DEPLOYMENT-SOT-1 Plan-Review

2026-05-14 UTC · master `7f65097` · **Read-only** **Volume-as-Code-SoT identifiziert** **Repo ↔ Volume aktuell synchron**

Executive Summary: Der Bot-Container nutzt ein **named Docker volume** (`steve-tradingbot_clawbot-openclaw`) als Source-of-Truth für seinen Python-Code. Repo und Volume sind **aktuell zufällig synchron** (alle 4 Kern-Files MD5-MATCH), aber es existiert **kein automatischer Sync-Mechanismus**. Edits am Repo wirken nicht live — sie müssen über manuelle OpenClaw-Sandbox-Operation, `docker cp` oder Image-Baking ins Volume kommen. Empfohlene Zielarchitektur: **Option B (Code ins Image baken)** für reproduzierbares Deploy. Named Volume als Code-Source-Layer abschaffen.

1. Live-State (Schritt 1)

Item	Wert
master HEAD	`7f65097`
git status (post-reset)	clean
Bot Container Host-PID	3082815 — running healthy, restartCount=0
Bot main.py PID (in-container)	5201 (Watchdog-Spawn)
Worker Host-PID	2658058 (unverändert)
GUI Host-PID	2656633 (unverändert)
BINANCE_TESTNET	true
cmd 13 / mp / history	cancelled / 0 / 0
decision_logs last	14:21:38 (age 65s) — Bot schreibt aktiv
position_snapshots last	14:21:19 (age 84s)
0 Tracebacks seit 14:17	bestätigt

2. Working-Tree-Sicherheit (Schritt 2)

Action	Status
Patch gesichert	`/root/data-cleanup-1-snapshot-lifecycle-unapplied.patch` · mode 600 · 40 Zeilen · 2278 Bytes
Working Tree	`git checkout HEAD -- trading/main.py` — clean
Edit-Status	**NICHT deployed**; Patch wartet auf SoT-Klärung

3. Mount-Inventory (Schritt 3)

Bot-Container clawbot — alle Mounts

```
type    | source                                                                | dest
--------|-----------------------------------------------------------------------|-------------------------------
bind    | /projekte/Steve-TradingBot/config                                     | /home/node/.openclaw/host-config
bind    | /var/run/docker.sock                                                  | /var/run/docker.sock
volume  | /var/lib/docker/volumes/steve-tradingbot_clawbot-data/_data           | /home/node/.clawbot
volume  | /var/lib/docker/volumes/steve-tradingbot_clawbot-workspace/_data      | /home/node/workspace
volume  | /var/lib/docker/volumes/steve-tradingbot_clawbot-openclaw/_data       | /home/node/.openclaw       ←  CODE LIVE HIER
volume  | /var/lib/docker/volumes/steve-tradingbot_clawbot-memu/_data           | /home/node/.openclaw/memUdata
volume  | /var/lib/docker/volumes/steve-tradingbot_clawbot-workflows/_data      | /home/node/.openclaw/workflows
```

Live-Lokationen Kern-Files (alle im Volume `clawbot-openclaw`)

```
/home/node/.openclaw/workspace/trading/main.py           ← Bot main.py liest aus hier
/home/node/.openclaw/workspace/trading/db_emitter.py
/home/node/.openclaw/workspace/trading/execution/paper_trade.py
/home/node/.openclaw/workspace/trading/dashboard.py
/home/node/.openclaw/workspace/trading-backup-pre-bugfix/  ← Operator-Backup-Snapshot, älter
```

Hash-Vergleich Repo ↔ Volume (post Working-Tree-Reset)

Datei	Host (Repo)	Container (Volume)	Status
trading/main.py	`850d38621a`	`850d38621a`	MATCH
trading/db_emitter.py	`4b01f365f3`	`4b01f365f3`	MATCH
trading/execution/paper_trade.py	`261a794947`	`261a794947`	MATCH
trading/dashboard.py	`441a0f1d04`	`441a0f1d04`	MATCH

Befund: Repo und Volume haben aktuell **identischen Inhalt für die 4 geprüften Kern-Files** — Drift wurde durch das vorherige `git checkout HEAD --`

`trading/main.py` beseitigt.

4. Sync-Historie (Schritt 4)

Suchpfad	Befund
crontab (Host-User)	nur `steve-tradingbot-backup` (DB-pg_dump) + certbot — **kein Code-Sync**
`/etc/cron.d/`	nichts trading-code-spezifisch
Sync-Scripts unter /root, /projekte (rsync/deploy)	nur fremde Projekte (`/root/openclaw-telegram/auth-sync.sh`, `/root/.codex/.tmp/app-server-remote-plugin-sync-v1`) — nicht TradingBot-relevant
Volume-Inhalt-Datierung	main.py mtime **2026-05-12 16:29**; sonstige Files vom 2026-05-09 bis 2026-05-12
Volume-Prefix-Drama (SEC-1c-3b-FU1)	nur `steve-tradingbot_*`-Volumes vorhanden — alte `clawbot-docker_*` bereits in FU1 entfernt (✓ sauber)
`trading-backup-pre-bugfix/`-Ordner im Volume	existiert — historisches Backup vom Operator vor einem früheren Bugfix-Recreate

Kein automatischer Repo→Volume Sync existiert. Wenn die MD5-Hashes aktuell matchen, dann nur weil: (a) Volume wurde initial bei Bot-Setup mit der damaligen Repo-Version gefüllt; (b) seitdem hat niemand uncommitted Edits im Repo erstellt + Operator hat das Volume möglicherweise via OpenClaw-Sandbox/manual editiert mit identischem Inhalt. Die Symmetrie ist **brüchig**.

5. Source-of-Truth — Realität

Aktueller Stand: Die effektive Source-of-Truth für den live laufenden Bot-Code ist das **Volume** `steve-tradingbot_clawbot-openclaw`. Das Repo (`/projekte/Steve-TradingBot/trading/`) ist eine **parallele Kopie**, die zufällig synchron ist. Ein Repo-Edit wirkt nicht automatisch — er muss manuell ins Volume getragen werden.

6. Risikoanalyse

Risiko	Wahrscheinlichkeit	Impact
Repo-Edit wirkt nicht im Bot (Volume-Drift)	HOCH	Fix-Phasen scheinen erfolgreich aber sind unwirksam (= DATA-CLEANUP-1-Erfahrung)
Volume manuell editiert ohne Git-Commit → Code-Drift	MITTEL	Verlust der Reproduzierbarkeit; Operator-Workflow via OpenClaw kann unbemerkt drift erzeugen
Container-Recreate ohne Code-Re-Init	NIEDRIG	Volumes überleben Container-Recreate (default), Code bleibt im Volume erhalten — aber Operator könnte Volume mal versehentlich neu erstellen
Verlust des Volumes (z.B. `docker volume rm`)	NIEDRIG	Code weg; Rebuild aus Repo möglich, aber keine etablierte Pipeline
Falsche Annahme "Repo ist live"	HOCH	Fehlende Sicherheits-/Compliance-Audits (Git-blame ≠ live-Code)

7. Zielarchitektur — bewertet

Option	Pro	Contra	Empfehlung
A — Bind-Mount Repo (`/projekte/Steve-TradingBot` → `/home/node/.openclaw/workspace`)	Edit-im-Repo-wirkt-live, schnelle Entwicklung, git-blame valide	Host-Code wirkt sofort live → riskant bei uncommitted Drafts; legacy Volume-Daten gehen verloren falls Mount-Switch (Operator-Backup nötig)	**DEV-Setup**
B — Code baked ins Image (`COPY trading/ /home/node/.openclaw/workspace/trading/` im Dockerfile)	reproduzierbar; commit=build=deploy; klare Audit-Spur; Volume nur noch für state/logs	Image-Rebuild bei jedem Code-Change (~5 min); Container-Recreate bei jedem Deploy	**PRODUCTION (empfohlen)**
C — Named Volume + explizitem Sync-Mechanismus	kleinster Eingriff	Drift bleibt grundsätzlich möglich; ext. Sync-Tool nötig (rsync via cron oder ähnlich)	**nur Übergang**

Empfehlung: **Option B**. Dockerfile-Edit, der `COPY trading/ /home/node/.openclaw/workspace/trading/` hinzufügt. Damit ist jeder Code-Stand reproduzierbar aus dem Image herstellbar. Volume `clawbot-openclaw` bleibt für *state* (logs, .env, runtime config), aber NICHT mehr für Code.

8. Empfohlener Cut — Folgephasen

Phase	Inhalt	Aufwand	Boundary-Profil
DEPLOYMENT-SOT-1a	Plan/Inventory = dieses Dokument	fertig	read-only ✓
DEPLOYMENT-SOT-1b	Image-baked Code vorbereiten: Dockerfile-Edit (`COPY trading/ ...`), Image-Build, Import-Verifikation (psycopg2, trading.db_emitter, trading.main).	~30-60 min	1 Datei-Edit, 1 Image-Build, kein Container-Recreate, lokaler Commit, kein Push
DEPLOYMENT-SOT-1c	Bot-Recreate aus baked Image; Verifikation: Bot stabil, decision_logs läuft, kein Restart-Loop, Image=baked statt Volume=Code.	~10 min	1 Recreate, Verify, 0 Code-Touch
DEPLOYMENT-SOT-1d (optional)	Volume-Cleanup: Code-Files aus `clawbot-openclaw`-Volume entfernen (Volume behält nur state/logs); ggf. `trading-backup-pre-bugfix/` als Reference behalten.	~15 min	Volume-Mutation, separates GO Pflicht

DATA-CLEANUP-1 Wiederaufnahme	SNAPSHOT-LIFECYCLE-1 (P0) deployen über die neue baked-Image-Pipeline. Patch unter `/root/data-cleanup-1-snapshot-lifecycle-unapplied.patch` bereit, kann nach 1c eingespielt werden.	~30 min	P0-Reuse-Pfad

9. Migrationsplan B (Code-Baking) im Detail

Dockerfile-Edit (Vorschau, noch nicht ausgeführt)

```
# NEU nach Zeile mit "RUN pip3 install ... requirements.txt":

# DEPLOYMENT-SOT-1: TradingBot-Code ins Image baken.
# trading/ und main.py werden aus dem Repo eingebakt und nicht mehr
# aus dem Volume gelesen. Volume clawbot-openclaw behaelt logs, .env,
# state-Files (runtime_config.json, baseline_holdings.json, managed_state.json).
COPY --chown=node:node trading/ /home/node/.openclaw/workspace/trading/
# main.py liegt im Repo unter /projekte/.../main.py? oder trading/main.py?
# Verifiziert: Bot-Code ist ALLES unter /projekte/Steve-TradingBot/trading/
```

Operator-Decisions vor Phase 1b:

1. Soll auch `/projekte/Steve-TradingBot/main.py` (falls existent) ins Image? — Recon zeigt: `main.py` ist **nicht** im Repo-Root, sondern in `trading/main.py`. Single Source. OK.
2. Soll `trading-backup-pre-bugfix/` aus Volume bewahrt werden? — Empfehlung: ja, als Reference-Backup im Volume belassen (nicht ins Image).
3. Volume `clawbot-openclaw` verbleibt für: `logs/`, `.env`, runtime_config/baseline/managed_state-JSONs falls dort. Code wird vom Image verdrängt (Image-COPY überschreibt Volume-Inhalt im Container-Mount — aber Volume bleibt davon unberührt = ein potentielles Problem). Operator-Decision: Volume-Code-Cleanup oder Volume mit Image-Override leben lassen?

> **Wichtige Volume-Mount-Mechanik**: Wenn ein named Volume auf einen Pfad gemountet wird, der bereits Image-Inhalt hat, dann **überdeckt das Volume den Image-Inhalt**. Das heißt: ein `COPY trading/ ...` ins Image wirkt nur, wenn **entweder** (a) das Volume für diesen Pfad nicht gemountet wird (Mount-Path-Change im docker-compose.yml) **oder** (b) das Volume initial leer ist (Docker kopiert dann Image-Inhalt rein) **oder** (c) das alte Volume gelöscht/getrennt wird. Bei aktuellem Setup mit gefülltem Volume würde Option B alleine **nichts** bewirken.

→ **Korrekter Pfad B** erfordert zwingend einen der drei Sub-Steps:

1. **Mount-Path-Refactor**: Code-Pfad aus Volume rausnehmen (z.B. Code unter `/opt/bot/trading/` im Image baken statt unter `/home/node/.openclaw/workspace/trading/`, dann sys.path im Bot anpassen) — invasive Code-Pfad-Änderung.
2. **Volume rebuild**: aktuelles `clawbot-openclaw` Volume backup'en + leeren + Container neu starten — Docker re-initialisiert Volume mit Image-Inhalt.
3. **Bind-Mount-Switch** (Option A): Volume durch Bind ersetzen — radikalere Architektur-Änderung.

Empfehlung: **Sub-Pfad B-2 (Volume rebuild)** — definiertes Verfahren, einmalig Volume-Inhalt aus Image neu initialisieren, danach Image als Code-SoT.

10. Rollback-Plan

Phase	Rollback-Pfad
1b (Dockerfile + Build)	`git checkout HEAD -- Dockerfile`; altes Image-Tag (vor 1b) bleibt unbenutzt; nichts deployt
1c (Bot-Recreate)	`docker tag <pre-1c-image-sha> steve-tradingbot-clawbot:latest`; Bot-Recreate mit altem Image (analog zu TELE-1b Rollback). Volume bleibt unverändert, Code re-aktiviert sich aus altem Volume-Stand.
1d (Volume-Cleanup)	Pre-Cleanup-Backup-Volume `clawbot-openclaw-pre-sot-1d` via `docker volume create` + `cp -a` sichern; Restore bei Bedarf.
DATA-CLEANUP-1-Wiederaufnahme	identisch zu 1c-Rollback (altes Image)

11. DATA-CLEANUP-1-Wiederaufnahme — Bedingungen

> **DATA-CLEANUP-1 darf fortgesetzt werden NACH**:
>
> 1. DEPLOYMENT-SOT-1b abgeschlossen (Image hat baked trading/-Code)
> 2. DEPLOYMENT-SOT-1c abgeschlossen (Bot läuft mit baked Image, Volume-Code wird nicht mehr gelesen)
> 3. Verifikation: ein gezielter Edit in `/projekte/Steve-TradingBot/trading/test_marker.py` + Image-rebuild + Recreate → File im Container vorhanden mit Inhalt aus Repo. End-to-End Sync-Pipeline funktioniert.
>
> Erst dann ist der SNAPSHOT-LIFECYCLE-1-Patch unter `/root/data-cleanup-1-snapshot-lifecycle-unapplied.patch` via `git apply` + Image-Rebuild + Recreate deploybar.

12. Stop-Regeln (eingehalten)

- 0× docker cp
- 0× Volume-Edit
- 0× Live-Code-Patch
- 0× Bot-/Worker-Restart
- 0× DB-Migration
- 0× Mainnet
- 0× Push
- 0× Secret-Output

- 0× env-Dumps
- 0× `compose config` mit Secrets

13. GO/NO-GO

> **GO empfohlen für DEPLOYMENT-SOT-1b** (Image-baked Code vorbereiten) als nächste Phase. Operator-Decisions vor 1b:
> 1. Sub-Pfad B-1 (Mount-Path-Refactor) oder B-2 (Volume rebuild)? — Empfehlung B-2.
> 2. Volume `clawbot-openclaw` nach SOT-1d: leeren / als Backup-Bewahren / komplett löschen? — Empfehlung: Backup-Volume erstellen, dann leeren.
> 3. Mit oder ohne Production-Test im Anschluss? — Empfehlung: full live-verify (Bot main.py + DB-Writes + Worker-Heartbeat).

2026-05-14 UTC · DEPLOYMENT-SOT-1 Plan-Review · master `7f65097` · Working Tree clean · Patch gesichert in `/root/data-cleanup-1-snapshot-lifecycle-unapplied.patch`