

# PR #2868 完整报告

radixark/miles

smoke: manual golden episodes on real sandbox APIs

合并时间: 2026-09-09 08:00

原文链接: <http://prhub.com.cn/radixark/miles/pull/2868>

## 执行摘要

- 一句话: 新增手动运行的沙箱冒烟测试脚本, 验证真实沙箱平台完整链路。
- 推荐动作: 这是一个设计良好、文档齐全的基础设施补充。它填补了测试覆盖的关键空白, 且风险可控。建议所有负责沙箱集成或部署验证的工程师精读此 PR, 尤其是:
  1. `run.py` 的整体架构: 学习其正交参数设计、懒加载基准任务和清晰的连接器抽象。
  2. 凭证处理模式: 理解其强制使用文件凭证以模拟真实训练路径的考量。
  3. README 中的使用示例和验证矩阵: 了解如何利用此工具进行快速验证。

## 功能与动机

## 实现拆解

1. 新增沙箱冒烟测试入口脚本(`scripts/sandbox_smoke/run.py`): 这是核心实现。脚本定义了一个 Benchmark 数据类来描述基准测试, 一个 `_tb2_tasks_dir` 函数用于懒加载 Terminal-Bench-2 任务目录 (支持环境变量覆盖或首次使用时自动克隆), 以及 CONNECTORS 字典映射连接器名称到其运行函数。main 函数解析命令行参数, 验证参数组合, 并调用选定的连接器函数执行测试。最终根据返回结果中的 `reward` 和 `exit_status` 判断测试是否通过。
2. 实现 Harbor 连接器运行函数(`_run_harbor`): 此函数封装了通过 Harbor 框架执行一次沙箱试验的逻辑。它动态导入 `harbor` 和 `harbor_agent_function` 模块, 设置必要的环境变量 (如 `HARBOR_TASKS_DIR`, `HARBOR_ENV_TYPE`), 并将 `golden agent` 映射为 Harbor 内部的 `oracle agent`。它通过 `harbor_agent_function.run` 执行任务。
3. 添加配套文档(`scripts/sandbox_smoke/README.md`): 提供了清晰的使用说明, 包括所有命令行标志的含义、不同后端的凭证配置方法 (强调使用文件而非环境变量以与训练路径一致), 以及已验证的组合列表。
4. 编写离线单元测试(`tests/fast/scripts/test_sandbox_smoke.py`): 测试验证了连接器注册表的形状和参数传递的正确性。它使用 `monkeypatch` 来模拟 `sys.argv` 和连接器函数, 确保 main 函数能正确解析参数并将其传递给连接器, 同时验证了非 `golden agent` 必须提供 `--base-url` 的约束。
5. 创建测试目录初始化文件(`tests/fast/scripts/__init__.py`): 作为测试套件的必要配套。

关键文件:

- scripts/sandbox\_smoke/run.py (模块 冒烟测试; 类别 source; 类型 dependency-wiring ; 符号 Benchmark, \_tb2\_tasks\_dir, \_run\_harbor, \_run\_openenv) : 核心实现文件, 包含了整个冒烟测试工具的逻辑, 包括命令行解析、基准测试管理、Harbor 连接器集成和测试执行流程。
- tests/fast/scripts/test\_sandbox\_smoke.py (模块 沙箱测试; 类别 test; 类型 test-coverage; 符号 smoke, test\_connectors\_are\_callable, test\_golden\_run\_wires\_all\_axes\_through, fake\_runner) : 关键的离线测试文件, 验证了冒烟测试工具本身的参数解析和连接器注册表逻辑, 确保工具的基本正确性。
- scripts/sandbox\_smoke/README.md (模块 冒烟测试; 类别 docs; 类型 documentation) : 配套文档, 详细说明了工具的使用方法、参数含义、凭证配置和已验证的组合, 是理解工具用途和正确使用的必要补充。

关键符号: Benchmark, \_tb2\_tasks\_dir, \_run\_harbor, \_run\_openenv, main, test\_connectors\_are\_callable, test\_golden\_run\_wires\_all\_axes\_through, test\_a\_real\_harness\_requires\_a\_model\_endpoint

## 关键源码片段

### tests/fast/scripts/test\_sandbox\_smoke.py

关键的离线测试文件, 验证了冒烟测试工具本身的参数解析和连接器注册表逻辑, 确保工具的基本正确性。

```
# 文件 : tests/fast/scripts/test_sandbox_smoke.py
# 该文件对冒烟测试工具进行离线单元测试。
```

```
import importlib.util
from pathlib import Path
from types import ModuleType
```

```
import pytest
```

```
REPO_ROOT = Path(__file__).resolve().parents[3]
RUN_PY = REPO_ROOT / "scripts" / "sandbox_smoke" / "run.py"
```

```
@pytest.fixture(scope="module")
def smoke() -> ModuleType:
    """动态导入冒烟测试脚本作为模块进行测试。"""
    spec = importlib.util.spec_from_file_location("sandbox_smoke_run", RUN_PY)
    module = importlib.util.module_from_spec(spec)
    spec.loader.exec_module(module)
    return module
```

```
def test_connectors_are_callable(smoke):
    """验证连接器注册表包含预期的键且值都是可调用的。"""
    assert "harbor" in smoke.CONNECTORS
```

```

for name, runner in smoke.CONNECTORS.items():
    assert callable(runner), name

def test_golden_run_wires_all_axes_through(smoke, monkeypatch, tmp_path):
    """验证 connector/backend/agent/task 所有参数都能正确传递到连接器函数；golden
    默认模式不需要 endpoint。"""
    seen = {}

    async def fake_runner(tasks_dir, task, *, backend, agent, base_url):
        """模拟的连接函数，记录所有收到的参数。"""
        seen.update(tasks_dir=tasks_dir, task=task, backend=backend, agent=agent, base_url=
            base_url)
        return {"reward": 1.0, "exit_status": "Submitted"}

    monkeypatch.setitem(smoke.CONNECTORS, "harbor", fake_runner)
    monkeypatch.setenv("TB2_TASKS_DIR", str(tmp_path)) # 环境变量优先于缓存克隆
    monkeypatch.setattr("sys.argv", ["run.py", "--connector", "harbor", "--backend", "daytona"])
    assert smoke.main() == 0
    # 基准测试预设填充了任务；golden 默认值原样传递给连接器
    assert seen["backend"] == "daytona" and seen["task"] == "fix-git" and seen["agent"] == smoke.
    GOLDEN
    assert seen["tasks_dir"] == tmp_path

def test_a_real_harness_requires_a_model_endpoint(smoke, monkeypatch, capsys):
    """验证非 golden agent 必须提供 --base-url 参数。"""
    monkeypatch.setattr(
        "sys.argv", ["run.py", "--connector", "harbor", "--backend", "e2b", "--agent", "mini-swe-
        agent"]
    )
    with pytest.raises(SystemExit) as excinfo:
        smoke.main()
    assert excinfo.value.code == 2
    assert "--base-url" in capsys.readouterr().err

```

## 评论区精华

核心讨论由 [claude\[bot\]](#) 发起，主要围绕两个可选优化点展开，均未构成合并阻塞：

1. 并发克隆竞态条件: [claude\[bot\]](#) 指出 `_tb2_tasks_dir` 中用于克隆的临时目录路径是固定的，如果两个冒烟测试实例并发运行，可能会互相干扰导致克隆失败或损坏。作者 [nblintao](#) 在后续提交中修复了中断克隆后的恢复问题（使用临时目录 + 重命名），但并发问题被视为低概率场景，暂未处理。
2. 未实现连接器的无谓克隆: [claude\[bot\]](#) 观察到 `main` 函数在分派到连接器之前会无条件地解析 `tasks_dir`（可能触发一次完整的 git 克隆），即使连接器（如 `openenv`）尚未实现并会立即抛出 `NotImplementedError`。这可能导致不必要的网络操作和令人困惑的错误信息。这是一个合理的代码结构建议。\* 结论: 这些发现被标记为 `optional`，[claude\[bot\]](#) 明确表

示 “Nothing blocking”，Shi-Dong 最终批准了 PR。

- TB2 任务目录缓存的中断恢复与并发安全 (design): 中断恢复问题已修复。并发竞态被视为已知但低影响的边缘情况，留待未来优化。
- 未实现连接器的无谓克隆 (design): 这是一个合理的代码结构建议，被视为可选优化，未阻塞合并。

## 风险与影响

- 风险：技术风险较低，因为这是一个全新的、独立的开发工具，未修改任何训练代码或 CI 流程。
  1. 凭证安全：风险被主动缓解。README 明确指出应使用文件凭证（如 `~/.config/e2b/api_key`）而非环境变量，并且该工具设计为从持有凭证的任意机器手动运行，避免将凭证存入仓库 secrets。
  2. 工具链依赖：脚本依赖 harbor 包的安装（通过 uv）和特定版本的 Terminal-Bench-2 克隆。如果上游 API 发生变化，工具可能会失效。这是一个维护性风险。
  3. 错误处理：对于并发克隆或未实现连接器触发的无谓克隆等边界情况，错误处理可以更优雅，但当前实现已足够清晰。
- 影响：影响范围非常有限，属于局部工具增强。
  1. 对用户：为开发者和测试人员提供了一个强大的手动验证工具，用于在部署后快速确认沙箱平台的基本功能。提高了基础设施的可观测性和信心。
  2. 对系统：无直接影响。新增的脚本和测试不干扰任何现有生产路径。
  3. 对团队：为未来扩展沙箱集成测试奠定了基础（例如，计划中的 OpenEnv 连接器），并建立了测试不同（连接器、后端）组合的模式。
- 风险标记：工具链依赖

## 关联脉络

- PR #2806 examples: in-process Harbor agent function: 本 PR 的 `_run_harbor` 函数直接调用了 PR#2806 引入的 `harbor_agent_function`，两者共同构成了 Harbor 集成测试的基础。