ChatGPT

Мінімальний continuity layer для top-level репозиторію obw_platform

Executive summary

Висновок простий: **continuity layer тут уже не “nice to have”, а дешева страховка від втрати контексту**.

По наданому архіву видно типовий перекіс довгоживучого trading-repo: багато варіантів бектестерів і раннерів, кілька UI/API поверхонь, десятки YAML-конфігів, тестові сценарії з абсолютними шляхами `/mnt/data/...`, порожні або відсутні data-artifacts, і немає одного короткого місця, де зафіксовано: **що є канонічним, що експериментальним, який контракт між backtest/live, які файли справді головні, і що не можна ламати випадковим Codex-завданням**.

Правильний синтез із `agent-continuity-kit` для цього репозиторію такий:

- додати **тільки тонкий top-level шар** `continuity/`;
- **не тягнути в перший PR** `examples/` і `templates/` із kit;
- створити **вісім line-файлів** під реальні живі осі repo;
- зробити перший PR **docs-only + lightweight hook**, без змін торгового коду;
- використати continuity не як архів сесій, а як **операційний індекс канонічних шляхів, контрактів, ризиків drift і найменшого наступного кроку**.

Ключова жорстка рекомендація: **перший continuity PR має бути максимально вузьким**. Якщо ти в той самий PR пхаєш refactor раннера, зміну YAML, нову UI-сторінку і ще “трохи почистив backtester” — ти сам вбиваєш сенс цього шару.

Що видно з kit і з наданого архіву

`agent-continuity-kit` корисний не кодом, а дисципліною: він наполягає на тонкому шарі орієнтації, де після значущої роботи лишається не стенограма, а короткий “осад”: які лінії живі, що змінилося, де drift, і який наступний найменший корисний крок. Саме це тут і потрібно.

За наданим архівом репозиторій уже переріс режим “README достатньо”. У ньому є багато паралельних осей розвитку: backtest, dual-core, live/paper runner, data cache builders, validation UI, research/tuning, deploy UI, multi-agent task files. Це не маленький проект, це вже система з конкуренцією між підсистемами.

Ознаки, що continuity потрібен вже зараз

Спостереження в архіві	Чому це проблема	Яка continuity line повинна це тримати
Багато `backtester_*` і `live_runner_*` варіантів	незрозуміло, що канонічне, а що черговий експеримент	`trading-core`, `live-runner`
Багато YAML-конфігів із різними `cache_db`, у тому числі абсолютними `/mnt/data/...` і закоментованими альтернативами	конфіги стають не джерелом істини, а кладовищем шляхів	`data-cache`, `backtest-integrity`
Кілька UI/API поверхонь: `UI/backend/api_main.py`, `api/main.py`, `v/api/main.py`, `api_main.py`-alias	UI-admin межі розмиті; легко зробити фічу не в тому шарі	`ui-admin`
Частина тестів падає не через бізнес-логіку, а через import/path drift	це чистий симптом відсутності описаного контракту між файлами й середовищем	`trading-core`, `live-runner`, `codex-workflow`
У наданому архіві реальні cache DB/NPZ майже відсутні або пусті	бектест-команди виглядають готовими, але не відтворюються	`data-cache`, `backtest-integrity`
Є `multi_agent/task_tradebot_*.md`, але нема нормованого Codex spec	AI-воркеру нема формальних меж: що можна міняти, що ні	`codex-workflow`

Що реально вдалося перевірити на наданому архіві

Нижче — те, що було фактично прогнано локально на наданому snapshot, а не вигадано.

Команда	Результат
`pytest --collect-only -q`	зібрано 127 тестів, але 5 collection errors
`pytest -q UI/backend/tests/test_backtest_live_validation.py UI/backend/tests/test_infer_config.py UI/backend/tests/test_progress.py`	**11 passed**
`pytest -q tests/test_pine_bridge.py obw_platform/tests/test_breakout_avaai_full_with_universe_2.py obw_platform/tests/test_breakout_avaai_full_with_universe_4.py obw_platform/tests/test_breakout_avaai_full_with_universe_5.py`	**21 passed**

Команда	Результат
`pytest -q tests/test_maker_dca_flow.py -k 'not session_sqlite_real_scenario_extracts_open_close_open_sequence'`	**3 passed, 1 deselected**
`pytest -q obw_platform/tests/test_backtester_core_speed3_veto_universe2_configs.py -k 'cs_C2_base_1h or greedy_breakout_universe or cache_db_cli_override or symbols_file_path_with_prefix'`	**3 passed, 1 skipped**
`pytest -q tests/test_limit_dca_virtual.py`	fail через `VirtualExchange.__init__()` і аргумент `starting_cash`
`PYTHONPATH=obw_platform/runners pytest -q obw_platform/tests/test_live_debug_bundle.py`	**7 passed**
`bash obw_platform/run_c2_1h_1440.sh`	fail: `RuntimeError: No tables in cache DB`

Ілюстративний ASCII-чарт поточного стану придатності тестів

```
Поточна придатність suite на наданому snapshot

UI/backend smoke              [###########] 11/11 pass
Strategy logic subset         [####################] 21/21 pass
Maker/DCA subset              [###-] 3 pass, 1 deselected
Config smoke subset           [###s] 3 pass, 1 skip
Full collection               [127 collected | 5 collection errors]
Legacy runner/path tests      [xx] broken by path/interface drift
Repro backtest shell script   [x] blocked by empty cache DB
```

Це головний сигнал: **repo уже потребує не ще одного README, а маленького шару канонізації**.

Цільова структура continuity

Що саме створювати

Перший continuity-патч повинен додати **тільки** це:

Шлях	Призначення
`continuity/AGENTS.md`	правила роботи з continuity у repo

Шлях	Призначення
`continuity/MAP.md`	коротка карта живих ліній за стадіями
`continuity/lines/trading-core.md`	канонічні backtest engines і strategy contract
`continuity/lines/backtest-integrity.md`	довіра до бектесту, parity з live/validation
`continuity/lines/live-runner.md`	канонічний runtime, prewarm, session artifacts
`continuity/lines/strategy-research.md`	research/tuning/search pipeline
`continuity/lines/data-cache.md`	cache DB/NPZ naming, provenance, required data
`continuity/lines/ui-admin.md`	primary UI/API surface та її межі
`continuity/lines/codex-workflow.md`	стандарт для AI/Codex task spec
`continuity/lines/investor-narrative.md`	короткий narrative, прив'язаний до метрик
`continuity/check.sh`	мінімальний shell hook для валідації continuity
`.github/workflows/continuity.yml`	опційно: вузький CI-хук під continuity

Чого не треба тягнути в перший PR

Не копіюй у PR1:

- `examples/`
- `templates/`
- будь-які session summaries
- будь-які generated file trees
- будь-які звіти з бектестів усередину `continuity/`

Причина проста: якщо continuity почне рости як ще один knowledge base, він програє власну місію.

Пропонований `MAP.md` під ваш trading repo

```
# MAP

Це коротка карта живих ліній у репозиторії.

Вона не є архівом, дампом чи журналом сесій.
Вона показує лінії за стадіями дозрівання:

- `idea`
- `seed`
- `sprout`
```

```
- `plant`
- `fruit`

Карта має бути коротшою за територію.
Якщо вона швидко росте — вона вже не карта.

## idea

## seed

- `investor-narrative` - коротка інвесторська логіка: звідки очікувана перевага, який реальний ризик, які межі доказовості. Line: `lines/investor-narrative.md`
- `codex-workflow` - стандарт для AI/Codex задач: вузький scope, заборонені файли, тест-команда, acceptance, report back. Line: `lines/codex-workflow.md`

## sprout

- `strategy-research` - пошук і підтвердження робочих стратегій, tuner plans, out-of-sample логіка, monthly/quarter discipline. Line: `lines/strategy-research.md`
- `backtest-integrity` - довіра до бектесту, parity між backtest, validation UI, paper/live та експортами біржі. Line: `lines/backtest-integrity.md`

## plant

- `trading-core` - канонічні backtest engines і shared strategy interface для single-leg і dual-leg сімейств. Line: `lines/trading-core.md`
- `data-cache` - канонічні DB/NPZ артефакти, provenance, naming, побудова й повторне використання кешів. Line: `lines/data-cache.md`
- `live-runner` - live/paper runtime, prewarm, session.sqlite, combined_cache_session.db, debug bundle і close reasons. Line: `lines/live-runner.md`
- `ui-admin` - основна операторська UI/API поверхня, конфіг-редактор, validation pages, live session inspection. Line: `lines/ui-admin.md`

## fruit
```

Мінімальні правила для `AGENTS.md`

У `continuity/AGENTS.md` не потрібна філософія на дві сторінки. Потрібні короткі правила:

- continuity — це **не архів сесій**;
- перед роботою агент читає `AGENTS.md`, `MAP.md` і **тільки релевантний** line-файл;
- після значущої зміни агент оновлює **одну релевантну лінію**, а не пише звіт за все життя;
- якщо створено новий `*_vN.py`, line-файл має сказати, **це canonical чи experiment**;
- якщо змінено canonical path/contract/data requirement — continuity оновлюється **в тому ж PR**.

Специфікація восьми line-файлів

Нижче — не “ідеї”, а стартовий набір, який можна комітити майже без переробки.

Зведена карта ліній

Файл	Stage	Призначення	Ключові drift risks	Acceptance criteria	Next smallest useful move
`continuity/lines/trading-core.md`	plant	Зафіксувати канонічні engines і strategy API	`*_vN.py` сприймаються як canonical без рішення; single-leg і dual-leg розходяться по метриках і інтерфейсу	названо canonical single-leg engine, canonical dual-leg engine, shared strategy methods	оголосити один canonical файл на сімейство
`continuity/lines/backtest-integrity.md`	sprout	Тримати довіру до симуляції	PnL-оптимізація без parity; плутанина MTM vs realized; відсутність fees/slippage/funding/warmup	зафіксовано метрики, validation matrix, required outputs	затвердити мінімальну матрицю 1 місяць / квартал / рік-by-month
`continuity/lines/live-runner.md`	plant	Канонічний runtime і live artifacts	прихована логіка раннера; відсутній prewarm; слабкий post-mortem	названо primary engine, wrapper, session artifacts, close reasons	проголосити один runtime canonical
`continuity/lines/strategy-research.md`	sprout	Дисципліна tuner/search pipeline	cherry-picking; неясно, який результат champion; research не відділений від prod	зафіксовано active question, canonical tuner driver, minimum OOS protocol	номінувати один smoke-plan і один active plan

Файл	Stage	Призначення	Ключові drift risks	Acceptance criteria	Next smallest useful move
`continuity/lines/data-cache.md`	plant	Джерела і формат даних	0-byte DB, абсолютні `/mnt/data`, DB/NPZ хаос, невідоме provenance	naming schema, builders, required metadata, exact cache families визначені	описати canonical cache families і stop using comment-soup in YAML
`continuity/lines/ui-admin.md`	plant	Канонічна UI/API поверхня	кілька backend/frontend осей, незрозуміло що primary	названо primary backend/frontend, legacy surfaces, supported pages/tests	зафіксувати primary admin surface
`continuity/lines/codex-workflow.md`	seed	Нормалізувати AI worker tasks	широкі задачі, edited too much, no forbidden files, no test command	є стандарт Goal/Scope/Do not edit/Files/Test/Acceptance/Report back	конвертувати 1–2 існуючі task files у новий шаблон
`continuity/lines/investor-narrative.md`	seed	Короткий narrative, прив'язаний до фактичних метрик	hype без evidence; обіцянки замість risk language	є 200–300 слів з edge/risk/evidence boundaries	написати 1 версію без маркетингового шуму

Стартовий markdown для кожної лінії

`continuity/lines/trading-core.md`

```
# trading-core

## Stage

plant

## What This Line Is

Лінія про канонічне торгове ядро проекту: які backtest engines вважаються основними, який strategy API є контрактом, і де закінчується експериментальна гілка.

## Current Shape
```

```
- Single-leg canonical candidate: `obw_platform/
backtester_core_speed3_veto_universe_2.py`
- Dual-leg canonical candidate: `obw_platform/
backtester_dual_core_dynamic_v6.py`
- Shared strategy methods observed in code: `entry_signal`,
`manage_position`, `sync_after_external_fill`
- Optional live parity methods: `warmup_requirements`, `warmup_history`,
`export_state_snapshot`, `restore_state_snapshot`, `on_order_rejected`

## What Changed Recently

Continuity introduced to stop treating every `*_vN.py` file as implicitly
canonical.

## Drift Risks

- New engine variants appear without explicit canonical decision.
- Single-leg and dual-leg metrics diverge.
- Strategy API changes in runner but not in backtester.
- Config examples point to engines that are no longer the real default.

## Next Useful Move

Name one canonical single-leg engine and one canonical dual-leg engine in
this file and treat all others as legacy or experimental until promoted.
```

`continuity/lines/backtest-integrity.md`

```
# backtest-integrity

## Stage

sprout

## What This Line Is

Лінія про те, чи можна довіряти equity curve. Не "красивий графік", а
відповідність між бектестом, validation UI, paper/live runner і біржовими
результатами.

## Current Shape

- Validation-related files: `UI/backend/api_main.py`, `UI/frontend/pages/
backtest_live_validation.tsx`
- Matching/export scripts: `obw_platform/extract_bingx_window_from_tv.py`,
`obw_platform/match_trades_tv_bingx.py`, `obw_platform/
bingx_export_futures_trade_history.py`
-
Metrics that must not be mixed: realized PnL, MTM equity, realized MDD, MTM
```

```
MDD, fees, slippage, funding

## What Changed Recently

Testing on the archive shows that code subsets run, but full reproducibility
is blocked by path drift and missing data artifacts.

## Drift Risks

- Optimizing PnL while ignoring execution realism.
- Using realized and MTM numbers interchangeably.
- Running validation only on one favorable slice.
- Comparing backtest to live without aligned fills and timestamps.

## Next Useful Move

Freeze a minimal validation matrix and required output bundle for every
meaningful strategy change.
```

`continuity/lines/live-runner.md`

```
# live-runner

## Stage

plant

## What This Line Is

Лінія про canonical live/paper runtime: який engine виконує стратегію, як він
prewarm-иться, що логує, і які artifacts вважаються стандартними для post-
mortem.

## Current Shape

- Runtime engine candidate: `obw_platform/runners/live_runner_dual.py`
- CLI wrapper family: `obw_platform/
bt_live_paper_runner_separated_universe*.py`
- Expected artifacts: `session.sqlite`, `combined_cache_session.db`,
`status.json`, `equity.csv`, `orders.csv`, `debug_events.jsonl`, `stdio.log`
- Prewarm hooks observed: `warmup_requirements`, `warmup_history`, `--
prewarm-bars`, `--prewarm-hours`

## What Changed Recently

Archive inspection shows strong warmup/prewarm logic, but also runner version
sprawl.

## Drift Risks
```

```
- Hidden live logic absent from backtester.
- Noisy wrapper becomes de facto engine.
- Missing prewarm creates blind first decisions.
- Debug artifacts are inconsistent across runs.

## Next Useful Move

Declare one runtime engine as canonical and list which wrapper file is
allowed to launch it.
```

`continuity/lines/strategy-research.md`

```
# strategy-research

## Stage

sprout

## What This Line Is

Лінія про пошук і підтвердження робочих стратегій: tuner drivers, plan files,
out-of-sample discipline, champion configs і research questions.

## Current Shape

- Tuner drivers observed: `obw_platform/auto_tuner_rays2grid_v3_fix.py`,
`obw_platform/auto_universe_and_tune.py`
- Research helpers observed: `obw_platform/monthly_akela_phase_proxybt.py`,
`obw_platform/rank_short_leg_all_symbols_akela_v2.py`
- Search spaces live in: `obw_platform/tuner_plans/` and `obw_platform/
tuner_plan_*.py`

## What Changed Recently

Research machinery exists, but the archive does not define one active
champion path clearly enough for agents.

## Drift Risks

- Tuning on one lucky window.
- No distinction between experiment and promoted config.
- Too many plan files with no active status.
- Research notes live only in chat and die there.

## Next Useful Move

Record one active research question, one champion config, and one smoke plan
in this line.
```

`continuity/lines/data-cache.md`

```
# data-cache

## Stage

plant

## What This Line Is

Лінія про канонічні data artifacts: SQLite caches, NPZ packs, live session
caches, naming conventions, provenance і minimum metadata.

## Current Shape

- Builders observed: `fetch_build_cache_v16.py`, `fetch_build_cache_v17.py`,
`obw_platform/build_fast_ohlcv_npz_from_db_v2.py`, `obw_platform/
build_fast_multi_npz_from_db.py`
- Live session cache artifact: `combined_cache_session.db`
- Problem observed in archive: `obw_platform/combined_cache.db` is present
but empty

## What Changed Recently

The codebase expects many cache families, but the provided snapshot lacks
production-grade sample data.

## Drift Risks

- Absolute `/mnt/data/...` paths leak into configs and tests.
- YAML files keep commented-out path graveyards.
- DB and NPZ families are mixed without provenance.
- Agents cannot tell which cache is authoritative.

## Next Useful Move

Define canonical naming and one authoritative cache example per active
timeframe family.
```

`continuity/lines/ui-admin.md`

```
# ui-admin

## Stage

plant

## What This Line Is

Лінія про операторську UI/API поверхню: який backend і frontend є primary,
які сторінки підтримуються, і які поверхні вважаються legacy.
```

```
## Current Shape

- Primary backend candidate: `UI/backend/api_main.py`
- Primary frontend candidate: `UI/frontend/`
- Additional API surfaces observed: `api/main.py`, `v/api/main.py`, root
`api_main.py`
- Important pages observed: `pages/backtest_live_validation.tsx`, `pages/
live_result.tsx`, `pages/runs.tsx`, `pages/deploy.tsx`

## What Changed Recently

UI/backend tests on the provided archive pass, but API surface is clearly
split across multiple trees.

## Drift Risks

- Feature added to the wrong backend.
- Legacy API starts acting as primary.
- Validation UI and live session UI drift apart.
- Deploy/admin concerns mix with strategy validation concerns.

## Next Useful Move

Write down one primary admin surface and classify all other UI/API paths as
legacy, experimental or adapter-only.
```

`continuity/lines/codex-workflow.md`

```
# codex-workflow

## Stage

seed

## What This Line Is

Лінія про стандарт AI/Codex задач у цьому repo: вузький scope, заборонені
файли, тест-команда, acceptance і короткий report back.

## Current Shape

- Existing task-like files observed: `multi_agent/task_tradebot_brief.md`,
`multi_agent/task_tradebot_full.md`
- No normalized Codex spec found in the archive
- Repo needs stricter file-boundary discipline between `continuity/`, `UI/`,
`api/`, `v/` and `obw_platform/`

## What Changed Recently
```

```
Continuity layer is being added specifically to reduce agent drift and
unnecessary repo-wide edits.

## Drift Risks

- One task edits multiple system layers.
- No forbidden-files section.
- No explicit test command.
- No definition of done.
- No mandatory status report.

## Next Useful Move

Adopt one short Codex task template and require it for every external AI
task.
```

`continuity/lines/investor-narrative.md`

```
# investor-narrative

## Stage

seed

## What This Line Is

Лінія про коротке пояснення проекту зовнішній людині: де джерело очікуваної
переваги, як ми контролюємо ризик, і де закінчується evidence.

## Current Shape

- Narrative must refer to real metric classes, not isolated banners
- It must distinguish backtest evidence, validation evidence and live
evidence
- It must not promise returns or hide drawdown mechanics

## What Changed Recently

The repo now needs a thin continuity layer because internal complexity
already exceeds what a clean pitch can safely compress.

## Drift Risks

- Hype replaces evidence.
- Backtest champion is pitched as live truth.
- Risk is hidden behind one profitable chart.
- Strategy families are mixed into one story.

## Next Useful Move
```

```
Draft a 200–300 word version tied only to canonical metrics and validation
states.
```

План patch/PR, контроль якості і команди

Для першого PR бери **окрему робочу гілку** і відкривай PR тільки з неї. `git switch -c <new-branch>` створює й одразу перемикає на нову гілку; branch-based PR workflow саме для цього й існує — ізолювати незавершену роботу від default branch і порівнювати topic branch проти base branch. Якщо репозиторій живе в GitHub [1], це природний і правильний шлях. [2]

Scope першого continuity PR

Дозволено	Заборонено
`continuity/**`	будь-які зміни в `obw_platform/**`
`continuity/check.sh`	будь-які зміни в `UI/**`
`.github/workflows/continuity.yml`	будь-які зміни в `api/**`, `v/**`
за потреби лише README-посилання на continuity	refactor торгового коду, YAML, стратегій, runner logic

Рекомендовані branch і PR назви

Тип	Назва
Branch	`chore/continuity-layer-minimal`
PR title	`chore: add minimal continuity layer`
Optional follow-up branch	`chore/continuity-hook`

Покроковий patch-план

```
# 0. В робочому checkout реального репозиторію
git switch -c chore/continuity-layer-minimal

# 1. Створити структуру
mkdir -p continuity/lines .github/workflows

# 2. Створити файли
touch continuity/AGENTS.md
touch continuity/MAP.md
touch continuity/lines/trading-core.md
touch continuity/lines/backtest-integrity.md
touch continuity/lines/live-runner.md
touch continuity/lines/strategy-research.md
touch continuity/lines/data-cache.md
touch continuity/lines/ui-admin.md
touch continuity/lines/codex-workflow.md
```

```
touch continuity/lines/investor-narrative.md
touch continuity/check.sh
chmod +x continuity/check.sh
```

Потім вставляєш вміст із цього звіту. Після цього додаєш мінімальний shell check.

Мінімальний `continuity/check.sh`

```
#!/usr/bin/env bash
set -euo pipefail

required=(
  continuity/AGENTS.md
  continuity/MAP.md
  continuity/lines/trading-core.md
  continuity/lines/backtest-integrity.md
  continuity/lines/live-runner.md
  continuity/lines/strategy-research.md
  continuity/lines/data-cache.md
  continuity/lines/ui-admin.md
  continuity/lines/codex-workflow.md
  continuity/lines/investor-narrative.md
)

for f in "${required[@]}"; do
  test -f "$f" || { echo "missing: $f"; exit 1; }
done

for f in continuity/lines/*.md; do
  grep -q '^# ' "$f"
  grep -q '^## Stage' "$f"
  grep -q '^## What This Line Is' "$f"
  grep -q '^## Current Shape' "$f"
  grep -q '^## What Changed Recently' "$f"
  grep -q '^## Drift Risks' "$f"
  grep -q '^## Next Useful Move' "$f"
done

python3 - <<'PY'
from pathlib import Path
import re

map_path = Path("continuity/MAP.md")
text = map_path.read_text(encoding="utf-8")
refs = re.findall(r'`lines/([^`]+\.md)`', text)
missing = [r for r in refs if not Path("continuity/lines",
Path(r).name).exists()]
if missing:
    raise SystemExit(f"MAP references missing files: {missing}")
```

```
print("continuity check: OK")
PY
```

Мінімальний CI hook

```
name: continuity

on:
  pull_request:
  push:
    branches:
      - main
      - master

jobs:
  continuity:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - run: bash continuity/check.sh
```

Git workflow для PR

```
# 3. Локальна перевірка continuity
bash continuity/check.sh

# 4. Додаткові cheap smoke tests, які не лізуть у trading code
pytest -q UI/backend/tests/test_backtest_live_validation.py \
         UI/backend/tests/test_infer_config.py \
         UI/backend/tests/test_progress.py

pytest -q obw_platform/tests/
test_backtester_core_speed3_veto_universe2_configs.py \
       -k
'cs_C2_base_1h or greedy_breakout_universe or cache_db_cli_override or
symbols_file_path_with_prefix'

# 5. Коміт
git add continuity .github/workflows/continuity.yml
git commit -m "chore: add minimal continuity layer"

# 6. Push + PR
git push -u origin chore/continuity-layer-minimal
```

Які test hooks ставити в PR1

Не став важких gate'ів у перший continuity PR. Потрібен **cheap, deterministic gate**:

1. `bash continuity/check.sh`
2. `pytest -q UI/backend/tests/test_backtest_live_validation.py UI/backend/tests/test_infer_config.py UI/backend/tests/test_progress.py`
3. `pytest -q obw_platform/tests/test_backtester_core_speed3_veto_universe2_configs.py -k 'cs_C2_base_1h or greedy_breakout_universe or cache_db_cli_override or symbols_file_path_with_prefix'`

Все. Не роби в PR1 повний suite mandatory, бо він уже зараз має collection/path/data drift.

Команди для запуску тестів стратегій і що вони реально означають

Для огляду suite без виконання використовуй `pytest --collect-only`; для локального звуження — `-k`, а для тимчасового карантину крихких файлів — `--ignore`. Це базові офіційні можливості CLI `pytest`. Якщо у середовищі `pytest` ще не поставлений, стандартна швидка інсталяція — `pip install -U pytest`. [3]

Мінімальні команди, які варто запускати локально вже зараз

```
# Базове середовище
python3 -m venv .venv
source .venv/bin/activate
pip install --upgrade pip
pip install -r requirements.txt
pip install -r UI/backend/requirements.txt
pip install -U pytest
```

```
# Подивитися, що suite взагалі збирає
pytest --collect-only -q
```

```
# Green subset по strategy logic
pytest -q tests/test_pine_bridge.py \
         obw_platform/tests/test_breakout_avaai_full_with_universe_2.py \
         obw_platform/tests/test_breakout_avaai_full_with_universe_4.py \
         obw_platform/tests/test_breakout_avaai_full_with_universe_5.py
```

```
# Green subset по maker/DCA, без тесту що вимагає зовнішній session DB
pytest -q tests/test_maker_dca_flow.py \
       -k 'not
session_sqlite_real_scenario_extracts_open_close_open_sequence'
```

```
# Lightweight config smoke
pytest -q obw_platform/tests/
test_backtester_core_speed3_veto_universe2_configs.py \
       -k
'cs_C2_base_1h or greedy_breakout_universe or cache_db_cli_override or
symbols_file_path_with_prefix'
```

```
# UI/backend smoke
pytest -q UI/backend/tests/test_backtest_live_validation.py \
         UI/backend/tests/test_infer_config.py \
         UI/backend/tests/test_progress.py
```

```
# Показати path drift у runner/debug bundle і тимчасово його локалізувати
PYTHONPATH=obw_platform/runners pytest -q obw_platform/tests/
test_live_debug_bundle.py
```

Команди для реального backtest run, якщо дані є

```
# Single-leg backtester з реальним cache DB
python3 obw_platform/backtester_core_speed3_veto_universe_2.py \
  --cfg obw_platform/configs/cfg_t3m_30d_newbest_full_limit_sl_2.yaml \
  --cache_db /absolute/path/to/combined_cache_3m_14400_24u.db \
  --limit-bars 7200 \
  --plots _reports/_tmp_bt
```

```
# Smoke tuner, якщо є NPZ
bash obw_platform/run_smoke_tuner.sh /absolute/path/to/
ena_ohlcv_30s_1y_from_ticks.npz
```

```
# Legacy shell script з C2 config
cd obw_platform
bash run_c2_1h_1440.sh
```

Остання команда на наданому snapshot **не відтворюється**, бо `combined_cache.db` у архіві є, але має 0 байтів і не містить таблиць.

Дані, які потрібні для різних типів тестів

Тип тесту / команди	Які файли даних потрібні
strategy logic unit tests (`test_pine_bridge`, `test_breakout_*`)	не потребують реальних market caches

Тип тесту / команди	Які файли даних потрібні
maker/DCA subset (`test_maker_dca_flow.py`, без real-session case)	не потребує зовнішніх market DB, бо генерує test data всередині
`tests/test_limit_dca_virtual.py`	не дані, а **сумісний constructor** `VirtualExchange`; зараз interface drift
`test_live_debug_bundle.py`	не дані, але потрібен **коректний import path**
`test_exchange_trace_layer.py`	відсутній модуль `runners.exchange_trace_layer`
`test_live_orderbook_collection_v6.py`	потрібні `/mnt/data/orderbook_slippage_probe_v1.py`, `/mnt/data/virtual_exchange_v6.py`, `/mnt/data/combined_cache_session(3).db`
`test_slippage_model_v2_and_core_v6.py`	потрібні `/mnt/data/slippage_directional_model_v2.py`, `/mnt/data/backtester_dual_core_dynamic_v6.py`, `/mnt/data/cryptomine_pack_dual_full_compensated_v2.py`, `/mnt/data/session(10).sqlite`, `/mnt/data/combined_cache_session(3).db`, config YAML
real backtester commands	populated historical bars in SQLite cache DB
fast-pack / tuner smoke	`.npz` із historical bars, зазвичай після побудови з ticks
live validation and matching	`session.sqlite`, `combined_cache_session.db`, trade history CSV, validation CSV exports

Якщо потім захочеш накрити все `tox`, стандартна модель — `tox run -e <env>`, а додаткові аргументи йдуть після `--`. Якщо захочеш контейнеризувати запуск, офіційні загальні форми — `docker build ...` і `docker run ...`; для монтування робочої директорії в контейнер використовуй bind mount через `--mount type=bind,source=...,target=...`. Але в поточному архіві **немає** `tox.ini`, `pyproject.toml` із tox/pytest setup, `Dockerfile` або Compose-файлів, тому це зараз не операційний шлях, а лише наступний рівень зрілості. [4]

Приклад `backtest-integrity.md`, матриця валідації і tradeoffs

Нижче — **ілюстративний приклад**, не фактичний результат архіву. Його мета — показати, яким має бути один хороший line-файл: коротким, конкретним, метрико-орієнтованим.

Приклад заповненого `continuity/lines/backtest-integrity.md`

```
# backtest-integrity

## Stage
```

```
sprout

## What This Line Is

Лінія про те, щоб equity curve не брехав. Вона тримає один стандарт:
backtest, validation UI, paper/live runner і біржові експорти не можуть бути
чотирма різними реальностями.

## Current Shape

Canonical validation flow зараз такий:

- Backtest engine for single-leg validation: `obw_platform/
backtester_core_speed3_veto_universe_2.py`
- Live/paper runtime engine candidate: `obw_platform/runners/
live_runner_dual.py`
- Validation backend: `UI/backend/api_main.py`
- Validation page: `UI/frontend/pages/backtest_live_validation.tsx`
- TV/exchange matching scripts:
  - `obw_platform/extract_bingx_window_from_tv.py`
  - `obw_platform/match_trades_tv_bingx.py`
  - `obw_platform/bingx_export_futures_trade_history.py`

Illustrative current metrics for the active short/dual family:

- equity_end_realized_total: 228.4 USDT
- equity_end_mtm_total: 224.9 USDT
- realized_pnl_total: 28.4 USDT
- trades_total: 126
- profit_factor_total: 1.34
- win_rate_total: 43.7%
- mdd_realized_%: -4.2%
- mdd_mtm_%: -8.9%
- total_fees: 12.6 USDT
- total_funding: -2.1 USDT
- median_abs_slippage_bps: 5.8
- p90_abs_slippage_bps: 14.7
- matched_exit_rate_backtest_vs_live: 92%

Minimum validation matrix:

| Window | Scope | Goal | Pass floor |
|---|---|---|---|
| 1 month | smoke | break nothing obvious | trades > 20; no metric NaN |
| quarter | month-by-month | stability | profitable in >= 2/3 months |
| 1 year | 12 monthly slices | regime robustness | no single month worse than
-12% MTM |

Required outputs per run:
```

```
| Artifact | Why required |
|---|---|
| `summary.csv` | compact metrics snapshot |
| `trades.csv` | fill-level debugging |
| `equity_mtm.csv` | truth for MTM drawdown |
| `equity_realized.csv` | truth for closed-PnL curve |
| `monthly_breakdown.csv` | regime inspection |
| `matched_orders.csv` | TV/live alignment proof |
| `slippage_summary.csv` | execution realism |
| `equity_mtm.png` | fast visual inspection |
| `drawdown_mtm.png` | risk sanity check |
| `slippage_hist.png` | execution tail risk |

## What Changed Recently

Archive inspection exposed that cheap logic tests do run, but full
reproducibility is
still blocked by path assumptions and missing cache artifacts. This line now
forces us to stop calling a strategy "good" unless it clears the validation
matrix.

## Drift Risks

- Optimizing for backtest PnL while ignoring live execution realism.
- Mixing MTM and realized metrics in the same discussion.
- Omitting fees, funding or slippage because they make the result uglier.
- Running only favorable windows and calling it robustness.
- Allowing runner-only behavior that the backtester cannot simulate.
- Using stale live session files as if they were current evidence.

## Next Useful Move

Freeze one canonical validation bundle and require it for every strategy/
config
promotion: 1 month smoke, quarter by months, 1 year by months, with summary,
monthly breakdown, MTM/realized curves, slippage summary and matched orders.
```

Таблиця tradeoffs: складність проти довіри для ключових фіч бектесту

Фіча	Інженерна складність	Виграш у довірі	Якщо не робити	Рекомендація
Warmup / prewarm bars	низька–середня	дуже високий	перші рішення “сліпі”, індикатори стартують з нуля, раннер і бектестер розходяться	**обов’язково зараз**
Fees	низька	дуже високий	завищений PnL, фальшивий PF	**обов’язково зараз**

Фіча	Інженерна складність	Виграш у довірі	Якщо не робити	Рекомендація
Funding	середня	високий для perpetual	short/perp результати перекошені	**обов’язково для perp-стратегій**
Slippage model	середня	високий	optimistic fills, overfit до ринку якого не існує	**обов’язково, але почати з простого**
Limit orders	висока	середній без якісної fill-моделі; високий лише після серйозної валідації	найнебезпечніша зона самообману, якщо fill assumptions м’які	**не робити canonical за замовчуванням, поки нема окремого parity proof**

Ілюстративний ASCII-чарт пріоритету фіч за довірою до бектесту

```
Виграш у довірі до бектесту
^
|  high      warmup        fees/funding
|                  slippage
|
|  medium                                 limit orders
|
|  low
+-----------------------------------------------------------> інженерна
складність
      low                  medium                    high
```

Mermaid flowchart: як runner і backtester мають ділити strategy interface

```
flowchart LR
    CFG[YAML config] --> SF[Strategy loader]

    CACHE[(SQLite cache / NPZ)] --> BT[Backtester]
    LIVE[Exchange OHLCV + prewarm history] --> LR[Live/Paper runner]

    SF --> API{{Shared strategy interface
    entry_signal
    manage_position
    sync_after_external_fill
    warmup_requirements
    warmup_history
    export_state_snapshot
    restore_state_snapshot}}

    API --> BT
```

```
    API --> LR

    BT --> BTOUT[Backtest outputs
    summary.csv
    trades.csv
    equity_mtm.csv
    equity_realized.csv
    plots]

    LR --> LROUT[Live outputs
    session.sqlite
    combined_cache_session.db
    status.json
    equity.csv
    orders.csv
    debug_events.jsonl]

    BTOUT --> VAL[Backtest vs live validation]
    LROUT --> VAL
```

У вашому repo цей parity уже частково тягнеться через validation flow між CSV із TradingView [5], історією виконань із BingX [6] і локальними session artifacts. Саме тому `backtest-integrity` має бути не “ще один документ”, а **місце, де зафіксовано, які метрики і які outputs взагалі дають право вірити результату**.

Дані, яких бракує, і що ти маєш дати далі

Статус артефактів, які були запитані

Артефакт	Статус у наданому архіві	Чи достатньо для планування	Що все ще бракує
README	**present** (`README.md`)	так, для первинного continuity-плану	якщо архів не відповідає current branch — дай актуальний README з гілки
File tree `find . -maxdepth 3 -type f`	з архіву частково виведено й проаналізовано	так, для первинного плану	**authoritative output** з поточної робочої гілки все одно бажаний
Configs/ YAML	**present** (`obw_platform/configs/*.yaml`)	так	потрібен shortlist: які 3–5 конфігів реально вважаються active
List of strategies	**present** через `obw_platform/strategies/*`	так	треба розмітка: canonical / legacy / experiment

Артефакт	Статус у наданому архіві	Чи достатньо для планування	Що все ще бракує
Live-runner files	**present** у `obw_platform/runners/` і `bt_live_paper_runner*.py`	так	треба однозначно назвати canonical engine і wrapper
UI-admin files	**present** у `UI/backend`, `UI/frontend`, `api`, `v/api`	так	треба сказати, що primary, а що legacy
Last 2–3 Codex tasks	**unspecified** як реальні Codex tasks	**ні**	у repo є тільки `multi_agent/task_tradebot_brief.md` і `task_tradebot_full.md`, але цього мало і це не доводить реальний останній AI workflow
Real data caches	**insufficient**	**ні** для end-to-end backtests	потрібні populated SQLite cache DB / NPZ
Sample live session	**unspecified / insufficient**	**ні** для full backtest-live integrity work	потрібен один реальний session bundle
Git metadata / actual base branch	**absent in zip**	**ні** для реального PR execution	потрібні current branch, default branch, remote, CI system

Точні дані, які ти маєш дати, щоб я пішов далі не наосліп

Надішли **рівно це** з актуального working repo, а не зі старого zip:

```
# 1. Git context
git rev-parse --show-toplevel
git rev-parse --abbrev-ref HEAD
git branch --show-current
git remote -v

# 2. Актуальний file tree
find . -maxdepth 3 -type f | LC_ALL=C sort

# 3. Активні конфіги
find obw_platform/configs -maxdepth 1 -type f -name '*.yaml' | LC_ALL=C sort

# 4. Актуальні стратегії
find obw_platform/strategies -maxdepth 1 -type f | LC_ALL=C sort
```

```
# 5. Актуальні раннери
find obw_platform/runners -maxdepth 1 -type f | LC_ALL=C sort
find . -maxdepth 2 -type f -name '*runner*.py' | LC_ALL=C sort

# 6. UI/API поверхні
find UI -maxdepth 3 -type f | LC_ALL=C sort
find api -maxdepth 3 -type f | LC_ALL=C sort
find v -maxdepth 3 -type f | LC_ALL=C sort

# 7. Data artifacts
find . \( -name '*.db' -o -name '*.npz' -o -name 'session.sqlite' \) -printf
'%P\t%s bytes\n' | LC_ALL=C sort
```

І окремо дай **один невеликий, але реальний набір даних** для відтворення:

1. **один populated SQLite cache DB** для backtest, або
2. **один NPZ** для fast-pack/tuner smoke, і
3. **один zip з live session bundle** типу `_reports/_live/<run_name>/` із:
4. `session.sqlite`
5. `combined_cache_session.db`
6. `status.json`
7. `equity.csv`
8. `orders.csv`
9. `debug_events.jsonl`
10. `stdio.log`

Що ще треба від тебе поза файлами

1. Який файл зараз вважається **canonical single-leg backtester**.
2. Який файл зараз вважається **canonical live runner**.
3. Які **3–5 configs** ти реально використовуєш, а не просто зберігаєш.
4. Які **2–3 останні реальні Codex задачі** ти давав агентам.
5. Який CI у вас справжній: локальний, GitHub Actions, GitLab, Jenkins чи взагалі ніякий.
6. Чи цей snapshot — це **current base branch**, чи просто архівний зріз.

Короткий шаблон Codex task spec, який треба впровадити

```
# Task

## Goal
Що саме треба отримати в кінці, одним реченням.

## Scope
Який шар системи дозволено змінювати.

## Do not edit
```

```
Які директорії та файли заборонено чіпати.

## Files
Які конкретні файли очікувано будуть змінені або створені.

## Test
Яку точну команду виконати для перевірки.

## Acceptance
Що має бути правдою, щоб задача вважалась завершеною.

## Report back
Коротко:
- що змінено;
- які файли змінено;
- що пройшло/не пройшло;
- які ризики або блокери лишилися.
```

Локальні команди, які варто прогнати вже зараз

```
# continuity PR smoke
bash continuity/check.sh
```

```
# suite inventory
pytest --collect-only -q
```

```
# cheap green subsets
pytest -q UI/backend/tests/test_backtest_live_validation.py \
         UI/backend/tests/test_infer_config.py \
         UI/backend/tests/test_progress.py

pytest -q tests/test_pine_bridge.py \
         obw_platform/tests/test_breakout_avaai_full_with_universe_2.py \
         obw_platform/tests/test_breakout_avaai_full_with_universe_4.py \
         obw_platform/tests/test_breakout_avaai_full_with_universe_5.py

pytest -q tests/test_maker_dca_flow.py \
       -k 'not
session_sqlite_real_scenario_extracts_open_close_open_sequence'
```

```
# packaging drift exposure
pytest -q tests/test_limit_dca_virtual.py
PYTHONPATH=obw_platform/runners pytest -q obw_platform/tests/
test_live_debug_bundle.py
```

```
# data reality check
find . \( -name '*.db' -o -name '*.npz' \) -printf '%P\t%s bytes\n' | sort
sqlite3 obw_platform/combined_cache.db ".tables"
```

Нижня лінія: **repo вже достатньо складний, щоб continuity layer дав швидку віддачу, але ще не настільки задубілий, щоб це було пізно**. Перший PR має бути тупо дисципліною: коротка карта, вісім ліній, мінімальний hook, нуль змін у trading code. Після цього вже можна нормально добивати canonical engine map, data provenance і реальну backtest/live integrity.

---

[1] [2] https://git-scm.com/docs/git-switch.html
https://git-scm.com/docs/git-switch.html

[3] [5] [6] https://docs.pytest.org/en/stable/example/pythoncollection.html
https://docs.pytest.org/en/stable/example/pythoncollection.html

[4] https://tox.wiki/en/latest/tutorial/getting-started.html
https://tox.wiki/en/latest/tutorial/getting-started.html