

PR #2805 完整报告

radixark/miles

rollout: shared miles-side layer for agent-function legs (session URL, sandbox credentials)

合并时间: 2026-09-02 11:17

原文链接: <http://prhub.com.cn/radixark/miles/pull/2805>

执行摘要

- 一句话: 将三个 agent 功能脚本中重复的 session URL 和沙箱凭证处理逻辑提取到共享包 miles/rollout/agentic/。
- 推荐动作: 这是一个值得精读的高质量重构 PR。尽管行为无变化, 但其设计决策 (安全合约、torch-free 约束、测试策略) 和实施方式 (结构化提交、机械性证明) 都值得学习。对于关注 agentic rollout、沙箱集成或安全凭证管理的工程师, 本 PR 提供了很好的参考。核心风险点在于移动的正确性和共享包对 torch-free 的维护, 已通过测试得到缓解。

功能与动机

OpenEnv、NeMo Gym 和 Harbor 三个 agent 功能的启动器和 worker 侧代码中存在完全相同的 session URL 解析逻辑 (`resolve_session_url`) 和沙箱凭证处理合约 (`PROVIDER_CREDENTIALS`, `sandbox_key_supply` 等)。这种重复代码带来了维护负担和潜在的漂移风险。PR 描述明确指出: 'Moves the logic that the OpenEnv, NeMo Gym and Harbor agent-function legs had in identical copies into a shared package', 旨在将这些共享逻辑集中到一个地方, 并为其建立直接、可独立测试的单元测试。

实现拆解

1. 创建共享包 `miles/rollout/agentic/`: 新增 `session.py` 和 `credentials.py` 两个核心模块。
`session.py` 中的 `resolve_session_url` 函数负责追加 `/v1` 后缀并根据 `MILES_ROUTER_EXTERNAL_HOST` 环境变量重写 URL 主机。`credentials.py` 集中定义了沙箱供应商凭证规范表 `PROVIDER_CREDENTIALS` (包含 Daytona, E2B, Modal), 并实现了四个关键函数: `sandbox_key_supply` (确保只转发密钥文件路径, 不转发密钥值)、`forward_address` (验证地址型环境变量不包含凭证信息, 通过拒绝含 '@' 的值)、`preflight_sdk` (在启动时预检 SDK 是否安装, 避免运行时失败)、`resolve_provider_api_key` (worker 侧解析 API 密钥, 优先使用环境变量, 其次读取文件)。
2. 重构现有代码: `examples/experimental/openenv/openenv_launch_common.py` 中的本地凭证处理逻辑 (`_PROVIDER_CREDENTIALS`, `_sandbox_key_supply`, `_forward_address`, `_preflight_sdk`) 被移除, 改为从新共享包导入公共版本。
`examples/experimental/nemo-gym/nemogym_agent_function.py` 和 `examples/experimental/openenv/openenv_agent_function.py` 中的本地 `_resolve_session_url` 函数被删除, 改为从 `miles.rollout.agentic.session` 导入

resolve_session_url。examples/experimental/openenv/tb2_sandbox_recipe.py 中的 resolve_api_key 函数被重命名并移至 credentials.py，更名为 resolve_provider_api_key。

3. 全面测试覆盖：新增 tests/fast/rollout/agentic/ 测试目录，包含 test_session.py 和 test_credentials.py。这些测试使用 tmp_path 和 monkeypatch 模拟环境变量和文件系统，验证了 resolve_session_url 的 URL 重写行为、sandbox_key_supply 的所有分支（文件存在 / 不存在 / 为空 / 不可读、环境变量供给）、forward_address 对包含 '@' 的 URL 的拒绝、preflight_sdk 的成功与失败路径、resolve_provider_api_key 的优先级逻辑。特别地，test_shared_package_is_torch_free 验证了新共享包在导入时不依赖 torch，确保其能在 CPU-only 主机上被 NeMo Gym 适配器安全导入。
4. 测试职责调整：原有的 tests/fast/examples/experimental/openenv/test_openenv_launch_common.py 被大幅精简，仅保留启动器自身的职责测试（如 test_every_backend_has_a_credential_spec），而将凭证合约的具体行为测试迁移到了共享包的新测试文件中，实现了测试职责的清晰分离。
5. 细节优化：共享模块中移除了日志消息里的 'openenv' 前缀，使输出更通用（如 f"{provider} credential supply: ..."）。PR 的提交历史采用了结构化的 'mechanical_provable' 和 'non_mechanical_provable' 分类，并提供了可复现的证明脚本，展示了高度可追溯的重构过程。

关键文件：

- miles/rollout/agentic/credentials.py（模块 凭证管理；类别 source；类型 core-logic；符号 PROVIDER_CREDENTIALS, forward_address, sandbox_key_supply, preflight_sdk）：新建的共享凭证处理核心模块，定义了供应商凭证规范表和所有关键的密钥供应、地址转发、SDK 预检函数，是本次重构的核心逻辑所在。
- miles/rollout/agentic/session.py（模块 会话路由；类别 source；类型 core-logic；符号 resolve_session_url）：新建的共享 session URL 解析模块，实现了将内部 session 地址重写为 agent 可访问的外部地址的逻辑，是 agent 功能的关键网络入口点。
- tests/fast/rollout/agentic/test_credentials.py（模块 凭证测试；类别 test；类型 test-coverage；符号 test_credential_spec_is_complete, test_forwarded_vars_are_addresses_not_secrets, test_a_forwarded_url_may_not_smuggle_a_credential, _supply）：新建的共享凭证合约测试文件，提供了对 credentials.py 所有函数和安全约束的全面单元测试，是保障本次重构正确性和安全性的关键。

关键符号：resolve_session_url, sandbox_key_supply, forward_address, preflight_sdk, resolve_provider_api_key

关键源码片段

miles/rollout/agentic/credentials.py

新建的共享凭证处理核心模块，定义了供应商凭证规范表和所有关键的密钥供应、地址转发、SDK 预检函数，是本次重构的核心逻辑所在。

- # 该函数负责在启动器侧处理沙箱凭证的供应。
- # 核心安全原则：只转发密钥文件的路径，绝不转发密钥值，

因为 worker 的环境变量会通过 ray 的 runtime_env 以明文形式记录在日志和元数据中。

```
def sandbox_key_supply(
    env: dict[str, str],
    *,
    provider: str,
    key_env_vars: tuple[str, ...],
    file_env_var: str,
    arg_path: str,
    default_path: str,
    provision_hint: str,
) -> None:
    """Key-supply contract, shared by the sandbox backends: rollout workers
    get the provider credential from their OWN environment (e.g.
    platform-injected) or from a file they can read (a dotfile, K8s Secret
    mount, or shared-FS path). The launcher forwards only the file PATH, never
    the value: worker env rides ray's runtime_env, which exec_command_cpu
    echoes into driver logs and ray persists in job metadata, all in
    plaintext."""
    key_file = Path(arg_path or default_path).expanduser()
    try:
        key_present = bool(key_file.read_text(encoding="utf-8").strip())
    except OSError:
        key_present = False
    # 两种供应方式（文件或环境变量）都可接受，但无法从启动器验证 worker 节点的情况。
    # 因此需要明确告知正在使用哪种方式。
    names = " + ".join(key_env_vars)
    if key_present:
        # 文件可读：只将文件路径放入环境变量，由 worker 自己读取。
        env[file_env_var] = str(key_file)
        print(
            f"{provider} credential supply: file {key_file} "
            "(readable here; forwarding the path, workers read it themselves)",
            flush=True,
        )
    elif arg_path:
        # 如果用户显式配置了路径但文件不存在或为空，这是配置错误，应立即失败。
        raise ValueError(f"{file_env_var}={arg_path} is missing or empty")
    elif all(os.environ.get(var, "").strip() for var in key_env_vars):
        # 所有必需的环境变量都已设置（如平台注入），则假定 worker 也有。
        print(
            f"{provider} credential supply: worker environment ({names} "
            "set here; workers are assumed to have them in their env — "
            "single-host inheritance or platform-injected pod env)",
            flush=True,
        )
    else:
        # 既无可读文件，环境变量也不完整，则抛出错误并提供配置指引。
        raise ValueError(
            f"the {provider} sandbox mode needs credentials: put them in a file "
```

```

        f"({key_file}; {file_env_var} overrides) or in the "
        f"environment as {names}. Provision the file with:\n"
        f" {provision_hint}"
    )

```

miles/rollout/agent/session.py

新建的共享 session URL 解析模块，实现了将内部 session 地址重写为 agent 可访问的外部地址的逻辑，是 agent 功能的关键网络入口点。

```

"""The session-scoped policy URL an agent function hands to its agent."""

import os
from urllib.parse import urlparse, urlunparse

def resolve_session_url(base_url: str) -> str:
    """Build the OpenAI-compatible policy URL, rewriting host for off-cluster agents."""
    # 1. 标准化: 追加 /v1 后缀, 使其符合 OpenAI 兼容 API 的路径规范。
    session_url = f"{base_url}/v1"
    # 2. 主机重写: 如果设置了 MILES_ROUTER_EXTERNAL_HOST 环境变量,
    # 则将内部主机地址 (如 10.0.0.1:30000) 替换为外部可路由的地址 (如 trainer.tailnet:
    # 30000) 。
    # 这允许运行在集群外部的 agent 进程访问内部的 session server。
    external_host = os.getenv("MILES_ROUTER_EXTERNAL_HOST")
    if external_host:
        parsed = urlparse(session_url)
        # 保留端口号 (如果存在), 否则只替换主机名。
        netloc = f"{external_host}:{parsed.port}" if parsed.port else external_host
        session_url = urlunparse(parsed._replace(netloc=netloc))
    return session_url

```

评论区精华

Review 讨论极少，仅有 Claude bot 的自动审查和 Shi-Dong 的一句 'LGTM!'。这表明重构的意图清晰，实现机械且正确，无明显争议。

- PR 整体审查与批准 (other): PR 获得批准，未在审查阶段发现技术问题。

风险与影响

- 风险:

1. 重构正确性: 本 PR 主要是代码移动和去重，行为上声明无变化。核心风险在于移动过程中是否遗漏或错误修改了逻辑。鉴于 PR 提供了可重现的机械性证明，并且测试覆盖了凭证合约和 session URL 的所有关键路径，风险较低。但需注意 tb2_sandbox_recipe.py 中 resolve_api_key 函数的重命名和移动 (commit 2cb5357)，以及 tb2_sandbox_daytona.py 和 tb2_sandbox_e2b.py 中导入语句的相应调整，这是最容易出错的环节。

2. 安全边界: `forward_address` 函数通过拒绝包含 '@' 的值来防止 URL 中嵌入凭证, 这是一个关键的安全控制。测试 `test_a_forwarded_url_may_not_smuggle_a_credential` 对此进行了验证。风险在于未来新增的地址型环境变量是否可能意外包含合法字符 '@', 导致误拒。
3. 依赖影响: 新增的 `miles/rollout/agentica/` 包被设计为 `torch-free`, 测试 `test_shared_package_is_torch_free` 确保了这一点。这对其在 CPU-only 环境 (如 NeMo Gym 测试) 中的可用性至关重要。如果未来开发者不慎在此包中引入 `torch` 依赖, 将破坏这一约束。
4. 测试冗余: 重构后, `tests/fast/examples/experimental/openenv/test_openenv_launch_common.py` 中的测试变得非常简单, 仅检查规范的存在性。这可能是预期的, 但需确保这些保留的测试依然有价值。

- 影响:

1. 用户影响: 无直接功能变化, 对终端用户透明。
2. 系统影响: 为 `agent` 功能的基础设施提供了一个更整洁、更安全的共享层。降低了未来为 OpenEnv、NeMo Gym 或 Harbor 添加新沙箱后端时的开发成本和出错概率。凭证处理逻辑的集中化也提升了安全审计的便利性。
3. 团队影响: 为负责 `agentica rollout` 的开发者提供了清晰的扩展点。共享的凭证合约和 `session URL` 解析逻辑减少了跨示例代码的同步负担。详尽的测试为未来修改这些核心逻辑提供了安全网。 - 风险标记: 重构正确性验证, 安全合约维护 (`torch-free`, 凭证转发)

关联脉络

- PR #2801 相关 PR (推测): PR 描述中提到 'independent of #2801/#2802', 表明本 PR 与这些 PR 在功能上相关但独立实施, 可能属于同一条 `agentica rollout` 功能线或基础设施改进。
- PR #2802 相关 PR (推测): 同上, PR 描述中提及与 #2802 独立。
- PR #2804 Tracking Issue: PR 描述中注明 'Tracking: #2804', 表明本 PR 是追踪 Issue #2804 中所述任务的实施。