PR #45274 完整报告

vllm-project/vllm

[CI] ci-fetch-log.sh: fetch all failed jobs from a build URL or PR number

合并时间：2026-06-12 15:42

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45274

执行摘要

- 一句话：增强 CI 日志获取脚本，支持批量拉取失败 job
- 推荐动作：对于团队中经常调试 CI 失败的成员，建议立即采用新用法；对于 Agent 开发者，AGENTS.md 中的更新是必读项。整体设计合理，可读性好，建议精读 ci-fetch-log.sh 以理解其逻辑。

功能与动机

之前的 ci-fetch-log.sh 需要用户提供 job UUID，且 Web UI URL（/list?sid=）会返回 403，因为 sid 是 step uuid 而非 job uuid。此次更新旨在让 CI 日志获取对人和 agent 都更直观，一个命令即可获得所有失败日志。

实现拆解

1. 重写 ci-fetch-log.sh：增加 --pr 参数，通过 gh pr checks 自动解析最新 build URL；新增解析 build URL 和 Web UI URL 的逻辑，通过 buildkite 公共 jobs 端点获取所有失败 job 的 UUID；增加 --soft 和 --all 参数控制 job 范围。
2. 新增 SID 解析逻辑：对于包含 ?sid= 的 URL，通过 /data/jobs 端点将其转换为实际 job UUID。
3. 改进输出命名：日志文件自动命名为 ci--.log，并打印 \t 供脚本处理；传入 - 可将日志流式输出到 stdout。
4. 增强 ci-clean-log.sh：增加内联时间戳标记（ESC _bk;t= BEL）的过滤，使日志更干净。
5. 更新文档：在 docs/contributing/ci/failures.md 中替换旧示例，新增 --pr 和 build URL 示例；在 AGENTS.md 中添加 "Diagnosing CI failures" 小节，让 AI 代理可以直接学会使用方法。
6. 保留向后兼容：旧格式 # 和 仍可工作。

关键文件：

- .buildkite/scripts/ci-fetch-log.sh（模块 CI 脚本；类别 other；类型 core-logic；符号 usage, die）：核心脚本，完全重写以支持批量获取失败 job 日志
- docs/contributing/ci/failures.md（模块 文档；类别 docs；类型 documentation）：更新文档说明新脚本用法，方便开发和代理使用

- AGENTS.md（模块 Agent 指引；类别 docs；类型 documentation）：为 AI 代理添加 CI 诊断指引，提高代理自主排查能力
- .buildkite/scripts/ci-clean-log.sh（模块 CI 脚本；类别 other；类型 core-logic）：增加内联时间戳标记清理，提升日志可读性

关键符号：usage, die

关键源码片段

.buildkite/scripts/ci-fetch-log.sh

核心脚本，完全重写以支持批量获取失败 job 日志

```
# ci-fetch-log.sh —— 从 Buildkite 获取 CI 日志（无需登录）
set -euo pipefail

ORG="vllm"
PIPELINE="ci"
UA="vllm-ci-fetch-log"
UUID_RE='[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}'

SCOPE="failed" # 默认只获取 hard-failed job

# 解析前置 --soft/--all 参数
while :; do
  case "${1:-}" in
    --soft) SCOPE="soft" ;;
    --all) SCOPE="all" ;;
    *) break ;;
  esac
  shift
done

# 根据参数类型设置 BUILD、JOB、SID 变量
case "${1:-}" in
--pr)
  PR="${2:-}"
  # 利用 gh pr checks 提取当前 PR 的最新 build URL
  URL=$(gh pr checks ${PR:+"$PR"} --repo vllm-project/vllm 2>/dev/null |
    grep -oE "https://buildkite.com/${ORG}/${PIPELINE}/builds/[0-9]+" |
    sort -t/ -k7 -n | tail -1 || true)
  [ -n "$URL" ] || die "No Buildkite build found"
  BUILD="${URL##*/}"
  ;;
https://*)
  BUILD=$(echo "$1" | sed -nE 's#.*/builds/([0-9]+).*#\1#p')
  JOB=$(echo "$1" | grep -oE "#${UUID_RE}" | head -1 | cut -c2- || true)
  SID=$(echo "$1" | grep -oE "[?&]sid=${UUID_RE}" | head -1 | sed 's/.*sid=//' || true)
  ;;
# 旧格式 <build> <uuid> 处理（省略）
```

```
esac

# 如果只指定了 build URL（无 JOB 和 SID），则获取所有失败 job
if [ -n "$BUILD" ] && [ -z "${JOB:-}" ] && [ -z "${SID:-}" ]; then
  echo "Getting jobs for build $BUILD (scope=$SCOPE)..."
  # 通过 Buildkite 公共 jobs API 获取 job 列表
  BASE="https://buildkite.com/${ORG}/${PIPELINE}/builds/${BUILD}"
  curl -sf -A "$UA" "${BASE}/data/jobs" | jq -c '.[]' | while IFS= read -r job; do
    state=$(echo "$job" | jq -r '.state')
    case "$SCOPE" in
      failed) [ "$state" = "failed" ] || continue ;;
      soft) [ "$state" = "failed" ] || [ "$state" = "soft_failed" ] || continue ;;
      all) ;;
    esac
    # 提取 job UUID 和 name，下载并清洗日志
    uuid=$(echo "$job" | jq -r '.id')
    name=$(echo "$job" | jq -r '.name')
    clean_name=$(echo "$name" | sed 's/[^a-zA-Z0-9_.-]/-/g')
    log_file="ci-${BUILD}-${clean_name}.log"
    # 下载、清洗、保存…（略）
    echo "$log_file => $name"
  done
fi
```

评论区精华

无实质性讨论，PR 得到 Mergify approvals 后合入。

- 暂无高价值评论线程

风险与影响

- 风险：低风险。变更仅限于 CI 辅助脚本和文档，不涉及核心功能代码。新脚本中的 gh pr checks 命令可能在某些环境中不存在，已通过 condition 和 die 提示处理。脚本需要 write 权限，已在注释中说明。
- 影响：改善开发者调试 CI 失败的体验：一个命令即可获取所有失败 job 日志，降低心智负担；对 Agent（如 AI 编码助手）友好，可以直接通过 AGENTS.md 了解使用方法。影响范围限于 CI 调试流程，无生产环境影响。
- 风险标记：依赖外部命令，缺少测试覆盖

关联脉络

- 暂无明显关联 PR