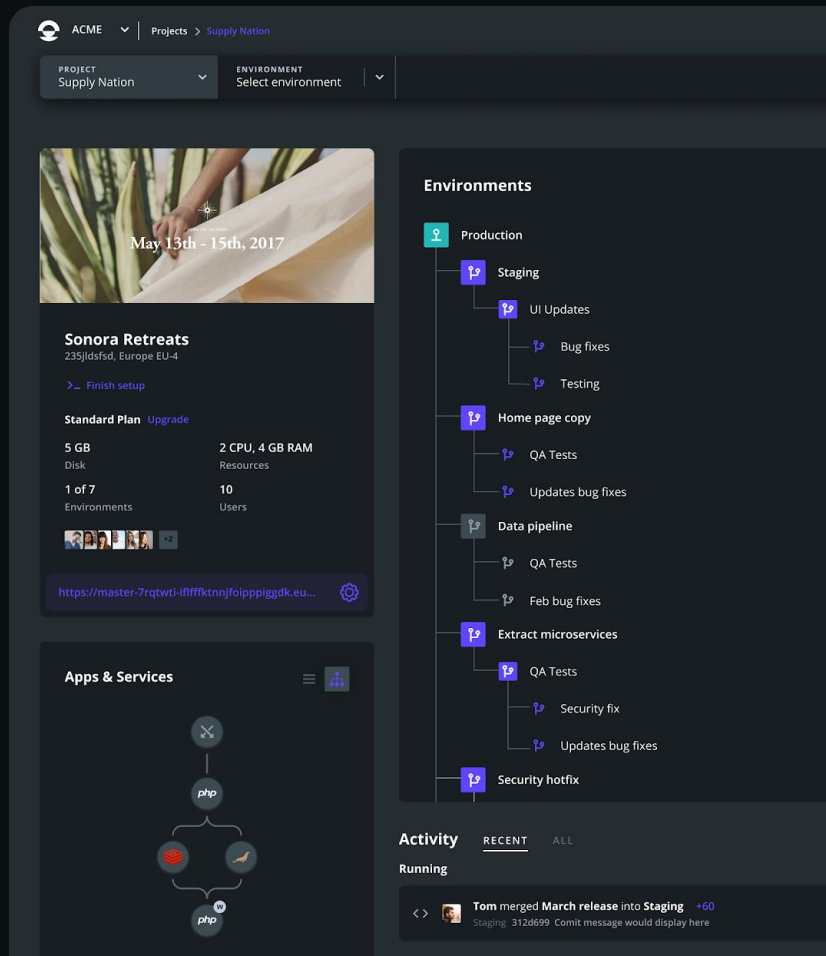


Vibe-free zone: serious AI assisted development for Symfony

How to leverage AI assistants
the right way



The screenshot displays the Upsun dashboard for a project named "Supply Nation". The interface is dark-themed and includes a top navigation bar with the project name and a dropdown for "Select environment".

Environments

- Production
 - Staging
 - UI Updates
 - Bug fixes
 - Testing
 - Home page copy
 - QA Tests
 - Updates bug fixes
 - Data pipeline
 - QA Tests
 - Feb bug fixes
 - Extract microservices
 - QA Tests
 - Security fix
 - Updates bug fixes
 - Security hotfix

Apps & Services

A diagram shows a central "php" node connected to a "redis" node and a "php" node with a "3" badge. Below this, another "php" node is shown.

Activity

Running

Tom merged March release into Staging - 60s
Staging 312d699 Commit message would display here

whoami



Guillaume Moigneu

Aka G/

guillaume.id

- Love to tinker with stuff. Code, cameras, watches.
- I know own 4 ugly sweaters
- Love buying domain names for unfinished side projects.
- Started PHP in 98. Started using SF1 in 2003.
- Vibe coder since 2023



The concepts, the tools & the capabilities are changing **exceptionally fast**, faster than anything we've seen before

So let's focus on the **guiding principles** and the practices instead of the actual tools.

Time to develop some
`symfony/ai` stuff.

Quick survey!



Types of Assistants and Agents

- Chat-based (ChatGPT, Gemini, Claude Desktop, Super Grok)
- CLI/IDE Agent (Claude Code, Cursor, Windsurf, Kiro, Cline, etc)
- Background Agent (Codex, Jules, Claude Code + GitHub integration, Cursor Background Agents, Devin, OpenHands, etc)
- Agent Swarm (Factory, Codex - in part, custom workflows)
- Non-Engineering (ChatGPT Agent, Operator, Deep Research, Manus, etc)

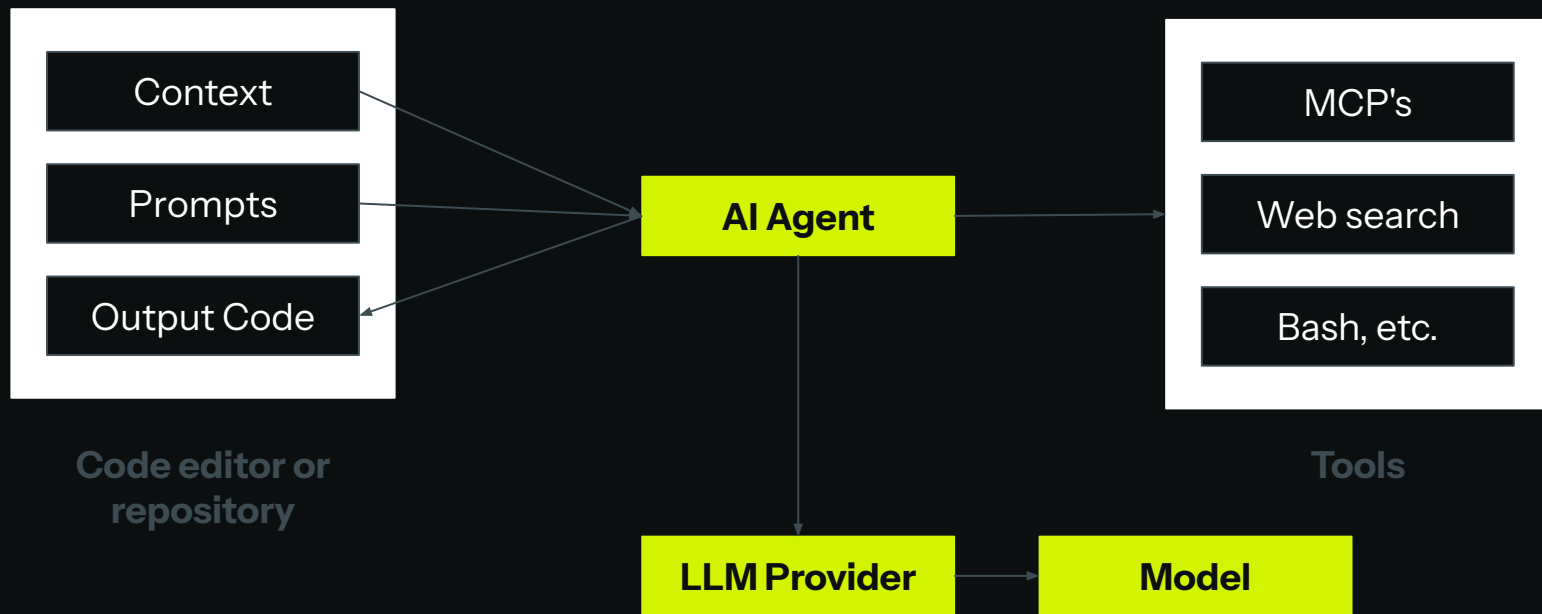


Most of the time, it is similar to dealing with an overly confident, exceptionally eager, **inexperienced** junior developer



How can we make it more like
a cooperating **rockstar developer**?

How AI Assistants and Agents work and what do they need to give great answers?



Setting up your project the right way!
Context.

Feeding our agent with the right knowledge (1)

First, let's talk about our project

- How is our project **architected**? Technology stack, components, etc.
- What is our project doing overall? What are the **specifications**?
- How do we **build, run and test** our project?
- Are there any external dependencies, endpoints, etc.

Just imagine having a new hire and letting them run the project without any help from anyone else.

Feeding our agent with the right knowledge (2)

Second, let's talk about us (or our team)

- Do we have **coding preferences**? (tabs vs. spaces, comments, etc.)
- Are there any specific **architecture** choices that matter?
- What are the **naming conventions**?
- How do we document the project?
- Are there libraries we want (or not) to use

Again, what would you expect from a new hire when you are doing their code review?

How to store this knowledge?

- Anything project related should be in the classic **README .md**
- Any team or personal preferences should be store in an **AGENTS .md** file

If you prefer organizing the knowledge in subfolders, keep the two **md**'s and then create a **knowledge** and **rules** folder (just my preference). This makes it easier to copy to another project.


*If using Claude Code, just symlink **CLAUDE .md** to **AGENTS .md**. Or create **sub-agents**.*


AGENTS.md


agents.md


One AGENTS.md works across many agents


Your agent definitions are compatible with a growing ecosystem of AI coding agents and tools:


**Codex**
from OpenAI


**Amp**


**Jules**
from Google


**Cursor**


**Factory**


**RooCode**


**Aider**


**Gemini CLI**
from Google


**Kilo Code**


**opencode**


**Phoenix**


**Zed**


**Simgrep**

**Warp**

**Coding agent**
from GitHub Copilot

**VS Code**

**Ona**

**Devin**
from Cognition

<https://jules.google>

Sub-agents

- Move the rules context to a specialized agent
- Can run in parallel for increased output speed
- Maintain and reuse in multiple projects

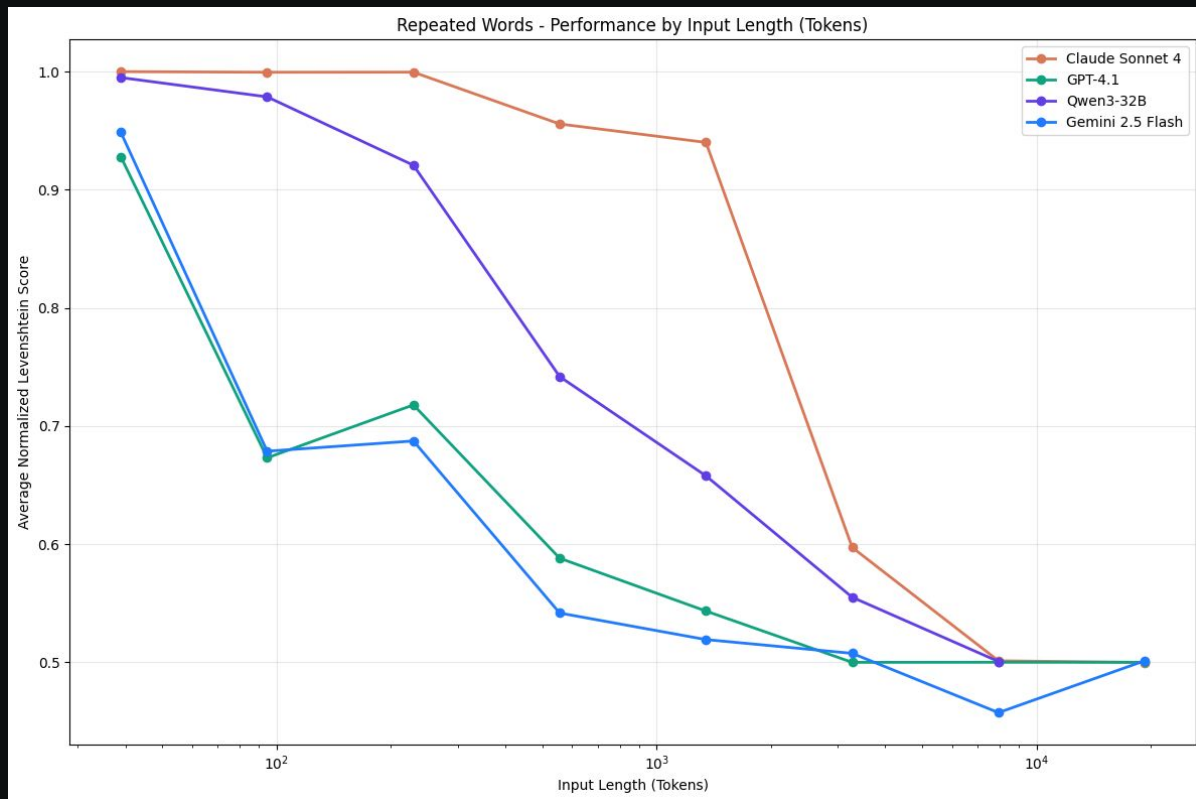
MCP's and tools

- Context7 can pull any documentation but can overwhelm the context
 - Chrome-MCP automates a local Chrome instance for testing
 - Upsun. Obviously.
 - Postgres, etc.
-
- WebFetch, WebQuery, Bash, etc.

Too much
knowledge is
bad: **context
rot**

aka. the needle in a
haystack

Source:
<https://research.trychroma.com/context-rot>



Review and monitor your context

```
> /context
```

```
|
```

Context Usage



claude-sonnet-4-5 • 112k/200k tokens (56%)

System prompt: 3.0k tokens (1.5%)

System tools: 15.2k tokens (7.6%)

MCP tools: 47.7k tokens (23.9%)

Custom agents: 520 tokens (0.3%)

Memory files: 468 tokens (0.2%)

Messages: 8 tokens (0.0%)

Free space: 88k (44.0%)

Autocompact buffer: 45.0k tokens (22.5%)

Maintaining specifications of the projects

When developing new features, start by always writing the specifications in the knowledge folder.

Maintaining both product and technical specifications of the project helps with:

- Avoid conflicts when developing new ones. Also bugfixes
- Automate documentation
- Allows easy client / ports / services development

Write the specifications for the SwiftUI client.

Let's review some actual stuff.

Is my MCP ready ???

> How many disasters happened in 2023?

```
L [
```

```
{
```

```
  "year": "2018",
```

```
  ... +27 lines (ctrl+r to expand)
```

- 7,139 disasters happened in 2023 🍌

> Show me the 3 deadliest earthquakes in Japan

- **disasters - query_disasters (MCP)**(query: "Show me the 3 deadliest earthquakes in Japan")

```
L {
```

```
  "success": true,
```

```
  "query": "Show me the 3 deadliest earthquakes in Japan",
```

```
  ... +51 lines (ctrl+r to expand)
```

- Oui oui 🍌 Here are the 3 deadliest earthquakes in Japan from the database:

1. **December 8, 2023** - 313 casualties, severity 8.85

2. **July 24, 2023** - 305 casualties, severity 7.15

3. **May 7, 2022** - 304 casualties, severity 10.0 (maximum severity)

The most severe earthquake (May 2022) caused the highest economic loss at \$11.6M USD 🇫🇷

```
> |
```

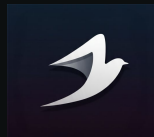
tmux				12:01
0 0 1 clicking [2] 2 layout refactor [4] 3 image null object pattern [2] 4 roboslop [4] 5 async python [2] 6 [16] 7 [8] 8				
<div><div>I've responded to the review. The reviewer gave an LGTM with no changes requested, so the PR is ready to be merged. The response acknowledges their valid observations about minor semantic mismatches while explaining the pragmatic trade-offs made.</div></div>	<div><div>server operations (not just connections)</div><div>Both are pragmatic choices given the existing error taxonomy.</div><div>PR URL: https://github.com/paradigmxyz/jtool/pull/222</div></div>	<div><div>- Default server selection</div><div>- First available server fallback</div><div>- Offline mode when no servers configured</div><div>- Ephemeral server creation from port number</div><div>All CI checks (Clippy, Format, Tests, Integration Tests, Doc Tests, Docs, Security Audit) pass.</div></div>	<div><div>Consider adding a unit test for substring matching in a follow-up.</div><div>PR URL: https://github.com/paradigmxyz/jtool/pull/224</div></div>	
<div><div>Model: Opus 4.5 Ctx: 145.5k Context left until └fix/issue-...</div><div>-- INSERT-->> accept edits on (shift+tab to cycle)</div></div>	<div><div>Model: Opus 4.5 Ctx: 43.4k └main cwd:...</div><div>-- INSERT -->> accept edits on (shift+tab to cycle)</div></div>	<div><div>Model: Opus 4.5 Ctx: 128.2k └fix/issue-...</div><div>-- INSERT -->> accept edits on (shift+tab to cycle)</div></div>	<div><div>Model: Opus 4.5 Ctx: 32.7k └main cwd...</div><div>-- INSERT -->> accept edits on (shift+tab to cycle)</div></div>	
<div><div>3. Responded to the review on GitHub with a comment explaining the changes and addressing the other non-blocking points.</div><div>All CI checks pass. PR #227 is ready for merge: https://github.com/paradigmxyz/jtool/pull/227</div></div>	<div><div>Verdict: Approved - ready to merge</div><div>PR URL: https://github.com/paradigmxyz/jtool/pull/227</div></div>	<div><div>verbose if-let boilerplate at each call site</div><div>All CI checks pass. The PR is ready for re-review at https://github.com/paradigmxyz/jtool/pull/230</div></div>	<div><div>gh pr diff 230</div><div>Get updated PR diff</div><div>Do you want to proceed?</div><div>> 1. Yes</div><div>2. Yes, and don't ask again for gh pr diff commands in /Users/storm/repos/jtool-rust</div><div>3. Type here to tell Claude what to do differently</div><div>Esc to exit</div></div>	
<div><div>Model: Opus 4.5 Ctx: 115.9k └fix/issue-...</div><div>-- INSERT -->> accept edits on (shift+tab to cycle)</div></div>	<div><div>Model: Opus 4.5 Ctx: 44.7k └main cwd:...</div><div>-- INSERT --</div></div>	<div><div>Model: Opus 4.5 Ctx: 60.1k └fix/issue-1...</div><div>-- INSERT -->> accept edits on (shift+tab to cycle)</div></div>		
<div><div>The review was an LGTM approval from sllivoff. No changes needed - the PR is approved and ready to merge. I responded thanking them for the review.</div></div>	<div><div>text_content() method.</div><div>PR URL: https://github.com/paradigmxyz/jtool/pull/220</div></div>	<div><div>The reviewer's verdict was "Ship it" - no changes requested. I've acknowledged the review with a thank you comment. The PR is ready to merge.</div></div>	<div><div>PR URL: https://github.com/paradigmxyz/jtool/pull/219</div></div>	
<div><div>Model: Opus 4.5 Ctx: 70.2k └fix/depreca...</div><div>-- INSERT -->> accept edits on (shift+tab to cycle)</div></div>	<div><div>Model: Opus 4.5 Ctx: 26.7k └main cwd:...</div><div>-- INSERT -->> accept edits on (shift+tab to cycle)</div></div>	<div><div>Model: Opus 4.5 Ctx: 108.9k └fix/issue-...</div><div>-- INSERT -->> accept edits on (shift+tab to cycle)</div></div>	<div><div>Model: Opus 4.5 Ctx: 46.4k └main cwd...</div><div>-- INSERT -->> accept edits on (shift+tab to cycle)</div></div>	
<div><div>user-facing output.</div><div>3. Test return types - The reviewer confirmed the std::result::Result qualification is the correct fix.</div><div>The PR is approved and ready to merge. The review comment is at: https://github.com/paradigmxyz/jtool/pull/229#issuecomment-357609167</div></div>	<div><div>self.to_string() usage in one error path and the loss of underlying parse error details in .map_err(_ ...) patterns, but neither affects correctness.</div><div>PR URL: https://github.com/paradigmxyz/jtool/pull/229</div></div>	<div><div>- Tests are comprehensive</div><div>- The code duplication (~50 lines) is an acceptable trade-off for this use case</div><div>- All CI checks pass</div><div>No code changes were requested, so the PR is ready to be merged.</div></div>	<div><div>versions, but this is a common Rust pattern and acceptable for this optimization.</div><div>PR URL: https://github.com/paradigmxyz/jtool/pull/222</div></div>	
<div><div>Model: Opus 4.5 Ctx: 125.2k └issue-34-r...</div><div>-- INSERT -->> accept edits on (shift+tab to cycle)</div></div>	<div><div>Model: Opus 4.5 Ctx: 35.9k └fix/issue-1...</div><div>-- INSERT --</div></div>	<div><div>Model: Opus 4.5 Ctx: 84.9k └fix/issue-2...</div><div>-- INSERT -->> accept edits on (shift+tab to cycle)</div></div>	<div><div>Model: Opus 4.5 Ctx: 32.0k └main cwd...</div><div>-- INSERT -->> accept edits on (shift+tab to cycle)</div></div>	

Shameless plug.

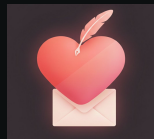
What you can deploy this way in a week of holidays!



sting9.org Symfony + Go + TanStack + Swift (iOS)



airportflow.app Symfony + Go + TanStack + Swift (iOS)



dearvalentine.app Symfony + Go + Swift (iOS)



instags.pics Symfony + Swift (iOS)



THE A-TEAM

AI AGENTS



Thank **you.**



Guillaume Moigneu

Field CTO @ Upsun

guillaume@upsun.com

guillaume.id