

PR #3215 完整报告

radixark/miles

harbor daytona: reclaim orphaned sandboxes by default, and a storage override

合并时间: 2026-09-15 03:55

原文链接: <http://prhub.com.cn/radixark/miles/pull/3215>

执行摘要

- 一句话: 修复 Daytona 孤儿沙箱永不回收, 新增存储配额 override
- 推荐动作: 值得精读。核心看点是 `_environment_config()` 中“业务上限 + 安全缓冲 + 显式拒绝危险值”的防御性配置模式, 以及只对特定后端生效的隔离方式。测试围绕 WHY 组织 (如 docstring 说明为何必须回收又不能误停活 trial), 对在 Harbor/ 沙箱供应商上做资源配置的团队有参考价值。改动精炼, 风险集中在默认行为变更与派生默认值的验证深度上, 合并前建议补齐非整型 `auto_stop_interval_mins` 的测试。

功能与动机

PR body 明确指出问题根源: Harbor 的 Daytona 默认值从不回收被杀 worker 遗留的沙箱 (`auto_stop/auto_delete` default to 0 = never), 导致资源泄漏。设计约束是 stop 间隔必须长于 trial 上限, 因为“an in-sandbox agent looks idle to Daytona”——沙箱内 agent 对 Daytona 表现为空闲, 过短的 stop 间隔会误杀活跃 trial。同时 `HARBOR_OVERRIDE_STORAGE_MB` 是 Daytona 磁盘配额计费的关键量, 需要暴露给用户配置。

实现拆解

1. 提取 trial 超时读取函数: 在 `examples/experimental/harbor/harbor_agent_function.py` 中新增模块级函数 `_trial_timeout_s()`, 把 `run()` 里内联的 `int(os.environ.get("AGENT_TRIAL_TIMEOUT", ...))` 上移为统一入口, 使 `run()` 与 `_environment_config()` 共用同一读取逻辑, 避免两处解析漂移。
2. 注入 Daytona 回收默认值: 在 `_environment_config()` 中, 当 `HARBOR_ENV_TYPE` 解析为 `EnvironmentType.DAYTONA` 时, 用 `kwargs.setdefault()` 设置 `auto_stop_interval_mins = ceil(AGENT_TRIAL_TIMEOUT / 60) + 30`、`auto_delete_interval_mins = 1440`; 若调用者通过 `HARBOR_ENV_KWARGS` 显式传入的 stop 间隔小于等于 trial 分钟上限, 则抛出 `ValueError` 拒绝该配置。
3. 统一资源 override 处理: 将 `HARBOR_OVERRIDE_MEMORY_MB` 与新增的 `HARBOR_OVERRIDE_STORAGE_MB` 放入同一循环, 分别映射为 `EnvironmentConfig.override_memory_mb` 与 `override_storage_mb`, 非正整数值统一拒绝, 内存与磁盘配额语义保持一致。
4. 配套接线与文档: `examples/experimental/harbor/launch_common.py` 的 `harbor_env_vars()` 转发列表新增 `HARBOR_OVERRIDE_STORAGE_MB`;

docs/user-guide/sandbox-providers.md 更新 Daytona 支持矩阵 (in-process 路径可直接使用, 移除对旧 recipe 目录的链接) 并补充配额配置说明。

5. 测试配套: 将原 `test_environment_type_is_passed_straight_through` 改用 `modal` 验证 (daytona 已不再直通); 新增 `test_daytona_reclaim_timer_outlasts_the_trial_cap` (覆盖派生默认值、调用者安全值优先、过小值拒绝三种情形)、`test_reclaim_timers_are_daytona_only` (e2b 不受影响)、`test_resource_overrides_reach_harbor_and_reject_nonpositive` (参数化验证内存 / 存储 `override` 传递与非正值拒绝)。

关键文件:

- `examples/experimental/harbor/harbor_agent_function.py` (模块 Harbor 代理; 类别 `source`; 类型 `core-logic`; 符号 `_trial_timeout_s`, `_environment_config`, `run`): 核心变更文件: 新增 `_trial_timeout_s()` 统一超时读取; 在 `_environment_config()` 中为 Daytona 注入 `auto_stop/auto_delete` 默认值并拒绝可能误停 `trial` 的调用者配置; 新增 `HARBOR_OVERRIDE_STORAGE_MB` 到 `override_storage_mb` 的映射, 是资源回收与配额配置的落地处。
- `tests/fast/examples/experimental/harbor/test_harbor_agent_function.py` (模块 Harbor 测试; 类别 `test`; 类型 `test-coverage`; 符号 `test_daytona_reclaim_timer_outlasts_the_trial_cap`, `test_reclaim_timers_are_daytona_only`, `test_resource_overrides_reach_harbor_and_reject_nonpositive`): 测试配套核心: 新增 3 个测试覆盖派生回收默认值、调用者安全值优先、过小值拒绝、daytona-only 隔离以及内存 / 存储 `override` 传递与非正值拒绝, 并将原直通测试改用 `modal` 以适配 daytona 不再直通的新语义。
- `examples/experimental/harbor/launch_common.py` (模块 启动配置; 类别 `source`; 类型 `configuration`; 符号 `harbor_env_vars`): 在 `harbor_env_vars()` 的转发变量列表中新增 `HARBOR_OVERRIDE_STORAGE_MB`, 使启动器能把磁盘配额配置传递到 rollout worker, 是新增配置能力的接线点。
- `docs/user-guide/sandbox-providers.md` (模块 用户文档; 类别 `docs`; 类型 `documentation`): 同步更新 Daytona 支持矩阵 (in-process Harbor 路径已支持, 移除对旧 recipe 目录的链接) 并补充 `HARBOR_OVERRIDE_STORAGE_MB` 配额配置说明。

关键符号: `_trial_timeout_s`, `_environment_config`, `run`, `harbor_env_vars`

关键源码片段

`examples/experimental/harbor/harbor_agent_function.py`

核心变更文件: 新增 `_trial_timeout_s()` 统一超时读取; 在 `_environment_config()` 中为 Daytona 注入 `auto_stop/auto_delete` 默认值并拒绝可能误停 `trial` 的调用者配置; 新增 `HARBOR_OVERRIDE_STORAGE_MB` 到 `override_storage_mb` 的映射, 是资源回收与配额配置的落地处。

```
def _trial_timeout_s() -> int:
    """run() 施加在单个 trial 上的墙钟上限, 即一个 trial 最长可存活时间。"""
    return int(os.environ.get("AGENT_TRIAL_TIMEOUT", _DEFAULT_AGENT_TRIAL_TIMEOUT_S))

def _environment_config():
```

```

"""把 HARBOR_ENV_TYPE 直通为 Harbor 的 EnvironmentType; 后端特定参数
通过 HARBOR_ENV_KWARGS (JSON 对象) 进入 EnvironmentConfig.kwargs. """
from harbor.models.environment_type import EnvironmentType
from harbor.models.trial.config import EnvironmentConfig

raw = os.getenv("HARBOR_ENV_TYPE", "").strip().lower()
if not raw:
    raise ValueError("set HARBOR_ENV_TYPE to a Harbor environment type (e.g. e2b,
    daytona)")
env_type = EnvironmentType(raw) # 未知后端直接抛错, 避免静默回退到 docker
kwargs = json.loads(os.getenv("HARBOR_ENV_KWARGS", "{}") or "{}")

if env_type == EnvironmentType.DAYTONA:
    # 被 kill 的 worker 不会走到 Harbor 的 teardown, 而 Harbor 的
    # Daytona 默认 auto_stop/auto_delete 为 0 (永不回收), 会永久
    # 占用配额; stop 间隔必须长于 trial 上限, 因为沙箱内 agent 对
    # Daytona 表现为空闲, 过短的 stop 会误杀活跃 trial。
    trial_min = math.ceil(_trial_timeout_s() / 60)
    kwargs.setdefault("auto_stop_interval_mins", trial_min + 30)
    kwargs.setdefault("auto_delete_interval_mins", 1440) # 24 小时后强制回收孤儿沙箱
    if int(kwargs["auto_stop_interval_mins"]) <= trial_min:
        raise ValueError(
            f"auto_stop_interval_mins={kwargs['auto_stop_interval_mins']} would stop "
            f"a Daytona sandbox mid-trial: AGENT_TRIAL_TIMEOUT allows {trial_min} minutes"
        )

# 内存与磁盘配额统一走 override, 非正值属于配置错误, 直接拒绝
overrides = {}
for field, var in (
    ("override_memory_mb", "HARBOR_OVERRIDE_MEMORY_MB"),
    ("override_storage_mb", "HARBOR_OVERRIDE_STORAGE_MB"),
):
    value = _env_int(var)
    if value is not None and value <= 0:
        raise ValueError(f"{var} must be a positive integer")
    overrides[field] = value
return EnvironmentConfig(type=env_type, delete=True, kwargs=kwargs, **overrides)

```

tests/fast/examples/experimental/harbor/test_harbor_agent_function.py

测试配套核心: 新增 3 个测试覆盖派生回收默认值、调用者安全值优先、过小值拒绝、daytona-only 隔离以及内存 / 存储 override 传递与非正值拒绝, 并将原直通测试改用 modal 以适配 daytona 不再直通的新语义。

```

def test_daytona_reclaim_timer_outlasts_the_trial_cap(tasks_dir, monkeypatch):
    """Harbor 的 Daytona 默认值从不回收被杀 worker 遗留的沙箱;
    本函数必须回收, 但又不能误停活着的 trial. """
    monkeypatch.setenv("HARBOR_ENV_TYPE", "daytona")
    monkeypatch.setenv("AGENT_TRIAL_TIMEOUT", "1200") # 20 分钟
    cfg = haf.build_trial_config({"instance_id": "task-1", "agent_name": "mini-swe-agent"}, "http://s/

```

```

v1", {}))
# 20 分钟上限 + 30 分钟缓冲 => auto_stop 50 分钟; delete 默认 24 小时
assert cfg.environment.kwargs == {"auto_stop_interval_mins": 50, "auto_delete_interval_mins":
1440}

# 调用者显式传入的安全值 (45 > 20) 优先于派生默认值
monkeypatch.setenv("HARBOR_ENV_KWARGS", '{"auto_stop_interval_mins": 45, "auto_
snapshot": true}')
cfg = haf.build_trial_config({"instance_id": "task-1", "agent_name": "mini-swe-agent"}, "http://s/
v1", {})
assert cfg.environment.kwargs == {
    "auto_stop_interval_mins": 45,
    "auto_delete_interval_mins": 1440,
    "auto_snapshot": True,
}

# 小于等于 trial 上限的值会导致沙箱在 trial 途中被停止, 必须拒绝
monkeypatch.setenv("HARBOR_ENV_KWARGS", '{"auto_stop_interval_mins": 20}')
with pytest.raises(ValueError, match="mid-trial"):
    haf.build_trial_config({"instance_id": "task-1", "agent_name": "mini-swe-agent"}, "http://s/
v1", {})

```

评论区精华

本 PR 没有实质性的技术争论, review 活动集中在自动审查与人工批准:

- claude[bot] 首次评论提示该仓库配置为手动 review; 第二次评论报告 "Code review found no issues", 但明确注明只覆盖了不再是最新的 commit d033e29, 后续 commit 未被自动审查覆盖。
- 维护者 Shi-Dong 给出 "LGTM!" 并 APPROVED, 完成人工把关。
- PR body 自带设计说明: stop 间隔取 trial 上限加 30 分钟缓冲, 因为 "in-sandbox agent looks idle to Daytona"; 派生默认值 (timeout + 30) 仅由单元测试覆盖, 真实 e2e 验证使用的是显式 timer (60/120 分钟)。
- 自动审查覆盖范围与人工批准 (other): 无代码问题被提出, 合并由人工批准完成; 自动审查未覆盖最终 commit 的风险被人工审查弥补。

风险与影响

- 风险:
 1. Daytona 路径默认行为变更: 所有 HARBOR_ENV_TYPE=daytona 的 trial 现在都会自动带上 auto_stop_interval_mins 与 auto_delete_interval_mins, 若调用者此前依赖“永不停止”的默认行为, 运行语义会改变; 正常完成的 trial 不受影响, 但这是无开关的行为变更。
 2. 非整型入参校验缺口: int(kwargs["auto_stop_interval_mins"]) 若收到无法解析的字符串 (如 "abc"), 会抛出未带自定义说明的 ValueError, 现有测试未覆盖该分支; 配置错误会表现为每次 trial 失败, 虽然这符合“loud stop”的设计哲学, 但错误信息不够明确。

3. 派生默认值缺少真实环境验证: PR body 承认 derived default 仅单元测试覆盖, 真实 e2e 用的是显式 timer; 若 AGENT_TRIAL_TIMEOUT 的环境解析或 ceil 计算与 Harbor 侧字段单位不一致, 单元测试可能只验证了自身逻辑。
 4. 环境变量解析失败会静默回退: `_trial_timeout_s()` 对非法值直接抛 `ValueError (int() 失败)`, 而原来在 `run()` 中同样如此, 行为未改变, 但新增的 daytona 分支使该失败会发生在配置构建阶段而非运行阶段。
- 影响: 影响范围集中在 `examples/experimental/harbor` 的 in-process 路径:
 - 对用户: Daytona 沙箱配额不再被 killed worker 遗留的孤儿沙箱长期占用, 同时获得 `HARBOR_OVERRIDE_STORAGE_MB` 磁盘配额配置能力; 需要显式覆盖回收策略的用户可通过 `HARBOR_ENV_KWARGS` 传入安全值。
 - 对系统: Harbor trial 的 `EnvironmentConfig` 构造逻辑从“直通”变为“按后端注入默认值”, 只对 `EnvironmentType.DAYTONA` 分支生效, `e2b/modal` 等其他后端行为不变。
 - 对团队: 该示例路径的配置契约更新 (新增环境变量、默认回收行为), 使用 daytona rollout 的脚本无需改动即可受益; `agent-server` 的 Daytona recipe 保持不变。
 - 风险标记: Daytona 路径默认行为变更, 派生默认值仅单元测试覆盖, 非整型 `HARBOR_ENV_KWARGS` 校验缺口, 自动审查未覆盖最终 commit

关联脉络

- PR #2838 Harbor Daytona 配置 (已关闭; 本 PR 承接其回收与存储配置): PR body 明确说明本 PR 的 `auto_stop/auto_delete` 默认值与 `HARBOR_OVERRIDE_STORAGE_MB` 均 "carried over from #2838"; #2838 关闭后未落地的 Daytona in-process 配置在本 PR 完成。
- PR #3195 test(agentica): run the sandbox tests on any provider, not just e2b: 改动了 `miles/rollout/agentica/credentials.py` 的 `PROVIDER_CREDENTIALS` 注册表, 本 PR 的 `_ensure_provider_key()` 依赖该注册表解析 Daytona 凭据, 同属 agentica 沙箱基础设施演进。
- PR #2837 docs: the in-process Harbor example: 为用户指南补充 in-process Harbor 模式; 本 PR 的 Daytona 支持矩阵更新延续了该路径的说明。
- PR #2876 ci(e2e): Harbor trials on real e2b sandboxes through the full rollout path (manual for now): Harbor 真实沙箱端到端测试的开端, 与本 PR 的 Harbor 资源配置变更同属 Harbor 支持体系。