PR #2720 完整报告

radixark/miles

fix(ci): grant pull-requests write to comment-gateway feedback jobs

合并时间：2026-08-24 03:27

原文链接：http://prhub.com.cn/radixark/miles/pull/2720

执行摘要

- 一句话：/rerun-test 反馈升级：同一条评论从 running 原地更新到终态
- 推荐动作：值得精读。该 PR 虽然只涉及 CI 基础设施，但其设计决策具有可迁移性：状态评论的 ownership 归属（run 自持而不是 gateway 代发）、comment ID 跨 job 传递、always() finalizer 汇总多 job 结果、以及结果映射的 fail-closed 语义，都是构建可靠的 PR 评论机器人的良好范本。尤其推荐关注 _file_run_status_inputs 的严格入参校验与 _file_run_outcome 的终态映射，这两处体现了"宁可报失败也不静默"的工程原则。

功能与动机

PR body 明确指出旧反馈机制不可信：评论链接的是几秒钟的权限探测 run 而不是真正分派的测试，没有 in-progress 状态、没有耗时、也不会随测试结果更新。实际运行证据是旧评论链接的 run 32637432938，其 jobs 只测试了反馈权限，而真正的测试结果在另一个 run 中，reviewer 无法从评论直接判断成败。此外 commit de5031c0 记录了权限根因：acknowledge 与 reply 两个 job 调用的 issues API 目标评论永远挂在 PR 上，GitHub 对这类调用按 pull-requests scope 门控，仅有 issues: write 时 run 32630924734 中 dispatch 成功但两个反馈 job 均返回 403。

实现拆解

1. 权限修复先行：.github/workflows/comment-ci-command.yml 中删除 reply-command job（连带移除 workflow_run_url output），并为 acknowledge job 增加 pull-requests: write。原因是 GitHub 对目标 issue 为 PR 的 issues-API 调用按 pull-requests scope 门控，单独 issues: write 会 403。这是 commit de5031c0 的独立修复，被后续重构吸收。
2. 状态评论架构取代静态回复：.github/workflows/run-ci-file.yml 新增两个 job。announce-file-run 在 run 启动时发布 running 状态评论（含测试文件、run 链接、开始时间、<!-- rerun-test-run:<id> --> 标记），并把 comment_id 作为 job output 传给下游；report-file-run 以 if: always() 挂载在 resolve、CUDA 执行、CPU 执行三个 job 之后，汇总 needs.*.result 映射为终态，读取 run_started_at 计算耗时后 PATCH 回同一个评论 ID。
3. handler 能力扩展：.github/workflows/scripts/comment_ci_command.py 中 create_issue_comment 改为返回新建评论的 ID，新增 update_issue_comment（PATCH）与 get_workflow_run（读 start 时间）；删除静态 reply_event 路径。新增 _file_run_status_inputs 统一解析 announce / report 两种模式的环境变量并严格校验（测试文件白名单、suite 格式、job result 枚举）；_file_run_outcome 定义 fail-closed 的终

态映射——只有执行 job 真正 success 才报 passed，resolver 失败、无执行 job、取消均显式报 failed / cancelled。

4. 测试与文档配套：tests/ci/test/test_comment_ci_command.py 新增 FakeAPI 的 update_issue_comment / get_workflow_run，并新增 announce、report、终态映射参数化测试（含非法输入 fail closed）；tests/ci/test/test_file_run.py 中的 workflow 不变式测试更新为断言 announce-file-run / report-file-run 的结构（checkout 次数从 2 变为 4、issues: write 与 pull-requests: write 各出现 2 次、report job 含 actions: read 且 if: always()）。文档 docs/ci/01-label.md 与 docs/ci/05-command-identity.md 同步重写反馈流程与 token 权限说明。

关键文件：

- .github/workflows/scripts/comment_ci_command.py（模块 命令网关；类别 infra；类型 infrastructure；符号 announce_file_run, report_file_run, _file_run_status_inputs, _file_run_outcome）：CI 评论命令处理器的核心脚本，本次重写了 /rerun-test 的反馈路径：删除静态 reply_event，新增 announce / report 双阶段状态评论能力、严格入参校验与耗时计算。
- tests/ci/test/test_comment_ci_command.py（模块 命令网关；类别 test；类型 test-coverage；符号 file_run_env, file_run_status, update_issue_comment, get_workflow_run）：覆盖最重的测试文件（+251/-47），新增 FakeAPI 的 update_issue_comment / get_workflow_run 能力，并补齐 announce、report、终态映射与非法输入 fail closed 的完整参数化测试。
- .github/workflows/run-ci-file.yml（模块 工作流编排；类别 infra；类型 infrastructure）：被分派的文件测试工作流，本次新增 announce-file-run 与 report-file-run 两个 job，构成了状态评论的完整生命周期：创建 running 评论，最后由 always() finalizer 汇总三个上游 job 结果并原地更新。
- .github/workflows/comment-ci-command.yml（模块 工作流编排；类别 infra；类型 infrastructure）：评论网关工作流。删除静态 reply-command job，并为 acknowledge job 增加 pull-requests: write 权限，修复 PR 评论上 issues API 调用 403 的根因。
- tests/ci/test/test_file_run.py（模块 文件运行；类别 test；类型 test-coverage）：workflow 不变式测试：把 checkout 次数断言从 2 更新为 4，新增 announce / report job 的结构断言（权限块、comment_id 输出、if: always()），防止工作流编排回归。
- docs/ci/01-label.md（模块 CI 文档；类别 docs；类型 documentation）：用户侧文档，重写 /rerun-test 的反馈描述：从静态 reply 链接改为 running 评论 + 原地更新终态，并明确 passed 只在执行 job 成功时报告。
- docs/ci/05-command-identity.md（模块 CI 文档；类别 docs；类型 documentation）：token 身份文档，记录 GitHub 对 PR 上的 issues-API 调用按 pull-requests scope 门控的规则，以及各 job 的权限矩阵。

关键符号：announce_file_run, report_file_run, _file_run_status_inputs, _file_run_outcome, _file_run_marker, _format_duration, _parse_run_timestamp, _run_elapsed, update_issue_comment, get_workflow_run, create_issue_comment, FileRunStatus, reply_event

关键源码片段

.github/workflows/scripts/comment_ci_command.py

CI 评论命令处理器的核心脚本，本次重写了 /rerun-test 的反馈路径：删除静态 reply_event，新增 announce / report 双阶段状态评论能力、严格入参校验与耗时计算。

```
def _file_run_status_inputs(environ):
    """从环境变量解析 announce / report 两种模式的入参。

    所有字段都做严格校验：测试文件路径必须匹配注册白名单，suite 必须是
    纯小写短横线名，job result 只能落在 {success, failure, cancelled,
    skipped, ""} 之内。任何异常输入都直接抛错，让 workflow job 失败，
    绝不把脏数据写进 PR 评论。
    """
    mode = environ.get("CI_COMMAND_FILE_RUN_STATUS", "")
    if mode not in FILE_RUN_STATUS_MODES:
        raise CommentCommandError("file run status mode must be announce or report")
    pull_number = _parse_positive_int(environ.get("FILE_RUN_PULL_NUMBER"), "pull request
    number")
    run_id = _parse_positive_int(environ.get("FILE_RUN_RUN_ID"), "workflow run ID")
    test_file = environ.get("FILE_RUN_TEST_FILE", "")
    if TEST_FILE_PATTERN.fullmatch(test_file) is None:
        raise CommentCommandError("file run test file is invalid")
    suite = environ.get("FILE_RUN_SUITE", "")
    if suite and SUITE_PATTERN.fullmatch(suite) is None:
        raise CommentCommandError("file run suite is invalid")
    results = {
        "resolve": environ.get("FILE_RUN_RESOLVE_RESULT", ""),
        "cuda": environ.get("FILE_RUN_CUDA_RESULT", ""),
        "cpu": environ.get("FILE_RUN_CPU_RESULT", ""),
    }
    for name, value in results.items():
        if value not in FILE_RUN_RESULTS:
            raise CommentCommandError(f"file run {name} result is invalid")
    # CUDA / CPU 两个执行 job 由各自的 if 互斥，未执行的一方为 skipped，
    # 所以真正的执行结果就是那个非 skipped 的值。
    executed = [value for value in (results["cuda"], results["cpu"]) if value not in {"skipped", ""}]
    if len(executed) > 1:
        raise CommentCommandError("file run reported two executing jobs")
    comment_id = None
    if mode == "report":
        comment_id = _parse_positive_int(environ.get("FILE_RUN_COMMENT_ID"), "comment ID")
    return (
        mode,
        FileRunStatus(
            pull_number=pull_number,
            test_file=test_file,
            run_id=run_id,
            suite=suite,
```

```
            resolve_result=results["resolve"],
            execute_result=executed[0] if executed else "",
        ),
        comment_id,
    )
```

评论区精华

该 PR 没有实质性的 human review 交锋：review 列表中只有 claude[bot] 的自动配置提示，评审意见为空的。设计讨论主要沉淀在 PR body 和 commit 消息中，值得提炼的点：

- 状态评论的所有权归属：PR body 的 Design Notes 明确采用“run-owned comment”心智模型——默认分支 gateway 只负责授权与分派，状态评论由被分派的 run 自己创建并持有，comment ID 通过 needs 输出流经 resolver / execution 进入 always() finalizer，避免 gateway 与 run 双方抢写同一条评论。
- 权限 scope 的边界推敲：commit de5031c0 描述中提到，reaction 与状态评论都是 PR-targeted 调用，GitHub 用 pull-requests scope 门控，因此 issues: write 单独不够；finalizer 额外需要 actions: read 才能读取 run_started_at。每个 job 的权限都按最小集精确声明。
- fail-closed 的结果语义：PR body 的 Review Focus 强调“只有 execution success 才报 passed；无效 status 输入 fail closed”。实现上 _file_run_outcome 把 resolver 失败、无执行 job、取消都显式映射为失败或取消，绝不把“没跑”的 run 粉饰成成功。
- 反馈 job 403 根因：issues: write 与 pull-requests: write 的 scope 门控 (correctness): 给 acknowledge 与 file-run 状态 job 均增加 pull-requests: write，并在 05-command-identity.md 记录该 scope 规则。
- 状态评论 ownership：gateway 静态回复 vs workflow run 自持 (design): 移除 gateway 的 reply-command job，采用 run-owned 状态评论，保证“谁执行谁报告”。
- 终态映射的 fail-closed 语义 (correctness): 实现上 _file_run_outcome 把 resolver 失败、无执行 job、取消均显式映射为 failed / cancelled，任何未知状态都会抛错而不是静默。
- 临时验证工作流与合入策略 (other): 验证完成后删除临时工作流，第 7 个 commit 合入正式实现。

风险与影响

- 风险：
  - comment ID 传递链断裂风险：report-file-run 依赖 announce-file-run.outputs.comment_id。若 announce 阶段评论创建失败或 output 缺失，report 阶段会因 FILE_RUN_COMMENT_ID 解析失败而 fail closed，PR 上不会出现任何状态评论——行为是安全的（不伪装成功），但会造成反馈缺失。
  - if: always() 单点失败：report job 自身失败时没有重试或兜底，可能出现“run 已结束但评论仍显示 running”的窗口。当前实现把这一点视为可接受，因为反馈 job 只跑固定默认分支代码，失败概率低。
  - 权限扩散：.github/workflows/comment-ci-command.yml 与 run-ci-file.yml 中新增的 pull-requests: write 作用域是 workflow 的 GITHUB_TOKEN，仅限当前仓库、不能跨

仓库，且这些 job 只执行默认分支固定代码；风险主要在未来工作流改动时权限块被误复制到执行 PR 代码的 job。

- 测试脆弱性：test_file_run.py 使用工作流文本断言（如 checkout 次数、issues: write 出现次数），对工作流格式调整敏感，后续修改 run-ci-file.yml 时需同步更新字符串断言，否则会脆断。
- 并发边界：queue: max + cancel-in-progress: false 下多个 /rerun-test 命令可并行，各自的 github.run_id 与 marker 一一对应，状态评论互不覆盖；但若有人手动编辑评论破坏 marker，跨 run 溯源会失效（低概率）。

- 影响：
  - 对开发者：以后跑 /rerun-test 只需盯住一条评论，从 running 到最终结果（含 suite 与 3m10s 级别的耗时）原地更新，不再需要去 Actions 页面人工核对 run 是否就是自己那条命令触发的。
  - 对 CI 系统：gateway 职责收窄为“授权 + 分派 + 点赞”，反馈责任下沉到 workflow run 自身，符合“谁执行谁报告”的 ownership 原则；移除静态 reply-command 后全流程只有一条状态评论，减少噪音。
  - 对团队：修复了反馈 job 的 403 导致的“评论链接的 run 根本没跑测试”的信任危机；文档同步更新了 token 身份矩阵，后续维护者能清楚看到每个 job 的权限边界。改动不触及 miles/ 训练、推理或 dashboard 代码，对产品运行时零影响。
  - 风险标记：新增 pull-requests 写权限，always() 终态链路依赖 comment ID，报告 job 自身失败无兜底，测试依赖工作流文本断言

关联脉络

- PR #2676 feat(ci): acknowledge rerun-test commands: 直接前身：为 /rerun-test 引入 +1 确认与 Actions 运行链接回复。本次 PR 把 2676 的静态 reply 机制升级为 run-owned 状态评论闭环，并修复其反馈 job 的 403 权限问题。
- PR #2496 feat(ci): add authorized comment-to-label gateway: 评论命令网关的根基实现，本次改动沿用其授权模型与 comment-command-access.json 策略，并收敛了 gateway 的反馈职责。
- PR #2675 [CI] rebuild a PR's docker image only when its build inputs change: 同一批 CI 基础设施演进，同期在打磨 PR 触发的 CI 体验；本 PR 与其共同构成对评论驱动 CI 工作流的可信度补强。