

PR #3251 完整报告

radixark/miles

ci(lark): rerun failed nightly jobs once and report flaky vs failed

合并时间: 2026-09-15 07:23

原文链接: <http://prhub.com.cn/radixark/miles/pull/3251>

执行摘要

- 一句话: nightly 失败 job 自动重跑一次, 卡片区分 flaky 与真实失败
- 推荐动作: 值得精读。这个 PR 是 CI 通知链路上的一次小而具有代表性的可靠性改进: 自动重跑 + flaky 区分的状态机设计 (attempt 1 重跑、attempt 2 报告、拒跑兜底、防循环守卫) 清晰且可测试。特别值得关注两点: 一是作者对未验证路径的诚实标注与 fallback 设计 (先合入、后观察、再切换 token 来源), 二是 identity boundaries 测试随权限提升同步收紧, 保证 actions: write 是唯一新增写权限。建议后续跟踪第一个失败的 nightly, 确认 workflow_run 事件确实触发, 否则需要切换到 App token。

功能与动机

nightly 只运行一次, 任何失败 job 都会发红卡, flaky 失败与真实失败看起来完全一样; 随后需要人工手动重跑失败 job, 而这会发布第二张卡, 第一张卡从不与结果 reconcile。PR body 给出了具体案例 run 34614419736: attempt-1 卡片出现在 Sep 11 17:20Z, attempt-2 卡片出现在 Sep 12 04:12Z。目标是让 flaky 与真实失败在卡片上区分开, 并消除人工重跑与卡片不一致的流程。

实现拆解

1. 重跑决策入口: `.github/workflows/scripts/lark_notify.py` 的 `cmd_ci_status` 在拿到 completed run 与 job 列表后, 先计算 `failed = failed_job_names(jobs)` 与 `attempt = run.get("run_attempt", 1)`。只有同时满足 `attempt == 1`、事件为 `schedule`、结论不是 `cancelled`、存在失败 job 四个条件才发起重跑, 保证重跑不会循环 (`attempt2` 永远不触发), 手动 dispatch 与 PR 事件不受影响。`--dry-run` 模式只打印 `would rerun N failed jobs` 并提前返回。
2. 新增 GitHub API 方法: `GitHub.rerun_failed_jobs(run_id)` 以 POST 请求调用 `/repos/{repo}/actions/runs/{id}/rerun-failed-jobs`, 要求返回 201; `HTTPError` / `URLError` 或非 201 都包装为 `RuntimeError`, 错误正文截断 300 字节。成功打印 `card follows attempt 2` 后直接返回, 不分析失败、不发布卡片。
3. 第二次尝试的卡片渲染: `attempt 2` 完成再次触发 `cmd_ci_status` 时, 用 `gh.run_attempt_jobs(run_id, attempt - 1)` 取 `attempt 1` 的失败集合, `diff_attempts` 按 job 名分成 `fixed` / `still` / `new`。`render_ci_status` 把原 `Fixed by rerun` 标题改为 `Flaky, passed on rerun`, 标题与 job summary 追加 `N flaky`, 并删除 `Rerun #N` - 前缀 (因为现在每次失败的 nightly 都是 rerun)。

- 拒跑兜底：若 POST 失败，`rerun_error` 被记录，仍然走完 attempt 1 的卡片发布（含 AI 分析），最后 `raise rerun_error` 让 notifier job 失败，使缺失的重跑对外可见。
- 权限与测试配套：`.github/workflows/ci-lark-notify.yml` 把 `actions: read` 改为 `actions: write`，仅用于 rerun-failed-jobs；AI 分析仍用只读 App token。
`tests/ci/test/test_lark_notify.py` 新增 13 个用例，覆盖重跑触发、防循环、cancelled 与 manual 豁免、dry-run、拒跑兜底、HTTP 请求格式，并把 identity boundaries 断言从“禁止 actions: write”改为“允许 actions: write 但禁止 contents / pull-requests 写权限”。

关键文件：

- `.github/workflows/scripts/lark_notify.py`（模块 CI 通知；类别 infra；类型 infrastructure；符号 `rerun_failed_jobs`, `cmd_ci_status`, `render_ci_status`）：核心逻辑所在：新增 `rerun_failed_jobs` 方法、在 `cmd_ci_status` 中加入自动重跑决策与拒跑兜底、在 `render_ci_status` 中把 rerun 修复的 job 渲染为 flaky 并去掉 Rerun #N 前缀。
- `tests/ci/test/test_lark_notify.py`（模块 CI 通知；类别 test；类型 test-coverage；符号 `test_jobs_fixed_by_the_rerun_count_as_flaky_not_failed`, `test_a_rerun_that_fixes_every_failure_passes_with_a_flaky_count`, `test_first_failed_nightly_attempt_reruns_its_failed_jobs_instead_of_posting`, `test_a_rerun_attempt_is_reported_and_never_rerun_again`）：行为契约的全部 13 个新用例都集中在这里：重跑触发、防循环、cancelled / manual 豁免、dry-run、拒跑兜底、HTTP 请求格式，以及 flaky 计数的标题断言，是理解新状态机的唯一完整入口。
- `.github/workflows/ci-lark-notify.yml`（模块 CI 通知；类别 infra；类型 configuration）：权限配置变更点：actions 从 read 提升为 write 以允许 rerun-failed-jobs，同时文件头注释说明自动重跑语义，是权限模型收紧与回退判断的依据。

关键符号：GitHub.`rerun_failed_jobs`, `cmd_ci_status`, `render_ci_status`, `diff_attempts`, `failed_job_names`

关键源码片段

`.github/workflows/scripts/lark_notify.py`

核心逻辑所在：新增 `rerun_failed_jobs` 方法、在 `cmd_ci_status` 中加入自动重跑决策与拒跑兜底、在 `render_ci_status` 中把 rerun 修复的 job 渲染为 flaky 并去掉 Rerun #N 前缀。

```
# GitHub API 客户端方法：请求重跑指定 run 的全部失败 job
def rerun_failed_jobs(self, run_id: int) -> None:
    # 只有返回 201 才认为重跑被接受，其余一律视为失败
    url = f"{GITHUB_API}/repos/{self.repo}/actions/runs/{run_id}/rerun-failed-jobs"
    req = urllib.request.Request(url, data=b"", method="POST")
    req.add_header("Authorization", f"Bearer {self.token}")
    req.add_header("Accept", "application/vnd.github+json")
    req.add_header("X-GitHub-API-Version", "2022-11-28")
    try:
        with urllib.request.urlopen(req, timeout=self.timeout) as resp:
            status = resp.status
    except urllib.error.HTTPError as e:
        # 错误详情截断到 300 字节，避免响应体污染日志
```

```

        body = e.read().decode("utf-8", errors="replace")
        raise RuntimeError(f"POST {url} -> {e.code}: {body[:300]}") from e
except urllib.error.URLError as e:
    raise RuntimeError(f"POST {url} failed: {e}") from e
if status != 201:
    raise RuntimeError(f"POST {url} -> {status}")

# cmd_ci_status 中的自动重跑决策核心分支
failed = failed_job_names(jobs)
attempt = run.get("run_attempt", 1)
rerun_error = None
# 仅首次失败的 schedule run 自动重跑: attempt 1 防止循环, cancelled 不重跑
if attempt == 1 and failed and run["event"] == "schedule" and run.get("conclusion") !=
"cancelled":
    if args.dry_run:
        # dry-run 只预告行为, 不真正发起重跑
        print(f"dry-run: would rerun {plural(len(failed), 'failed job')} of run {run['id']}")
        return
    try:
        gh.rerun_failed_jobs(run["id"])
    except RuntimeError as exc:
        # 拒跑不能牺牲报告: 仍发 attempt 1 卡片, 最后再把错误抛出让 notifier 可见
        rerun_error = exc
    else:
        # 重跑成功后不分析、不发卡, 等 attempt 2 的 completed 事件再来打报告
        print(f"rerun requested for {plural(len(failed), 'failed job')} of run {run['id']}; card follows
attempt 2")
        return

# render_ci_status 中的 flaky 计数与标题生成
# 第二次尝试才与 attempt 1 diff: 被重跑修复的 job 记为 flaky
diff = diff_attempts(failed, prev_failed) if prev_failed is not None else None
flaky = diff["fixed"] if diff else []
flaky_note = f", {len(flaky)} flaky" if flaky else ""

if conclusion == "cancelled":
    title = f"{name}: CANCELLED"
    color = "grey"
else:
    # failed 只统计 attempt 2 仍在失败的 job, 重跑修复的不算失败
    if failed:
        title = f"{name}: FAILED ({len(failed)} of {plural(len(counted), 'job')}{flaky_note})"
        color = "red"
    else:
        # 全部通过但曾有 flaky 时, 绿卡也会带上 flaky 计数
        title = f"{name}: PASSED ({plural(len(counted), 'job')}{flaky_note})"
        color = "green"

```

tests/ci/test/test_lark_notify.py

行为契约的全部 13 个新用例都集中在这里：重跑触发、防循环、cancelled / manual 豁免、dry-run、拒跑兜底、HTTP 请求格式，以及 flaky 计数的标题断言，是理解新状态机的唯一完整入口。

```
def test_first_failed_nightly_attempt_reruns_its_failed_jobs_instead_of_posting(monkeypatch):
    posted = []
    monkeypatch.setattr(HANDLER, "post_card", lambda *values: posted.append(values))
    # 重跑前不允许任何 AI 分析，attempt 1 唯一动作就是发起重跑
    monkeypatch.setattr(HANDLER, "analyze_failures", lambda **kwargs: pytest.fail("analysis
    before the rerun"))
    gh = FakeGitHub(run(), [job(), job(20, "pass", "success")])
    HANDLER.cmd_ci_status(args(), gh)
    # 关键契约：只请求重跑，不发布任何卡片
    assert gh.rerun_calls() == [("rerun_failed_jobs", 123)] and posted == []

def test_jobs_fixed_by_the_rerun_count_as_flaky_not_failed():
    previous = {"flaky": job(10, "flaky"), "still": job(19, "still")}
    current = [job(20, "still"), job(11, "flaky", "success")]
    card = HANDLER.render_ci_status(run(run_attempt=2), current, previous)
    # 标题只统计 attempt 2 仍失败的 job，被重跑修复的单独记为 flaky
    assert card["card"]["header"]["title"]["content"] == "Nightly Test: FAILED (1 of 2 jobs, 1 flaky)"
    content = markdown(card)
    assert "***Flaky, passed on rerun (1)**" in content and "***Still failing (1)**" in content
```

评论区精华

本 PR 没有实质性 review 评论或讨论线程：claude[bot] 只发布了仓库配置的引导说明，guapisolo 与 yushengsu-thu 直接 APPROVED，无 review comments。唯一的技术不确定性由作者本人写在 PR body 的“Not validated until it runs on main”一节：[workflow_run: completed](#) 对 [github-actions\[bot\]](#) 以 job token 发起的 rerun attempt 是否触发尚未验证；作者说明该模式在手动 rerun 时已验证（notifier run 34672389600 在 attempt 2 完成时触发），仓库内常见的 auto-retry 方案依赖同一行为，但 miles 仓库本身没有实际运行过该路径。作者给出的 fallback 是改用 CI analysis App 的 token 并配 [permission-actions: write](#)。

- 暂无高价值评论线程

风险与影响

- 风险：
 1. 未验证的事件触发路径（核心风险）：[workflow_run: completed](#) 是否对 job token 发起的 rerun attempt 触发，仓库内没有实证。若触发失败，卡片永远不会发布，nightly 失败将静默丢失。作者计划在第一次失败 nightly 后检查 PR Test 是否出现两个 attempt 且 notifier 只跑一次。
 1. 权限面扩大：[ci-lark-notify.yml](#) 把 [actions: read](#) 提升为 [actions: write](#)，notifier job 的 token 现在具备发起 rerun-failed-jobs 的能力；虽然测试断言了这是唯一写权限（

contents / pull-requests 仍为 read)，但 token 泄露或脚本注入的影响面变大了。

2. 运行成本与延迟翻倍：失败 job 和 check-all-jobs 会运行两次（第一次失败 + 重跑），卡片到达时间从 cron 后约 2 小时推迟到 attempt 2 全部完成之后；周六的 weekly 同样是 scheduled run，也会被纳入自动重跑，成本随之增加。
3. flaky 判定依赖 job 名匹配：diff 基于 `failed_job_names` 在 attempt 1 / attempt 2 之间做 job 名匹配，若 job 名因环境差异变化或 attempt 2 新增 job（由重跑依赖解锁），会被归入 new failures 而非 flaky，统计口径依赖既有 diff_attempts 的语义。
4. 新路径缺少线上实证：dry-run 只在真实 run 上打印预告，真正 rerun 请求的端到端行为要等 main 上第一次失败 nightly 才能确认，属于上线的固有观察期。- 影响：影响范围集中在 CI 通知链路（[.github/workflows/scripts/lark_notify.py](#)、[ci-lark-notify.yml](#) 及对应测试），不涉及训练、推理或 rollout 代码。对团队的日常影响：nightly / weekly 红卡更可信，flaky 与真实失败分开呈现，减少人工重跑与手动核对；失败 job 的总执行次数增加，GPU / CPU 测试资源消耗相应上升。对通知系统的语义是一次契约变更：“失败”从“一次尝试失败”变成“重跑后仍失败”，所有依赖旧卡片语义的脚本或人工习惯需要跟随调整。- 风险标记：权限提升 actions: write, workflow_run 触发路径未验证，失败 job 成本翻倍，CI 报告路径变更，新路径缺少线上实证

关联脉络

- PR #3212 ci(lark): give each failure block room in the evidence packet: 同一 lark_notify.py 与 test_lark_notify.py 的卡片渲染演进，本次的 flaky 区块是对失败区块布局的又一扩展。
- PR #3197 ci(lark): collapse jobs that failed identically into one card row: 同一通知脚本的失败行聚合逻辑，与本次 flaky / still / new 分区配套。
- PR #3168 ci(lark): report every failing test in a job: 同一 lark_notify.py 与失败分析管线，本 PR 复用其 AI 分析只读 App token，并只给 job token 新增 actions: write。