

PR #55014 完整报告

vllm-project/vllm

[ROCm][CI] Build and publish TheRock nightly docker images

合并时间: 2026-09-04 10:34

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/55014>

执行摘要

- 一句话: 为 TheRock (ROCK) Docker 镜像构建搭建 CI/CD 管道, 并增强发布脚本的多变体支持。
- 推荐动作: 此 PR 是 ROCm 基础设施建设的重要一步, 值得 CI/CD 维护者和 ROCm 相关开发者精读。其核心价值在于: 1) 为新的构建变体提供了完整的 CI/CD 集成模板; 2) 展示了如何通过参数化和元数据传递来扩展现有的复杂发布管道。重点关注以下设计决策:
 - ECR 缓存策略 (基于文件哈希的缓存键)。
 - 发布脚本从硬编码到参数化的重构方式, 以及错误处理的增强。
 - 新构建步骤与现有流程的并行关系 (共享清理预算、依赖 block 步骤)。
 - PR body 中对“清理策略冲突”和“未来扩展性”的主动说明, 体现了良好的工程权衡意识。此 PR 的风险较低, 且作者已明确标记了已知问题和后续计划, 适合作为基础设施演进的参考案例。

功能与动机

根据 PR body, `docker/Dockerfile.rock_base` 和 `docker/Dockerfile.rock` 在 PR#49925 中被引入, 但未被任何流水线引用。本次变更的目标是复制 ROCm 发布流水线的关键步骤 (Jobs 1 和 6, 加上 nightly 发布), 为这些 TheRock Dockerfile 镜像构建提供完整的 CI/CD 支持, 使得在 `NIGHTLY=1` 时能自动构建并推送到 `vllm/vllm-openai-rocm` 仓库, 使用 `rocm714` 标签变体进行区分, 为未来更多的 TheRock 变体 (如 `rocm100`) 预留了扩展空间。

实现拆解

1. 构建管道配置扩展: 在 `.buildkite/release-pipeline.yaml` 中, 在现有 ROCm nightly 发布步骤之后, 新增了一个名为 "Build ROCK Image" 的步骤组。该组包含两个核心步骤:
 - `build-rock-base-image`: 负责构建基础镜像。它使用 `Dockerfile.rock_base`, 并实现了基于文件哈希的 ECR 镜像缓存逻辑 (`CACHE_KEY`), 只有缓存未命中时才进行实际构建, 并将生成的 ECR 标签元数据保存下来供下游步骤使用。
 - `build-rock-release-image`: 负责构建最终的 `vllm-openai` 发布镜像。它从上游步骤拉取基础镜像 ECR 标签, 基于 `Dockerfile.rock` 构建, 并将构建产物推送到 ECR。
2. 发布脚本参数化与加固: 重构了 `.buildkite/scripts/push-nightly-builds-rocm.sh`。脚本现在支持可选参数 `[ECR_TAG_SUFFIX TAG_VARIANT]`, 用于指定构建变体 (如 `rock rocm714`)。当不带参数时, 行为与原 ROCm 流程保持一致; 带参数时, 会基于参数生成

相应的 Docker Hub 标签（如 nightly-rocm714）。同时增强了错误处理和本地测试支持（`DRY_RUN + BASE_ECR_IMAGE`）。

3. Nightly 发布与清理：在新增的 "Build ROCK Image" 步骤组中，添加了 "Publish nightly ROCK image to DockerHub" 步骤。该步骤依赖于 `build-rock-release-image`，仅在 `NIGHTLY=1` 时运行，调用参数化后的脚本 `bash .buildkite/scripts/push-nightly-builds-rocm.sh rocm rocm714`，并随后调用清理脚本，使用 `nightly-rocm714-` 和 `base-nightly-rocm714-` 前缀来管理该变体的镜像保留数量。

关键文件：

- `.buildkite/release-pipeline.yaml`（模块 构建管道；类别 `config`；类型 `configuration`）：新增了完整的 ROCK 镜像构建与发布流水线步骤组，是本次 CI/CD 集成的核心配置变更。
- `.buildkite/scripts/push-nightly-builds-rocm.sh`（模块 发布脚本；类别 `other`；类型 `core-logic`；符号 `push_nightly_builds_rocm`）：脚本被重构以支持参数化调用，是支持多构建变体（ROCm apt 和 TheRock）发布的关键逻辑文件。

关键符号：未识别

关键源码片段

`.buildkite/scripts/push-nightly-builds-rocm.sh`

脚本被重构以支持参数化调用，是支持多构建变体（ROCm apt 和 TheRock）发布的关键逻辑文件。

```
#!/usr/bin/env bash
# --- push-nightly-builds-rocm.sh 核心逻辑重构 ---
set -euxo pipefail

# 参数解析：支持 0 或 2 个参数
DOCKERHUB_REPO="vllm/vllm-openai-rocm"
case "$#" in
  0) # 默认行为，用于 ROCm apt 构建
    ECR_TAG_SUFFIX="rocm"
    TAG_VARIANT=""
    ;;
  2) # 用于 TheRock 等其他变体
    if [[ -z "$1" || -z "$2" ]]; then
      usage; exit 2
    fi
    ECR_TAG_SUFFIX="$1"
    TAG_VARIANT="$2"
    ;;
  *) usage; exit 2 ;;
esac

# 根据 ECR_TAG_SUFFIX 从 Buildkite 元数据获取对应的 ECR 基础镜像标签
BASE_METADATA_KEY="{ECR_TAG_SUFFIX}-base-ecr-tag"
# ... 错误处理和元数据获取 ...
```

```
# 构建 Docker Hub 标签名
VARIANT_SUFFIX=""
[[ -n "$TAG_VARIANT" ]] && VARIANT_SUFFIX="-${TAG_VARIANT}"
BASE_TAG_NAME="base-nightly${VARIANT_SUFFIX}" # 如 "base-nightly" 或 "base-nightly-rocm714"
TAG_NAME="nightly${VARIANT_SUFFIX}" # 如 "nightly" 或 "nightly-rocm714"
BASE_TAG_NAME_COMMIT="base-nightly${VARIANT_SUFFIX}-${BUILDKITE_COMMIT}"
TAG_NAME_COMMIT="nightly${VARIANT_SUFFIX}-${BUILDKITE_COMMIT}"

# ... 从 ECR 拉取、打标签、推送到 Docker Hub 的流程 ...
```

评论区精华

讨论主要集中在实现细节和代码风格上：

- **DOCKER_BUILDKIT** 环境变量的必要性：rasmith 多次询问在使用 docker buildx build 时是否还需要设置 DOCKER_BUILDKIT=1。micah-wil 指出这可能是从现有 ROCm 构建步骤中复制过来的残留，可能并不必要，建议在未来的清理中移除。作者 Rohan138 同意留待后续清理。
- 依赖文件处理：rasmith 提到是否需要在构建前将 build/requirements/rock.txt 拷贝为 rocm.txt。Rohan138 解释说这两个文件目前内容基本相同，且主要使用 Dockerfile 作为构建源，因此暂不处理，留待未来统一。
- 版本命名约定与可维护性：micah-wil 在 approved 评论中提出了重要的前瞻性问题：随着 ROCm 版本演进（如 7.2 -> 10），**nightly-rocm714** 这种硬编码的标签是否合适，以及当前 PR 是否会随 ROCm 7.2 的淘汰而被 revert。这揭示了当前实现是针对特定版本的过渡方案，需要在架构上保持对版本变量化的考虑。
 - DOCKER_BUILDKIT 环境变量是否冗余 (style)：作者 Rohan138 表示认同，并同意将其留作后续清理工作。当前 PR 保留该设置以维持与现有流程的一致性。
 - 依赖文件 rock.txt 和 rocm.txt 的处理 (correctness)：决定留待未来 PR 进行统一和清理。当前变更不影响构建正确性。
 - 硬编码版本标签与未来维护性 (design)：这揭示了当前实现是一个针对特定版本（ROCm 7.14）的过渡性方案，需要在未来版本升级时重新评估和更新标签策略。设计本身已预留了扩展空间（如 nightly-rocm100）。
 - Nightly 镜像清理策略的潜在冲突 (design)：此风险被明确标记，但决定不在本 PR 中解决（避免复杂化），留待后续统一处理清理前缀。

风险与影响

- 风险：
 1. 清理策略的潜在冲突：现有 ROCm nightly 清理脚本使用 startswith("nightly-")，这也会匹配到新增的 nightly-rocm714-<commit> 标签。作者在 PR body 中已明确指出这一点：两种变体将共享同一个 14 标签的保留预算。这意味着一次密集的 TheRock nightly 构建可能意外地过早清除普通的 ROCm nightly 镜像，存在间接影响。此风险已被标记，但未在当前 PR 中解决。

2. 首次构建的超时风险：基于文件哈希的 ECR 缓存 (CACHE_KEY) 机制依赖于首次构建能够成功完成。PR 的最后一个 commit 将 base 镜像构建超时时间从 180 分钟提高到 360 分钟，正是为了解决冷启动构建（编译 AITER、flash-attention）可能超时的风险。这表明存在一定的初始稳定性风险。

3. 发布流程一致性：脚本重构增强了通用性，但也意味着所有 ROCm-family 镜像的发布都依赖于同一个脚本和相似的清理逻辑，增加了共享组件故障的影响面。不过，通过参数化隔离了不同变体的具体标签，控制了风险。

- 影响：此 PR 是一次纯粹的 CI/CD 基础设施变更，其影响主要局限于构建和发布系统：
 - 对用户：无直接影响。不涉及任何 vLLM 运行时代码、模型或 API。
 - 对系统：在 Docker Hub 的 vllm/vllm-openai-rocm 仓库中引入了新的镜像标签变体（nightly-rocm714 和 base-nightly-rocm714）。未来当 TheRock 构建稳定后，用户将能拉取到基于 TheRock wheels 构建的 ROCm 容器。
 - 对团队：极大地提升了 TheRock 构建路径的自动化水平。维护 ROCm 构建的工程师（如 AMD 团队）现在可以像管理标准 ROCm 构建一样管理 TheRock 构建的 nightly 交付。同时，参数化的发布脚本为未来引入更多构建变体（如 Ubuntu 24.04、不同 ROCm 版本）奠定了基础。
 - 风险标记：CI/CD 管道变更，共享清理策略冲突，冷启动构建超时风险，版本标签硬编码

关联脉络

- PR #49925 [ROCm] Add Dockerfile.rock_base and Dockerfile.rock: 本 PR 直接依赖于 #49925 中引入的 Dockerfile.rock_base 和 Dockerfile.rock 文件，为其搭建 CI/CD 流水线。
- PR #52826 [ROCm] Bump AITER to 0.1.21.post1: 同属 ROCm 基础设施和内核维护的范畴，体现了 ROCm 构建路径和依赖的持续更新演进。