

Onboarding Your Agent

How eventim integrates agentic engineering into enterprise workflows

Fabian Meyer

Chapter Lead **Vue** @ eventim Tech

whoami

Chapter Lead **Vue** @ eventim Tech

- Leading our Vue chapter - standards, tooling, developer experience
- Tech Lead for the eventim.Tixx ticket online shop
- Passionate about agentic engineering

off-work: gym, Formula 1, 3D printing, and way too many homeserver projects

eventim & Tixx

eventim is Europe's leading ticketing provider.

eventim.Tixx is our ticketing platform for sports & festivals: 200+ customers, PHP & Vue frontends, legacy core, multi-team GitLab setup.

Starting with AI

The ChatGPT era

- Talks with data privacy and IT security
- ChatGPT as a replacement/companion for Stack Overflow and Google
- Copy-paste code snippets, glue them in ourselves

In between came Copilot, autocomplete on steroids.

Codex & Claude Code

- Code-aware tools with real context
- Plan mode → implement → coding at inference speed
- Fixed bugs, built small features, refactors, documentation

Faster code, low impact - what if coding was never the bottleneck?

The agent writes code. You do everything else.

- × You read the Jira ticket and translate it into a plan
- × You let Claude implement the plan and test the result yourself
- × You create a merge request and assign reviewers
- × You babysit the pipeline: linting, code style, unit testing, security scans
- × You have to deploy on a dev/stg instance for qa
- × An e2e test is failing
- × ... and so much more

Your process requires a human clicking on buttons.

In enterprise, the process *around* code is where you get stuck.

Code is generated fast. Getting through the process? Still slow.

"So let's just throw out the process, right?"

It's enterprise. You can't - and you shouldn't.

- Human-in-the-loop for code review and QA
- Standards, guardrails and security
- We still need to deploy on complex infrastructure

It's not overhead - it's the reason things don't break in production.

But agents can remove *a lot* of the bottleneck.

How do we get humans into the process?

We onboard them

What if we did the same for our agent?

The Onboarding Checklist

New Engineer - Day 1

- GitLab access
- Jira board access
- Deploy permissions
- Process documentation
- Pair with a colleague

New Agent - Day 1

- `glab` CLI
- `jira` CLI
- Deploy instructions
- Process instructions
- Human-in-the-loop (= pairing)

Your process stays the same. The agent just learns to follow it.

Teach The Skills

Tools

- Equip the agent with your dev tools: GitLab CLI, Jira CLI, Playwright CLI
- Vendor and community skills exist for common tools, use them as a starting point.
- Always read third-party skills before installing. Know what they do, don't be scared to change.

Process

- Once the tools are in place, teach the agent how your team ships software.
- Spec-Driven Development (SDD): from reading the requirements to shipping working code.
- There are a lot of good frameworks out there, get inspired and build around your process.

The keyword here is skills. Literally.

Agents Skills



A markdown file

SKILL.md in `.claude/skills/<skill-name>/` .



References & scripts

Optional directories for additional context and deterministic behavior.



Skill knowledge, packaged

Instructions, constraints, and examples for a specific workflow kept short to avoid bloat.



Iterate on them

Skills are living documents. Review the results, refine the instructions, and keep improving.

Why not CLAUDE.md? Great for project conventions and general rules.

Why not MCP? Excellent for external integrations, but more infra than cli tools & team workflows need.

Spec-Driven Development

SDD frameworks exist but build your own around your process. We came up with this:



Spec

Fetch the **Jira** ticket, research the codebase, write a specification together.



Build

Implement with TDD, iterate on failures, commit per task, push, run local checks



Verify

Open a draft merge request with **glab**, pass CI, assess against spec, self-review.



QA

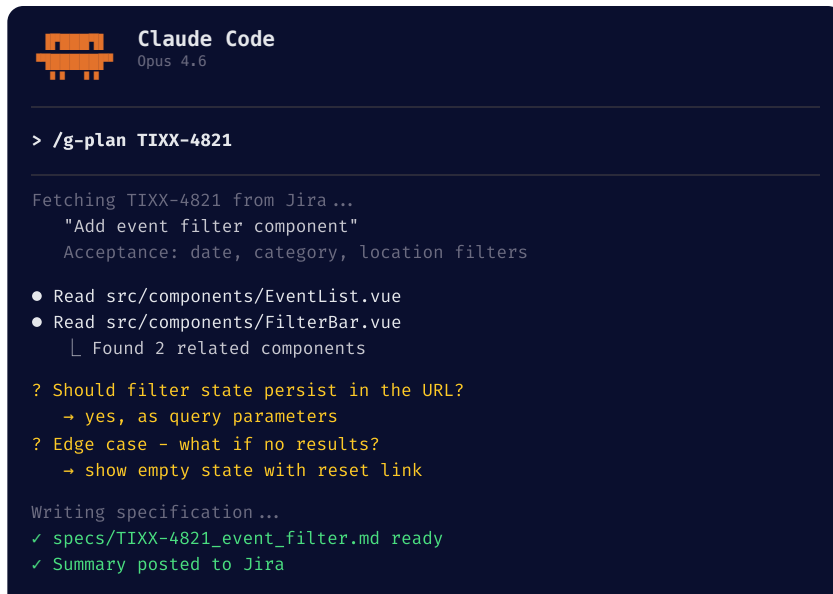
Deploy on dev/stg, run manual tests with **playwright-cli**, take screenshots.

Our SDD framework. Four phases - four skills.

Spec it together

- Agent fetches the **Jira ticket** and reads the acceptance criteria and description
- Explores the codebase for related components and patterns
- Runs an interactive Q&A with you, asks clarifying questions, proposes options
- Writes a structured specification before any code changes
- It contains research, constraints, requirements, decisions, tasks and verification criteria
- Post summary as Jira comment

The spec should be reviewed carefully. It becomes the contract between you and the agent.



```
Claude Code
Opus 4.6

> /g-plan TIXX-4821

Fetching TIXX-4821 from Jira...
"Add event filter component"
Acceptance: date, category, location filters

• Read src/components/EventList.vue
• Read src/components/FilterBar.vue
  | Found 2 related components

? Should filter state persist in the URL?
  → yes, as query parameters
? Edge case - what if no results?
  → show empty state with reset link

Writing specification ...
✓ specs/TIXX-4821_event_filter.md ready
✓ Summary posted to Jira
```

Build it iteratively

- Agent reads the spec and loads project conventions
- Agent moves ticket to "In Development" and creates a branch to our conventions
- Agent implements task-by-task
- Writes failing tests first **RED**
- Implements until pass **GREEN**
- Iterate until lint, type-check, unit test all pass
- One conventional commit per task, pushes to remote

```
Claude Code
Opus 4.6

> /g-dev TIXX-4821

• Read specs/TIXX-4821_event_filter.md
• Read CLAUDE.md
  └─ Loaded spec + conventions

Jira TIXX-4821 → "In Development"
✓ Branch: g/TIXX-4821_event-filter

T1: FilterBar component
RED
Writing FilterBar.spec.ts ...
✗ 3 tests failing (component not found)
GREEN
Writing FilterBar.vue ...
✓ Tests: 3/3 passed
✓ Lint: passed
→ feat(filter): T1 - add FilterBar

✓ Pushed to origin/g/TIXX-4821_event-filter
```

No interaction required until all tasks are complete - Iterate, commit, repeat.

Verify against the spec

- Agent creates a draft merge request with **glab**
- Completeness check - all tasks marked done, all commits are there. Only files in spec were modified
- Validates implementation against spec requirements and acceptance criteria
- Self-review - spawns a fresh agent and review the MR and leaving comments
- Monitor MR pipeline. On failure, inspect the output, fix, repeat until pass
- Remove draft from MR, move the Jira ticket to "Ready for review"

Ready for human review.

```
Claude Code
Opus 4.6

> /g-verify TIXX-4821

• Bash glab mr create --draft --title "TIXX-4821: Event filter"
  └ MR !312 opened (draft)

Completeness: 4/4 tasks ✓
Requirements: 3/3 must-haves MET ✓

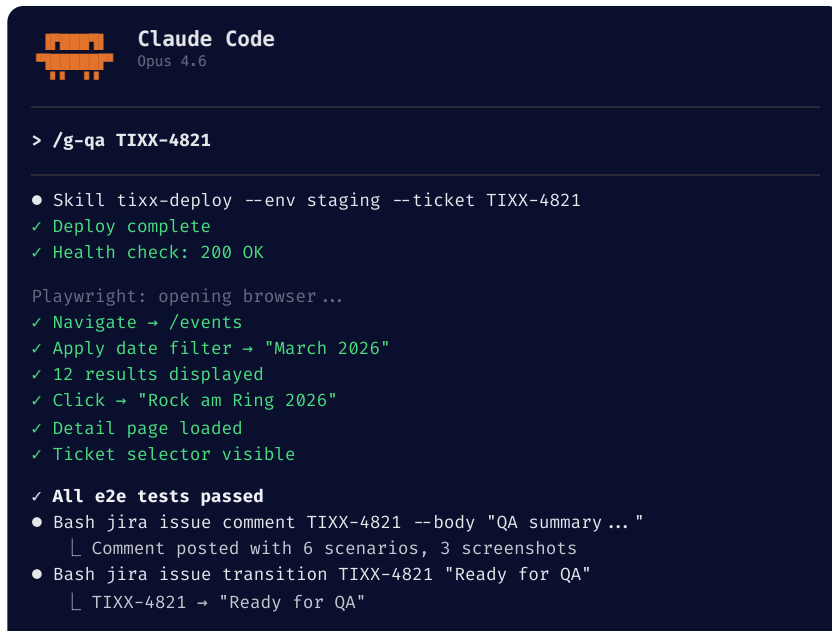
Self-review (fresh agent):
  → "Missing null check on line 47"
  Fixing... pushed

CI: lint ✓ | unit ✓ | build ✓
✓ Pipeline: green

• Bash glab mr update 312 --ready
  └ MR !312 ready for review
• Bash jira issue transition TIXX-4821 "Ready for Review"
  └ TIXX-4821 → "Ready for Review"
```

Deploy and verify

- Agent triggers ask for deploy target and applications, then deploys with `/tixx-deploy`
- Monitor deployment until it's completed, investigate on failure.
- Run `/playwright-cli` to access the deployed environment
- For each scenario navigate to the feature, interact, verify behavior
- Record scenarios, results and screenshots, then attach a summary to the ticket
- Transition ticket to "Ready for QA"



```
Claude Code
Opus 4.6

> /g-qa TIXX-4821

• Skill tixx-deploy --env staging --ticket TIXX-4821
✓ Deploy complete
✓ Health check: 200 OK

Playwright: opening browser ...
✓ Navigate → /events
✓ Apply date filter → "March 2026"
✓ 12 results displayed
✓ Click → "Rock am Ring 2026"
✓ Detail page loaded
✓ Ticket selector visible

✓ All e2e tests passed
• Bash jira issue comment TIXX-4821 --body "QA summary ..."
  | Comment posted with 6 scenarios, 3 screenshots
• Bash jira issue transition TIXX-4821 "Ready for QA"
  | TIXX-4821 → "Ready for QA"
```

The human makes the final qa - but the agent has already done all the legwork.

What About Us?

Now that the agent is onboarded - what happens to us?

Before:

- Language mastery
- Framework expertise
- Project experience
- Debugging skills

Now:

- Product sense - understanding *what* to build
- UX & design instinct - knowing what *feels* right
- Architecture decisions - system-level thinking
- Business impact - connecting work to outcomes
- Spec writing - your primary output
- Review & QA judgment - your quality gate

Coding yourself has become inefficient.

You went from software engineer to generalist. IMO that's a superpower.

Everyone Gets an Agent

If we can onboard an agent as an engineer - why stop at engineering?

Product Manager

- Uses the agent to translate business needs into tickets with technical knowledge on hand
- Finally can find that *one* ticket in Jira because the agent can do JQL

UX Designer

- Can prototype directly in application code - sees the real thing, not a mockup. Devs add logic

QA

- Writes test specifications, the agent executes, reports and documents

The same pattern works for everyone.

Give them access. Teach them the process. Help them automating.

The roles don't disappear - they evolve.

Start Monday

Pick your most blocking process

For us this was deploying, since our local development is limited in some areas.

Create it with your agent

A skill file describing how *you* do it. Your steps, your tools, your constraints.

Let the agent run it. Refine. Repeat.

Try it. It won't be perfect the first time. That's fine. Iterate on the skill like you iterate on code.

One skill. One afternoon. That's all it takes to up the impact of your agents.

**"The best engineers won't write the best code -
they'll write the best specs."**

"And so will everyone else."
