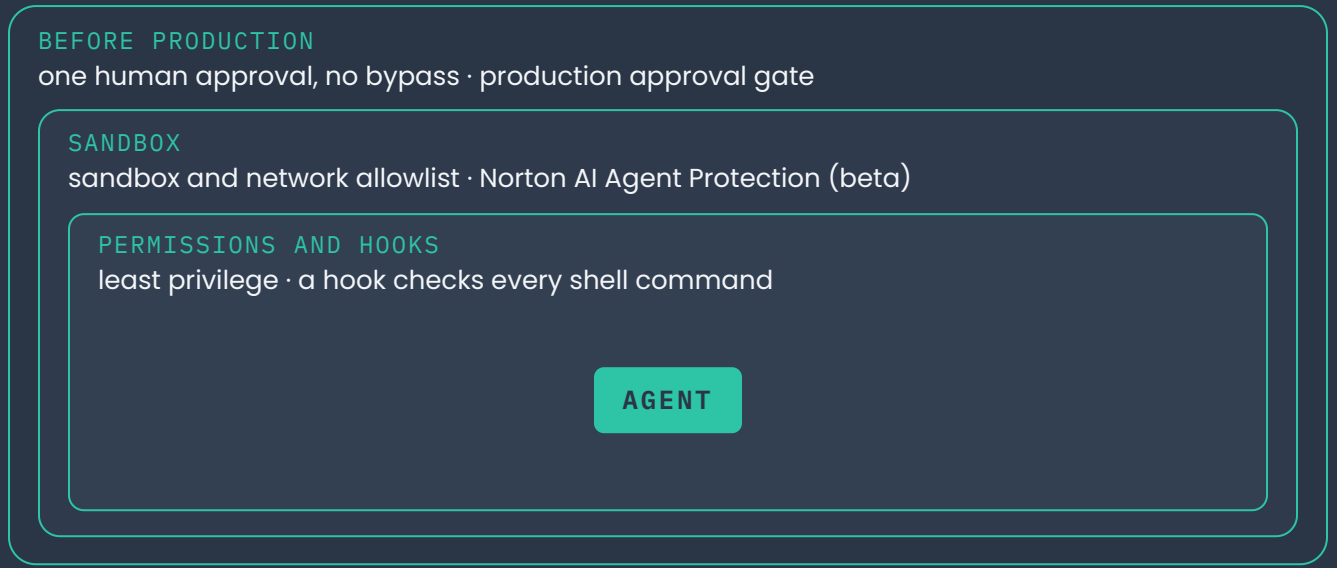
- One ruleset protects main, development and the integration branch
- Every change goes through a pull request; a merge needs one approving review, with no bypass
- No force pushes; deletions are restricted



Safe to run: fences around the agent

SOME OF THE RISKS

- ! too many rights →
- ! destructive commands →
- ! data leaving →
- ! injected instructions →
- ! merging its own work →



Safe to ship: what the code must pass

SOME OF THE RISKS ! injection ! authorization ! secrets in code ! ‘fixed’ security ! vulnerable dependencies ! misconfiguration … OWASP Top 10

01

RULES IN THE CONTEXT

tenant from token · policy per endpoint · secrets in vault

→

02

BOT REVIEW ON EVERY PR

security finding = BLOCKER

→

03

HUMAN REVIEW

security-sensitive areas

→

04

BEFORE PRODUCTION

full security review of dev → main

Fast today. Maintainable next year?

THE RISK

! Ten agents, hundreds of locally logical choices

WHAT HOLDS THE LINE

✓ Conventions as rules per file type

✓ Search before you write; one issue, one PR, about 500 lines

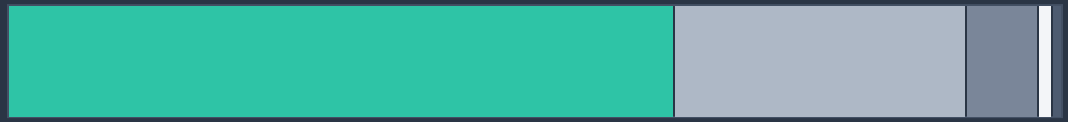
✓ Every plan starts with a failing test

✓ Instructions are code: a budget, tests, deletion

TIP FOR ANY STACK A mistake that could repeat? Write it where sessions read.

Parallel can also become a token bill

COST AT API LIST PRICE, AUGUST 2026



≈ $17,250

- Cache reads $11,250 (65%)
- Cache writes 1-hour $4,500
- Output $1,290
- Cache writes 5-min $190
- Uncached input $1

$200 / month

paid: Claude Max 20x subscription

≈ $17k

same month at API list price

- 65% is re-reading history; cost grows quadratically with turns

What works, what doesn’t, and where I stay essential

WHAT WORKS

- ✓ One issue, one worktree, its own pull requests
- ✓ Review loops that run without me
- ✓ Mistakes become rules, skills, hooks or memories

WHAT DOESN’T (YET)

- ! Two sessions picking up the same task
- ! A context file describing last month’s portal

WHERE I STAY ESSENTIAL

- → What to build
- → Architecture and security trade-offs
- → What “done” means

How to start: guardrails first, speed later

1 · GUARDRAILS

Quality, security, context, maintainability

Tests on every PR · security rules in the context · conventions written down · small PRs

→

2 · CLEAN UP

Legacy out, tech debt down

Every shortcut in the codebase is one the agent will copy

→

3 · THEN THROUGHPUT

More sessions, fewer bottlenecks

Only now does parallel pay off

At first it only costs you time. That is normal.

Take home, on any stack

- One worktree and branch per session; fetch and reset its base before the first change
- Set permissions up front, so routine work never waits for you
- Cap what waits for you, not the number of sessions: batch reviews, keep each ask small

Start with two sessions, not ten. The habits are the same. Then keep looking for the next bottleneck: where is the idle waiting, and how do you speed it up responsibly?

Discussion

QUESTIONS, EXPERIENCES, PUSHBACK

Questions?

DOWNLOAD SLIDES · CONNECT

Chaïm Zonnenberg

SOFTWARE ASSIST