How to do Code Review like a Pro

Treat review as the fastest learning loop.

Jacek: hands-on engineer & architect, code reviews daily, Gerrit maintainer

OpenInfra SUMMIT › EUROPE'25

Why reviews matter

- Build shared understanding, not just gate code
- Mentor across experience levels, daily
- Align practices, patterns, invariants
- Spread insights across the codebase
- Quality improves **via knowledge transfer**

Bacchelli & Bird (ICSE'13); Wen–Lamothe–McIntosh (ICSE-SEIP'22)

Tangible outcomes

Readable diffs. Reliable changes. Repeatable decisions

- **Review turnaround** ≤ 1 business day
- **Median review size** ≤ ??? changed lines
- **Tests updated:** CI green on merge
- **Traceability:** link issue / ADR / RFC
- **Post-release defects** trending down

"Higher participation & coverage → fewer post-release defects." McIntosh et al., MSR 2014; EMSE 2016

Author mindset

"If it's hard to review, it's hard to maintain."

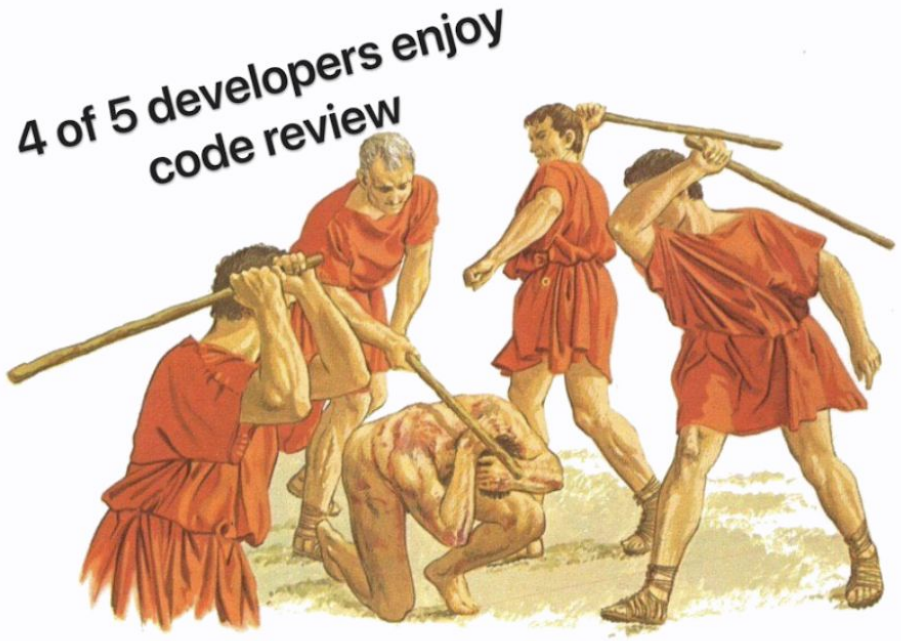
Your job: help reviewers help you.

Author techniques

- Small, single-concern reviews; separate refactors.
- Write the **what** first (template what / why / extra context).
- **Conventional Commits:** feat / fix / chore / ... for history and context.
- Self-review checklist before request.
- Let tools catch nits (fmt, lint, tests).
- Use agents (copilot, etc.) for scaffolding - you verify.

https://www.conventionalcommits.org/

Reviewer mindset shift from "find flaws"



r/ProgrammerHumor

Reviewer mindset shift to "coach craft"

Label every comment: blocking / non-blocking / nit

Principles over opinions.

Reviewer Techniques: Conventional Comments

For clear, actionable feedback.

```
<TAG> (decorations): <brief summary>
Context: why it matters (principle/invariant)
Suggestion: concrete change (or diff)
Outcome: what improves if applied
```

Tags: **issue, suggestion, question, praise, nit, thought, ...**

Decorations: **blocking, non-blocking, if-minor, ...**

https://conventionalcomments.org/

Bad → Good (comment example)

Bad: *“Just use a map here.”*

Good (Conventional Comments):

```
issue (blocking): Replace linear search with HashMap
Context: List lookup is O(n) - bigger n slower lookup. Map preserves O(1).
Suggestion: userMap.get(id) instead of users.find()
Outcome: Maintains performance under load
```

Coach with principles; show the outcome.

Team practices & tooling

- Review templates; CODEOWNERS/rotation.
- SLAs: first response ≤1 business day.
- Link ADR/RFC when principles apply.
- Tooling: IDE, formatters, linters, tests, review with agent.
- **Conventional Commits** for clean history.
- **Conventional Comments** for replies.

Make great reviews the default.

Anti-patterns & fixes

Anti-pattern	Fix
Mega-PRs	Split by feature or commits
Drive-by nit-picking	Use "issue(blocking)"
Rubber-stamps	"What I checked" comment
Bikeshedding	Cite guideline, short RFC, follow-up

Block on principles, not preferences.

Summary & call-to-action

Focus the diff. Coach with principles. Let tools police nits.

By next Friday:

- Use review template & self-review checklist.
- Adopt **Conventional Comments** + tags and severity labels.
- Pilot **Conventional Commits** to improve history.

Thanks! https://geminicaprograms.github.io/code_review_like_a_pro/events-openinfra-2025/