

# git-context YouTube Script

FastAPI Demo + Ethics + Composer . 4:30 minutes

---

| Section        | Time        | Duration |
|----------------|-------------|----------|
| Hook + Problem | 0:00 - 0:30 | 30s      |
| Transition     | 0:30 - 0:45 | 15s      |
| Live Demo      | 0:45 - 2:10 | 85s      |
| Ethics         | 2:10 - 2:40 | 30s      |
| Composer Demo  | 2:40 - 4:10 | 90s      |
| CTA            | 4:10 - 4:30 | 20s      |

---

## SECTION 1 HOOK + PROBLEM

0:00 to 0:30 | 30 seconds

*[ fastapi/fastapi on GitHub - 80k stars, file tree visible ]*

"FastAPI. 80,000 stars. The framework that convinced an entire generation of Python devs they understood async - until they opened the source code.

routing.py - 800 lines. There's a class called Dependant - yes, spelled without the E, intentionally, this is load-bearing confusion - with a field called path\_fields and enough nested generics to make your IDE give up and go home.

You need this whole thing inside an LLM. One shot. Let's go."

---

## SECTION 2 TRANSITION

0:30 to 0:45 | 15 seconds

*[ Address bar - github.com/fastapi/fastapi ]*

"You're already on the GitHub page. Don't copy anything. Click the address bar, swap github.com for git-context.com, hit Enter.

"Yes, it's that stupid simple. We're already in."

*[ Slow zoom - github.com becomes git-context.com - Enter ]*

github.com/fastapi/fastapi  
becomes

**git-context.com/fastapi/fastapi**

---

## SECTION 3 LIVE DEMO

0:45 to 2:10 | 85 seconds

*[ git-context loads - ETA card shows 39 MB, Python, estimated time ]*

"Before you've touched anything - repo size, primary language, processing estimate. It hit the GitHub API while you were blinking. 39 megabytes of pure async Python waiting to be understood."

*[ Select Claude Sonnet 4.6 - hit Generate ]*

"Claude Sonnet, 160k tokens. That's the entire FastAPI internals with room left over to complain about it. Generate."

*[ Clone and Scan completes - hover tooltip as you speak ]*

"Phase one - shallow clone, depth one. Not pulling a decade of commits from Tiangolo's git history. Just the latest snapshot, into a temp directory, gone when we're done."

*[ Filter and Classify completes - hover tooltip ]*

"Phase two - surgically removing everything that isn't code. Docs folder, test fixtures, mkdocs config, the GitHub Actions YAML that runs on every push and breaks on Tuesdays - all of it gone. Only the source that actually matters makes it through."

*[ Parse and Extract completes - hover tooltip ]*

"Phase three - tree-sitter AST extraction, running on your CPU right now. Class definitions, function signatures, decorators, import chains. This is how it knows @app.get is architecturally critical and # noqa is a cry for help, not a signal."

*[ Semantic Analysis completes - hover tooltip ]*

"Phase four - dependency centrality, architectural layers, cross-file call graphs. It has determined that routing.py is the gravitational centre of this codebase and everything else orbits it. In about two seconds. You're welcome."

*[ Assemble Context completes - output scrolls into view ]*

"Done. FastAPI - the routing engine, the DI system, the OpenAPI generator, the Pydantic bridge - one structured markdown file, token-budgeted to exactly what

Claude can hold. Paste it in. Ask how Depends() resolves at runtime. You will finally understand what Dependant without the E actually does."

---

## SECTION 4 ETHICS

2:10 to 2:40 | 30 seconds

*[ Hover the Processed Locally badge - then pan to ethics shields ]*

"Now - because you're about to paste proprietary source code into an AI and your CISO is watching this video - let's talk about what actually happened to your code."

*[ Hover Clone shield ]*

"Cloned to a temp dir on your machine. Nothing left the box."

*[ Hover Filter shield ]*

"Env files, secrets, credentials - stripped before a single byte is parsed."

*[ Hover Parse shield ]*

"tree-sitter ran locally. Your source code has not visited us-east-1."

*[ Hover Assemble shield - hold on Processed Locally badge ]*

"Assembled in your browser. Not cached, not logged, not sold to a data broker in a trench coat. That badge is a technical fact, not a privacy policy written by a lawyer."

---

## SECTION 5 COMPOSER DEMO

2:40 to 4:10 | 90 seconds

*[ Click Composer tab ]*

"One more thing. Composer. And I'm going to do something slightly unhinged - I'm pointing it at git-context itself. We are using the tool on its own source code."

*[ Add git-context repo URL - knowledge graph builds ]*

"It runs the same pipeline, then builds a knowledge graph across the whole codebase. It's already figured out that the orchestrator is the spine, the pipeline phases are the vertebrae, and the assembler is where everything either works or silently produces 12 tokens of garbage."

*[ Type prompt - show clearly on screen, read it out ]*

"I'll give it a real job:"

*"Take the git-context pipeline and give me a minimal standalone Python script - just clone, filter, and assemble. No web server, no tree-sitter, no UI. Just tiktoken, basic regex, markdown out. Single file. I want to run it with one command."*

*[ Composer generating ]*

"Here's what's different from asking ChatGPT the same thing. ChatGPT will hallucinate a function called `build_context()` that doesn't exist. Composer read the actual source - it knows `clone_and_scan` returns a `CloneScanResult`, it knows `filter_and_classify` takes a list of `FileInfo` objects, it knows the token counter lives in `utils/tokens.py`. The output compiles because the input was the real codebase."

*[ Output code appears - scroll slowly ]*

"Single file. python script.py and a GitHub URL and you get a context doc out the other end. No server, no config, no existential dread."

That's the difference. ChatGPT guesses. Composer knows."

---

## SECTION 6 CTA

4:10 to 4:30 | 20 seconds

*[ git-context GitHub repo ]*

"Fully open source. Link in the description. Go star it - and go finally understand whatever codebase has been silently judging you from your bookmarks bar.

"See you in the next one."



---

## PRODUCTION NOTES

- URL edit - zoom in tight, slow keystroke by keystroke. The audience needs to read it. This is the hook.
- Dependant without the E - if you've touched FastAPI source you'll laugh. Pause for it.
- 160k tokens line - slight smirk when you say 'room left over to complain about it'.
- # noqa is a cry for help - pause after it. That's a laugh line.
- us-east-1 - tech crowd gets it immediately. Pause for the reaction.
- Composer knowledge graph - 2-3 seconds of silence before speaking. Visual first.
- The 12 tokens of garbage line - delivery matters. Say it like it's happened to you personally.
- Final line: full stop before 'ChatGPT guesses. Composer knows.' No filler. Cut straight to CTA.
- Thumbnail: @app.get("/understand-this") + "I finally get how FastAPI works".