PR #2584 完整报告

radixark/miles

fix(ci): honor nightly fast-fail bypass

合并时间：2026-08-18 03:31

原文链接：http://prhub.com.cn/radixark/miles/pull/2584

执行摘要

- 一句话：修复 nightly 快速失败绕过被 Stage A 设置失败吞掉
- 推荐动作：值得 CI 维护者快速精读，是 GitHub Actions 门控条件设计的典型反面教材与修正样例：可复用工作流的 setup 阶段失败不会以普通 job 'failure' 呈现，门控条件应避免对精确状态值耦合，策略输出应作为独立信号消费。核心设计决策是「policy 输出与 job 状态解耦」，可推广到其他 CI 门控场景。

功能与动机

PR body 指出 Nightly run 32041342404 在 policy 输出 bypass_fastfail=true 的情况下仍跳过了 docker-build，原因是 stage-a-cpu 在设置 astral-sh/setup-uv@v5 时失败，导致所有 GPU 阶段被跳过。根因是 jobs.docker-build.if 与 GPU job 门控把 bypass_fastfail 嵌套在 needs.stage-a-cpu.result == 'failure' 之后，而可复用矩阵的 setup 失败在 caller 门控处并不满足精确的 'failure' 状态，使 bypass 完全失效。

实现拆解

1. 定位根因：在 .github/workflows/pr-test.yml 中，docker-build 与五个 GPU 阶段（stage-b-2-gpu-h200、stage-c-8-gpu-h100、stage-c-8-gpu-h200、stage-c-4-gpu-h200、stage-c-2-gpu-h200）的门控条件都写成 (needs.stage-a-cpu.result == 'failure' && needs.resolve-ci-policy.outputs.bypass_fastfail == 'true')。可复用工作流 setup 阶段失败时，caller 侧聚合结果并非精确的 'failure'，于是 bypass 分支永不触发。
2. 放宽门控条件：将上述六处条件统一简化为独立的 needs.resolve-ci-policy.outputs.bypass_fastfail == 'true'，使 bypass 成为与 Stage A 成功并列的平等放行路径；Stage A 无论失败、跳过还是其他非成功状态，只要 policy 明确要求快速失败绕过，下游镜像构建与 GPU 测试就继续推进。
3. 收紧测试 seam：在 tests/ci/test/test_run_suite.py 中，test_docker_build_waits_for_cpu_gate_and_preserves_bypass 把 needs.stage-a-cpu.result == 'failure' 的断言从「必须存在」反转为「不得存在」；test_gpu_gates_consume_shared_bypass_output 对五个 GPU 门控整体追加同样的反向断言，形成防止回归的静态护栏。
4. 保留原有守卫：cancellation（always() && !cancelled()）、PR 关闭（github.event.action != 'closed'）、policy 自身成功、镜像构建成功、以及 skipped_stages 阶段选择逻辑均原样保留，bypass 只放宽 Stage A 相关路径，不扩大其

他守卫的语义。

5. 验证配套：作者执行 pytest -q tests/ci/test/test_run_suite.py tests/ci/test/test_ci_policy.py 得到 103 passed，并通过 git diff --check 与含 YAML 校验的 commit hooks。

关键文件：

- .github/workflows/pr-test.yml（模块 CI 工作流；类别 infra；类型 infrastructure；符号 docker-build, stage-b-2-gpu-h200, stage-c-8-gpu-h100, stage-c-8-gpu-h200）：修复主体：docker-build 与五个 GPU 阶段的门控条件从「Stage A 失败且 bypass」的嵌套结构改为「bypass_fastfail=true」独立放行，是本次 bugfix 的核心变更文件。
- tests/ci/test/test_run_suite.py（模块 CI 测试；类别 test；类型 test-coverage；符号 test_docker_build_waits_for_cpu_gate_and_preserves_bypass, test_gpu_gates_consume_shared_bypass_output）：测试防护：将 failure 断言反转为否定断言，并对 GPU 门控整体追加同款检查，防止未来重新引入对精确 'failure' 状态的嵌套依赖。

关键符号：test_docker_build_waits_for_cpu_gate_and_preserves_bypass, test_gpu_gates_consume_shared_bypass_output

关键源码片段

.github/workflows/pr-test.yml

修复主体：docker-build 与五个 GPU 阶段的门控条件从「Stage A 失败且 bypass」的嵌套结构改为「bypass_fastfail=true」独立放行，是本次 bugfix 的核心变更文件。

```
  # docker-build 门控：Stage A 无论成败，只要 policy 明确输出 bypass_fastfail=true
  # 就继续构建镜像；取消、PR 关闭、policy 自身失败等守卫仍然前置生效。
  docker-build:
    needs: [resolve-ci-policy, stage-a-cpu]
    if: |
      always() && !cancelled() &&
      github.event.action != 'closed' &&
      needs.resolve-ci-policy.result == 'success' &&
      (needs.stage-a-cpu.result == 'success' ||
       needs.resolve-ci-policy.outputs.bypass_fastfail == 'true')
    uses: ./.github/workflows/_build-pr-ci-image.yml
    secrets: inherit

  # 五个 GPU 阶段共用同一组守卫：先要求镜像构建成功、且未被 skip 列表排除，
  # 然后 Stage A 成功或者 bypass_fastfail=true 均可放行，覆盖 setup 失败的边界场景。
  stage-b-2-gpu-h200:
    needs: [resolve-ci-policy, stage-a-cpu, docker-build, resolve-ci-image]
    if: |
      always() && !cancelled() &&
      github.event.action != 'closed' &&
      needs.resolve-ci-policy.result == 'success' &&
      needs.resolve-ci-image.result == 'success' &&
```

```
    !contains(fromJSON(needs.resolve-ci-policy.outputs.skipped_stages || '[]'), 'stage-b-2-gpu-
    h200') &&
    (needs.stage-a-cpu.result == 'success' ||
     needs.resolve-ci-policy.outputs.bypass_fastfail == 'true')
  uses: ./.github/workflows/_run-ci.yml
  with:
    runs_on: '["h200", "2gpu"]'
  strategy:
    fail-fast: false
    max-parallel: ${{ needs.resolve-ci-policy.outputs.cadence == 'weekly' && 1 || 2 }}
```

tests/ci/test/test_run_suite.py

测试防护：将 failure 断言反转为否定断言，并对 GPU 门控整体追加同款检查，防止未来重新引入对精确 'failure' 状态的嵌套依赖。

```
    def test_docker_build_waits_for_cpu_gate_and_preserves_bypass(self):
        workflow = self._workflow()
        caller = workflow.split("  docker-build:", 1)[1].split("  resolve-ci-image:", 1)[0]

        # 核心守卫仍然保留：取消、PR 关闭、policy 成功、Stage A 成功
        assert "needs: [resolve-ci-policy, stage-a-cpu]" in caller
        assert "always() && !cancelled()" in caller
        assert "github.event.action != 'closed'" in caller
        assert "needs.resolve-ci-policy.result == 'success'" in caller
        assert "needs.stage-a-cpu.result == 'success'" in caller
        # 关键 seam：bypass 必须是独立替代条件，禁止重新耦合精确 'failure' 状态
        assert "needs.resolve-ci-policy.outputs.bypass_fastfail == 'true'" in caller
        assert "needs.stage-a-cpu.result == 'failure'" not in caller
        assert "stage-b-cpu" not in caller
        assert "uses: ./.github/workflows/_build-pr-ci-image.yml" in caller
        assert "secrets: inherit" in caller

    def test_gpu_gates_consume_shared_bypass_output(self):
        workflow = self._workflow()
        gpu_stages = workflow.split("  stage-b-2-gpu-h200:", 1)[1]
        bypass_gate = "needs.resolve-ci-policy.outputs.bypass_fastfail == 'true'"
        # 五个 GPU 阶段必须全部消费 shared bypass 输出，且不得依赖精确 'failure'
        assert gpu_stages.count(bypass_gate) == 5
        assert gpu_stages.count("needs.resolve-ci-policy.result == 'success'") == 5
        assert gpu_stages.count("needs.resolve-ci-image.result == 'success'") == 5
        assert "needs.stage-a-cpu.result == 'failure'" not in gpu_stages
```

评论区精华

本 PR 没有实质性的 reviewer 讨论：claude[bot] 仅自动提示本仓库配置了手动 review，yueming-yuan 直接 APPROVED 且未留文字。作者在 PR body 的 Review Focus 中自查了两个验证点：

- 确认 bypass 应覆盖 Stage A 的每一种非成功结果，但不能绕过 cancellation、policy 失败、镜像失败或阶段选择守卫。
- 确认 test_run_suite.py 的 seam 断言能拒绝未来重新引入精确 Stage A 'failure' 依赖。这两点恰好构成修复的语义边界，可视为作者自证的设计意图。
- bypass 语义边界的自查验证 (question): 最终实现将 bypass 提升为独立替代条件，并保留全部原有守卫，测试断言反转为否定形式。

风险与影响

- 风险：
  1. fail-open 语义扩大：修复后只要 policy 输出 bypass_fastfail=true，Stage A 的真实回归（如代码级测试失败）也会被放行到 GPU 阶段，失败暴露延迟且浪费 GPU 资源；该风险由 ci_policy 的输出正确性兜底，本次未改动 policy 生成逻辑。
  2. 测试覆盖的静态局限：tests/ci/test/test_run_suite.py 只做字符串级断言，无法模拟 GitHub 在 reusable workflow setup 失败时 caller 侧真实的 job result 语义，也无法验证「非 failure 状态确实能被 bypass 覆盖」这一运行时行为。
  3. 核心 CI 门控路径变更：.github/workflows/pr-test.yml 是 PR 与 nightly 共用的主干工作流，若条件语法或守卫搭配有误，会直接导致镜像构建或 GPU 阶段异常放行；但改动仅删去一层嵌套，风险相对可控。 - 影响：影响范围限定在 CI 基础设施层：nightly 与 PR 运行在 Stage A 基础设施故障（如 setup-uv 安装失败）时不再中断整轮验证，docker-build 与五个 GPU 阶段能按 nightly 策略继续推进。对开发团队而言，CI 可靠性提升、夜间结果更稳定；对产品用户无任何功能影响。改动文件仅两个，共 15 行，回滚成本低。 - 风险标记：门控放宽 fail-open，静态断言无法覆盖运行时语义，核心 CI 门控路径变更

关联脉络

- PR #2529 refactor(ci): gate PR image builds on CPU tests: 本 PR 修复的门控嵌套结构正是 2529 引入的：docker-build 依赖 stage-a-cpu 并把 bypass 嵌套在 'failure' 判断之后；2584 修正了该结构的边界缺陷。
- PR #2499 feat(ci): add weekly full-suite cadence: 2499 引入了 resolve-ci-policy 输出与 skipped_stages 机制，bypass_fastfail 的生成语义来自 ci_policy；本 PR 确保该输出在 setup 失败的边界场景下仍能被下游门控正确消费。