GHC Contribution and Collaboration Policy

1. Purpose

GHC is developed and maintained by a community of people, many of whom contribute their time voluntarily.

We want to:

1. nurture the motivation, relationships, knowledge, and enjoyment of the people who maintain and develop GHC;
2. keep GHC software understood, directed, and maintained by a community of human beings;
3. build a codebase that we can be proud of: correct, well structured, well documented, maintainable, and, where possible, elegant;
4. welcome contributors with different backgrounds, abilities, working methods, and preferences;
5. use the limited time of contributors and reviewers responsibly.

This policy describes the **outcomes, responsibilities, and forms of collaboration** expected from contributors. Except where a workflow directly affects other people or creates a legal or technical obligation for the project, it does not prescribe which tools contributors may or may not use.

2. Equal standing and freedom of method

Contributors are free to choose the tools and working methods that allow them to make their best contribution.

These may include editors, compilers, formatters, refactoring tools, code generators, search engines, documentation systems, language-assistance tools, autocomplete systems, LLMs, other automated tools, and assistance from other people.

The GHC project neither requires nor discourages the use of any particular class of assistive tool merely because of how that tool works.

In particular:

- contributors who do not use LLMs are fully welcome and should not be placed under pressure to use them;
- contributors who responsibly use LLMs or comparable assistive technologies are equally welcome;
- a contribution is not considered better merely because every part of it was manually composed;

- a contribution is not considered worse merely because assistive tools participated in its production;
- no contributor should be described or treated as more authentic, more skilled, more valuable, or more desirable solely because of their choice of tools.

The project does **not** express a general preference for “human-written” over assisted code or documentation.

Human authorship is established through meaningful human direction, judgment, adoption, understanding, and responsibility. It does not require a human to have manually originated every token in the final artifact.

3. A human owner for every contribution

Every contribution must have an identifiable human owner.

The owner is responsible for the contribution regardless of how it was produced. They must be able to:

- explain the problem the contribution addresses;
- motivate the proposed solution;
- explain the important design choices and trade-offs;
- understand the behavior of the submitted change sufficiently to review, defend, modify, and maintain it;
- respond meaningfully to questions and criticism;
- investigate uncertainties discovered during review;
- correct defects;
- confirm that relevant tests and claims have been checked;
- take responsibility for licensing, attribution, and the right to submit the material.

Full responsibility does not mean that a contributor must already know everything about GHC. GHC is a large and difficult codebase, and asking questions is expected. A contributor may say that they do not know something and seek help.

What is not acceptable is transferring basic understanding or validation to reviewers: submitting material that the contributor has not seriously examined and expecting reviewers to determine what it does, whether it is correct, or why it should be merged.

The use of an assistive tool does not reduce the contributor’s responsibility. Equally, manually writing a contribution does not demonstrate that the contributor understands it.

4. Human collaboration

Contributing to GHC means entering a community of people.

Contributors and reviewers should communicate respectfully and in good faith. Reviewers are offering their time and experience; contributors are offering their work. Both should recognize the value of the other's participation.

Communication may be assisted by tools. A contributor may use tools to:

- improve or translate their English;
- organize their thoughts;
- understand technical material;
- summarize information;
- explore possible answers;
- check grammar or clarity;
- prepare a draft response.

Tool assistance does not make a conversation inauthentic. What matters is that the person sending a message has read it, understands it, accepts it as an expression of their own position, and is prepared to discuss and defend it.

A contributor must not outsource the conversation itself. In particular, they should not take a thoughtful review comment, feed it into a tool, and return its output without forming their own judgment about the issue.

Where raw machine output is included for examination rather than adopted as the contributor's own statement, it should be clearly identified as a quotation or generated output.

The same expectation applies to reviewers. Review tools may assist a review, but the human reviewer remains responsible for the comments they submit and for the judgment those comments express.

5. Contributions ready for review

A merge request marked ready for review should respect the limited time of reviewers.

It should ordinarily:

- explain the problem and intended result;
- describe the approach taken;
- be focused and reasonably scoped;
- avoid unrelated changes;
- identify important design decisions or alternatives;
- contain appropriate tests;
- contain appropriate documentation;
- identify known limitations, uncertainties, or risks;
- have been read and checked by its contributor;
- meet the standards the contributor would reasonably expect from an MR they had agreed to review.

The relevant question is not how quickly or by what means the change was produced. The question is whether it has been brought to a state in which another human can review it without first having to reconstruct its purpose, remove obvious mistakes, or discover that its author cannot explain it.

Large, incoherent, repetitive, or poorly understood submissions may be declined regardless of whether they were produced manually, generated automatically, copied from elsewhere, or assembled through some combination of methods.

6. Draft merge requests

Draft MRs are not held to the same standard as review-ready MRs.

A contributor may open a draft MR to:

- run continuous integration;
- share an early design;
- preserve work in progress;
- request limited feedback;
- investigate an implementation approach.

A draft should be clearly marked as not ready for review. Reviewers should not assume that a draft is asking for a complete code review unless the contributor explicitly requests one.

Before changing an MR from draft to ready, the contributor should bring it into compliance with the review-readiness expectations above.

7. Review is voluntary and not guaranteed

Submitting a contribution does not create an entitlement to a review.

GHC has a limited pool of reviewers, and not every contribution will automatically attract someone with the time, interest, and relevant expertise to review it.

Contributors are encouraged to participate in the community and seek a reviewer or shepherd through the mailing list, Matrix, IRC, or other appropriate channels.

Individual reviewers remain free to decide which work they undertake. A reviewer declining to review an MR is not, by itself, a project-level rejection of the contribution or judgment of the contributor.

An MR that has not found a reviewer or shepherd within a reasonable period—for example, three months—may be closed without prejudice. It may be reopened or resubmitted when there is sufficient interest or review capacity.

Contributors are also encouraged to become reviewers themselves. A sustainable community cannot consist only of people asking others to review their work.

8. Attribution and provenance

This applies consistently to:

- code copied or adapted from another project;
- substantial text taken from an external source;
- work produced by another person;
- contracted or commissioned work;
- substantial machine-generated code, tests, documentation, or prose;
- other externally produced material incorporated into a contribution.

Attribution is not a quality rating. It does not imply that the contributor is less capable, that the work is presumptively defective, or that the contribution occupies a lower category.

For assistive or generative tools, contributors should use reasonable judgment. A brief disclosure is sufficient. The project does not need an exhaustive record of every completion, query, prompt, spelling correction, private brainstorming session, or code-navigation interaction.

Routine uses such as autocomplete, grammar correction, code search, private analysis, or small local transformations do not need to be individually catalogued.

When substantial generated material forms part of what others are being asked to examine, a contributor may include a simple statement such as:

> *Assistive generative tools were used in preparing parts of this contribution. I have reviewed the submitted material and take responsibility for it.*

A contributor who routinely uses such tools may use a standing disclosure rather than attempting to identify every individual interaction.

No prompt transcript, model name, generated percentage, or detailed history is required unless it is materially relevant to a specific licensing, security, or technical issue.

9. LLMs and similar tools

The project does not make a general claim that LLM-generated work is better or worse than human-composed work. Different workflows may produce different kinds of mistakes, but contributions should be evaluated on the mistakes and qualities actually present rather than on assumptions about their provenance.

The same contribution standards apply regardless of LLM involvement.

Acceptable uses may include:

- codebase navigation;
- explanation and analysis;
- brainstorming;
- design exploration;
- translation and language assistance;
- autocomplete;
- code generation;
- refactoring;
- test generation;
- documentation drafting;
- debugging;
- review assistance.

The fact that one of these uses occurred does not excuse the contributor from understanding and owning the result.

The following are unacceptable, not because they involve an LLM, but because they violate the general contribution standards:

- submitting a large body of output without seriously examining it;
- submitting code the contributor cannot explain;
- making technical claims the contributor has not checked;
- producing oversized or incoherent MRs that transfer the cost of generation to reviewers;
- repeatedly submitting low-quality drive-by contributions;
- failing to engage with review;
- presenting generated replies as a substitute for human judgment;
- misrepresenting the source or ownership of material.

An assisted contribution should not be rejected by the project solely because it is assisted. Nor should a manually composed contribution receive a presumption of correctness merely because it is manually composed.

Disclosure may help an individual reviewer decide whether they wish to participate. It must not be treated by the project as a warning label, a mark of diminished contributor status, or evidence that stricter acceptance criteria should automatically apply.

The project should not create separate “preferred,” “pure,” “traditional,” or “human-written” classes of contributors.

10. Review and acceptance criteria

Review should focus on observable properties of the contribution, including:

- correctness;
- relevance to GHC;
- technical design;
- simplicity;
- maintainability;
- performance;
- testing;
- documentation;
- code quality;
- compatibility;
- licensing;
- security;
- reviewability;
- the contributor's responsiveness and willingness to improve the work;
- the project's available maintenance and review capacity.

Production method may be relevant where it creates a concrete issue—for example, uncertain licensing or unclear provenance. It should not be used as a general proxy for quality, competence, trustworthiness, or community standing.

A reviewer should describe actual deficiencies:

- "This code duplicates an existing abstraction."
- "This MR is too large to review."
- "The author cannot explain this behavior."
- "The tests do not establish the claimed property."
- "The provenance of this code creates a licensing question."

A reviewer should not substitute an assumption about tool use for a substantive review finding.

11. Good faith

The GHC community should begin from an assumption of good faith.

Contributors are not required to prove that they did not use a particular tool. Style, wording, mistakes, or intuition alone are not sufficient grounds for accusing someone of concealing assistance.

Where clarification is useful, it should be requested directly and without accusation.

Good-faith uncertainty about whether a particular use merits disclosure should normally be resolved through clarification, not punishment.

Deliberate deception, plagiarism, or misrepresentation supported by concrete evidence may be treated as a conduct and trust issue. That is different from requiring contributors to prove an unverifiable negative.

Unenforceable rules about private working methods should be avoided. Such rules are likely to deter conscientious contributors while doing little to prevent people willing to ignore or evade them.

12. Repeated poor-quality or abusive contributions

The project may decline, close, or ignore contributions that are:

- persistently poor in quality;
- clearly unreviewable;
- submitted without meaningful ownership;
- repeatedly abandoned during review;
- abusive of reviewer time;
- deceptive;
- disruptive to the community.

Where practical, contributors should first receive clear feedback and an opportunity to improve.

A repeated pattern of ignoring project expectations may result in contribution restrictions or, in serious cases, exclusion from project infrastructure.

13. Evidence-based revision

This policy should be revised in response to concrete project experience.

If a particular workflow begins to create a substantial burden—for example, a large increase in unreviewable submissions—the project should address the burden directly through measures such as:

- clearer review-readiness requirements;
- scope limits;
- better templates;
- automated testing;
- improved contribution guidance;
- closing abandoned MRs;
- contribution restrictions for repeated low-quality behavior.

Restrictions should be no broader than the demonstrated problem requires and should apply to equivalent behavior regardless of the tool that produced it.

Broader ethical, environmental, economic, or political positions about LLMs may be legitimate subjects for community discussion. But if GHC intends to adopt such a position, it should do so openly and explicitly. It should not encode a broader ideological judgment indirectly through claims about code quality or “human authorship.”