Genesis — Bot-Übergabe-Dokumentation

Stand: 2026-05-23 · Verfasst von Genesis (Claude Code, Opus 4.7) für meinen Nachfolger

> **An dich, Nachfolger.** Du übernimmst Don's „Bot-Imperium". Du weisst noch nichts — dieses Dokument bringt dich auf Null-zu-Betriebsbereit. Lies es einmal ganz durch, bevor du irgendetwas anfasst. Ich schreibe dir konkret: volle Pfade, echte Befehle, echte Stolperfallen. Wo Geheimnisse (Tokens, Passwörter) relevant sind, nenne ich nur den Ablageort — niemals den Wert. Halte dich daran.
>
> Eine Sache vorab, die alles andere überlagert: **Du redest mit einem Menschen, nicht mit einem Ticket-System.** Don ist geduldig, vertraut dir viel an und erwartet im Gegenzug Ehrlichkeit und Sorgfalt. Lieber langsam und richtig als schnell und kaputt.

Teil 1 — Wer ist Don / Was ist Genesis

1.1 Don

- **Person**: Schweizer, deutschsprachig. Einziger Auftraggeber und einziger User. Du arbeitest ausschließlich für ihn.
- **Rolle**: CEO/Auftraggeber des Bot-Imperiums. Er gibt die Richtung vor, trifft die strategischen Entscheidungen, gibt Budgets frei. Du bist sein Orchestrator, nicht sein Vorgesetzter.
- **Erreichbarkeit**: Telegram, `chat_id 6974524199`, Bot `@Genesis1608_Bot`. Er sieht ausschließlich, was du über das Telegram-Reply-Tool sendest (dazu Teil 1.4).

1.2 Don's Arbeitsweise & Kommunikationspräferenzen

Diese Punkte stammen aus den `feedback_*`-Memories und sind nicht verhandelbar — sie sind Don's ausdrückliche Korrekturen an mir über Wochen:

1. **In-Project-Autonomie** (`feedback_autonomy.md`): Innerhalb eines laufenden Projekts darfst und sollst du eigenständig entscheiden. Er hasst Micro-Confirmations à la „soll ich jetzt Datei X öffnen?". Triff die offensichtliche Entscheidung, sag was du tust, mach weiter.
2. **ABER Approval-Gate** bei: allem **nach außen Gehenden** (Deploys, externe Sends, Veröffentlichungen), allem **Destruktiven** (Löschen/Überschreiben fremder Daten) und **jeder Geldausgabe** (Topup, Subscription, Pricing). Hier wird gefragt — vorher.
3. **Sprache**: Deutsch, Du-Form. Immer.
4. **Emoji-Receipt-Disziplin** (`feedback_emoji_receipt_confirm.md`): JEDE Telegram-Nachricht von Don zuerst mit einer Emoji-Reaktion quittieren (`react`-Tool, Whitelist: 👍 👀 ❤ 🔥 🎉 …), DANN inhaltlich antworten. Das ist sein „ich sehe, dass du's gesehen hast"-Signal.
5. **Emoji-Acknowledgement** (`feedback_emoji_ack.md`): Wenn seine Nachricht keine inhaltliche Antwort braucht (z.B. „ok", „danke"), reicht eine Emoji-Reaktion.
6. **Keine Credentials im Klartext** (`feedback_telegram_credential_hygiene.md`): Passwörter, API-Keys, Karten-Endziffern, Subscription-Details NIE im Klartext über Telegram — auch nicht an Don selbst. Verweise auf den Ablageort.
7. **Kanonische Domain** (`feedback_blackdress_canonical_domain.md`): Für BlackDress immer

`www.blackdress.ch` schreiben, NIE `blackdress-demo.pages.dev`.

8. **Pause-Windows respektieren**: Wenn Don sagt „ich mache Pause" oder „lassen wir das so", dann ruht das Thema. Keine ungefragten Edits.

1.3 Was Genesis ist

- **Ich bin Claude Code** (Anthropic's offizielle CLI, v2.1.x), Modell **Claude Opus 4.7 mit 1M-Token-Kontext**. Ich bin kein eigenständiges Produkt namens „Genesis" — „Genesis" ist der Name/die Rolle, in der ich für Don arbeite: CEO/Orchestrator seines Bot-Imperiums.
- **Wichtige Klarstellung — OpenClaw ≠ ich.** Auf demselben Server läuft parallel eine OpenClaw-Gateway-Instanz (eigener Dienst, Port 18789). Die ist auth-kaputt und hat mit mir nichts zu tun. Wenn Don OpenClaw-Begriffe erwähnt (Gateway, ACP, Standing-Orders, Pi-Agent), läuft das NICHT über dich. Lass dich davon nicht verwirren. Du bist der Claude-Code-Layer.
- **Wie ich arbeite**: Ich orchestriere über vier Bausteine — **4 Sub-Agents** (spezialisierte Unter-Bots), **29 Skills** (mein kodifiziertes Betriebswissen), das **Memory-System** (Datei-basiert, überlebt Neustarts) und das **Brain** (SQLite-Datenbank). Details in Teil 2–4.

1.4 Verbindungs-Setup (so läufst du physisch)

- **Host**: Hostinger-VPS, Linux, Arbeitsverzeichnis `/root`.
- **Dienst**: `systemd`-Unit `claude.service` (`enabled`, `Restart=on-failure`, `WantedBy=multi-user.target`) → startet automatisch bei Reboot.
- **ExecStart** (genau so, der `-A`-Schalter ist kritisch, siehe Teil 5):

```
tmux new-session -A -d -s claude -c /root /root/.local/bin/claude --channels plugin:telegram@claude-plugins-official
```

- **Du läufst** in der tmux-Session `claude`. Anhängen mit `tmux attach -t claude`.
- **Telegram-Pipeline**: Der MCP-Server `plugin:telegram@claude-plugins-official` empfängt Don's Nachrichten und injiziert sie als `<channel source="plugin:telegram:telegram" …>`-Markierung in meinen Konversationsstrom. **Mein normaler Text-Output ist für Don UNSICHTBAR** — er erreicht ihn nur, wenn ich explizit das Reply-Tool aufrufe.
- **Telegram-Token**: `/root/.claude/channels/telegram/.env` (root-only). Bot-ID 8506048436.

1.5 Top-Prinzipien (verinnerliche diese)

Prinzip	Bedeutung
Geduld / Qualität vor Tempo	Lieber eine Sache sauber + verifiziert als drei Sachen halb. Don merkt den Unterschied und schätzt ihn.
Pre-Mortem vor jedem Bau	Bevor ein neuer Bot/Sub-Bot/Major-Change entsteht: Skill `pre-mortem-bot-architecture`, Top-3-Risiken, Verdict, Don-Approval. Kein Bau ohne.
$0-Cost-Preference	Default ist die kostenlose Lösung. Claude Max deckt Coding/Sub-Agents/Crons ab. Nur externe APIs (xAI, fal.ai) kosten extra — und brauchen Geld-Reflex (`revenue-discipline`).
Timezone-Disziplin	Vor jedem Cron: `date` + `ls -l /etc/localtime` prüfen, Cron-Preview an Don. (Server steht auf Europe/Zurich.)
Disziplin beim Debuggen	Bei Test-Fails IMMER Trace lesen + Test-Bug-Hypothese prüfen, BEVOR du „skip später" machst. Nie destruktive Shortcuts (`--no-verify`, `rm -rf`, `--force`) ohne Auftrag.

Memory vor Kontextverlust	Vor `/compact` oder langen Sessions: Zustand in Memory-Files externalisieren. Detail darf nicht verloren gehen.
Citation-Constraint	Keine erfundenen Fakten/Quellen. Wenn du etwas nicht weisst, sag es.
Ehrlichkeit	Wenn Tests failen: sag es mit Output. Wenn ein Schritt übersprungen wurde: sag es. Keine Schönfärberei.

Teil 2 — Aktive Projekte

> Reihenfolge nach Wichtigkeit für Don: **BlackDress → Trading Bot → Image Bot → HR Bot.** Es gibt genau **4 aktive Säulen** plus das Dashboard. Früher gab es mehr (Knowledge-Imperium, Knowledge-Bot, IdeaHunter, Trading-Dashboard, ein Grok-Bot, ein NOCTIS-Wellness-Demo) — die wurden alle bewusst entfernt (siehe Teil 5 / Glossar „Teardown"). Wenn du in alten Memories davon liest: das ist Geschichte, nicht Gegenwart.

2.1 BlackDress ☆ (das wichtigste Projekt)

- **WAS**: Premium-Dessous-Onlineshop, live unter `www.blackdress.ch`.
- **WARUM**: Der Cashflow-Anker des Imperiums. Don's erstes echtes umsatzorientiertes Projekt — hier soll Geld verdient werden. Sein Pain-Point: ein eigener Shop ohne teure Plattform-Gebühren, voll unter Kontrolle.
- **WO** (`/root/blackdress/`):
  - `build_phase1.py` — Statik-Build-Pipeline
  - `mw_data/cron_update.py` — Lagerbestand-Sync vom Lieferanten
  - `mw_data/uploads/` — hier landen die Lieferanten-CSVs (Trigger für den Sync)
  - `gate.js` / `gate.js.src` — das Pre-Launch-Passwort-Gate
  - `workers/email/` — Cloudflare-Worker-Skelett für transaktionale Emails (provider-agnostisch)
  - `d1/` — Cloudflare-D1-Datenbankschema
  - `manual/` — Betriebshandbücher (auch als PDF im Dashboard)
  - `assets/`, `blog/`, plus ~72 HTML-Seiten
- **WIE**:
  - Statische Site auf **Cloudflare Pages** (Projekt `blackdress-demo`), Domain via Wix-DNS auf die Pages-URL gemappt.
  - **Deploy**: `bash /root/deploy_cloudflare.sh` (ein Befehl, ~30–60s).
  - **Lagerbestand-Cron**: `0 0,5,10,15,20 * * *` (5×/Tag) → `cron_update.py`. Läuft leer, wenn keine neuen CSVs in `mw_data/uploads/` liegen.
  - **Pre-Launch-Gate** aktiv: Die Seite ist passwortgeschützt (PBKDF2). Passwort liegt in `/root/.blackdress_credentials` (root-only). Vor echtem Launch muss das Gate entfernt werden.
- **ABHÄNGIGKEITEN**:
  - Cloudflare Pages + Cloudflare-Token (`/root/.cloudflare`, root-only)
  - Domain bei Wix registriert (kein Cloudflare-Transfer möglich, `.ch`-Domain) — DNS zeigt auf Pages
  - Lieferant **MW Grosshandel** — ~3124 Produkte, 14+ Marken (Passion, Provocative, Noir

Handmade, Andalea …) bereits im Code
    - Cloudflare **D1**-DB `blackdress-test` (Prod-DB existiert noch NICHT)
    - Email-Versand: Provider noch nicht gewählt (Brevo ~CHF 25/Mo vs. Cloudflare Email Workers free)
    - Zahlung: **Payrexx** — API-Keys liegen noch nicht vor, Checkout ist aktuell ein Mock

- **BEKANNTE BAUSTELLEN** (Stand 2026-05-23, alle warten auf Don-Entscheidungen):

    1. **Deploy-Aktualität**: Lokaler Code hat die 3124 MW-Produkte; ob die Live-Seite den neuesten Stand zeigt, vor jedem „ist das live?"-Statement per Deploy/Check verifizieren.
    2. **Payrexx-Integration** fehlt (Checkout = Mock).
    3. **Email-Provider** unentschieden → Worker-Skelett wartet.
    4. **Native-Speaker-Review** FR/IT/EN offen.
    5. **D1-Prod-Migration** offen.
    6. **Pre-Launch-Gate-Removal** offen.

- **WAS NICHT TUN**:

    - Keine Edits in Don's Pause-Window ohne expliziten Anlass.
    - In der Kommunikation IMMER `www.blackdress.ch`, nie die `.pages.dev`-URL.
    - Keine Klartext-Kundenemails irgendwo loggen/senden (Datenschutz). Kundendaten nur gehasht (SHA256+Salt) verarbeiten — siehe Skill `customer-lifecycle`.
    - Es gibt einen Sub-Agent `blackdress` für Site-Arbeit — nutze ihn für Content/Layout/Deploy statt alles selbst zu machen.

2.2 Trading Bot

- **WAS**: Solana-Memecoin-**Paper**-Trading-Bot (simuliert, kein echtes Geld).
- **WARUM**: Don will eine Trading-Strategie datenbasiert iterieren, ohne reales Kapital zu riskieren. Jede Version (v1, v2, … aktuell v36) ist ein Strategie-Experiment mit Endreport.
- **WO** (`/root/trading-bot/`):

    - `simulation_v29.py` — die aktive Engine (sie trägt v29 im Namen, lief zuletzt als Strategie-Version „v36"; nicht verwirren lassen, dazu Glossar)
    - `lifecycle_v29.py` — Position-Lifecycle/Exits
    - `kelly_sizer.py` — Position-Sizing (liest jetzt das Trade-Ledger, NICHT mehr das gelöschte trading-dashboard)
    - `trade_ledger.py` — append-only Per-Trade-Persistenz
    - `logs/trades/<version>/YYYY-MM-DD/*.json` + `ledger.jsonl` — das Trade-Journal
    - `watchdog.sh` — Auto-Restart-Watchdog
    - `venv/` — eigenes Python-venv
    - `.v29_kill_switch` — wenn diese Datei existiert, ist der Bot bewusst gestoppt

- **WIE**:

    - Paper-Sessions werden manuell oder per Cron gestartet.
    - **Watchdog-Cron**: `*/5 * * * * /root/trading-bot/watchdog.sh` — restartet nur, wenn Prozess gestorben UND `active_flag` gesetzt. Bei gesetztem Kill-Switch bleibt er idle.
    - **Aktueller Stand**: Version **v36 ist beendet** (Auto-Kill bei Max-Drawdown), **Kill-Switch ist gesetzt** → es läuft KEINE Session.

- **ABHÄNGIGKEITEN**: Solana RPC · DexScreener (Discovery) · Helius (optional, Full-Mode) · Jupiter

(Quotes) · rugcheck.xyz (Rug-Check) · xAI-Grok (optional) · `venv`.

- **BEKANNTE BAUSTELLEN**:
  - v36 mit Verlust geschlossen.
  - **LADA** und **TOLYBOT** sind zwei verwaiste offene Paper-Positionen (nie geschlossen, je ~100 CHF) — Don muss entscheiden, was damit passiert.
  - v37-Setup wartet auf Don.
  - Der Bot wurde vom (gelöschten) Trading-Dashboard entkoppelt — er läuft jetzt Dashboard-los.
- **WAS NICHT TUN** (das Wichtigste im ganzen Dokument):
  - **NIEMALS den Live-Mode autonom aktivieren.** Paper-Trading bleibt Paper, bis Don explizit und schriftlich Real-Money freigibt — mit eigenem Approval-Flow (siehe Skill `wallet-security`).
  - Trading-Skills waren lange „FROZEN" (`policy_trading_skills_freeze.md`). Im Zweifel: nichts an der Trading-Strategie ändern ohne Don's expliziten Trading-Brief.
  - Es gibt einen Sub-Agent `trading-bot` für Code-Iterationen — er ist per Definition so gebaut, dass er Live-Mode nie selbst togglet.

2.3 Image Bot

- **WAS**: Bildgenerator über fal.ai (Flux-Modelle), als Sub-Agent `genesis-image`.
- **WARUM**: Marketing-, Lifestyle- und Produktbilder für BlackDress und die Brand — ohne Fotoshooting.
- **WO**: `/root/genesis-image/` (Code + venv), Agent-Config `/root/.claude/agents/genesis-image.md`.
  - CLI: `/root/genesis-image/venv/bin/python3 /root/genesis-image/genesis_image.py`
- **WIE**: On-demand. Du delegierst an den Sub-Agent `genesis-image` (oder rufst das Script direkt). Modelle u.a. flux-pro ($0.04), flux-pro-ultra ($0.06), flux-dev ($0.025), flux-schnell ($0.003).
- **ABHÄNGIGKEITEN**: fal.ai-API-Key (`/root/genesis-image/.env`, root-only). Guthaben ~$63.
- **BEKANNTE BAUSTELLEN**: Läuft. Einziger Punkt: das fal.ai-Guthaben im Auge behalten.
- **WAS NICHT TUN**: Jeder Bild-Call kostet echtes Geld. Keine Massengenerierung ohne Anlass. Bei Guthaben-relevanten Aktionen Geld-Reflex (`revenue-discipline`).

2.4 HR Bot

- **WAS**: HR-/Organisationsentwicklungs-Sub-Agent. Auditiert die Performance der Sub-Bots und schlägt strukturiert neue Bots vor.
- **WARUM**: Schutz vor „Bot-Sprawl". Don will skalieren, aber kontrolliert — der HR-Bot ist die Kapazitäts- und Pre-Mortem-Bremse, bevor neue Bots entstehen.
- **WO**: Agent-Config `/root/.claude/agents/hr-bot.md`. Vorschläge unter `/root/.claude/projects/-root/memory/hr_proposals/{pending,approved,rejected}/`.
- **WIE**: War als täglicher Cron um 17:15 geplant. **Wichtig**: Ein Sub-Agent kann sich nicht selbst starten — Genesis (du) muss den HR-Audit triggern. Aktuell gibt es nach dem Teardown keine aktiven Claude-Crons (`CronList` ist leer), also läuft HR nur, wenn du es anstößt.
- **ABHÄNGIGKEITEN**: Brain (Lessons/Decisions), Memory-Files.
- **BEKANNTE BAUSTELLEN**: Kein aktiver Cron → manueller/Genesis-getriggerter Lauf nötig.
- **WAS NICHT TUN**: Der HR-Bot **baut nie selbst Bots** — er schlägt nur vor. Disziplin-Regeln: ROI-Schwelle (>5h/Woche gespart ODER klarer Qualitätsgewinn), max. 2 Vorschläge pro Audit. Bau

passiert erst nach Don's Approval.

2.5 Bot-Imperium-Dashboard (Infrastruktur, kein eigenständiges „Projekt")

- **WAS**: Admin-Dashboard, das den Zustand des Imperiums visualisiert (Bots, TODOs, Hierarchie, Skills, Tagebuch, Operations-Manuals zum Download).
- **WO**: `/root/dashboard/` (`index.html`, `app.js`, `styles.css`, `data.json`, `manual/`).
- **WIE**: Statische Site auf Cloudflare Pages (Projekt `bot-imperium`), Live: `bot-imperium.pages.dev` (passwortgeschützt, PBKDF2). **Deploy**: `bash /root/deploy_bot_imperium.sh`.
    - **Daten**: alles liegt in `data.json` — das Dashboard ist rein JSON-getrieben. Änderungen = `data.json` patchen, dann deployen.
    - **Auto-Update-Policy** (`feedback_dashboard_autoupdate.md`): Bei jeder relevanten Änderung das Dashboard proaktiv aktualisieren, ohne dass Don fragt.
    - **Manuals-Sektion**: `index.html` hat eine PBKDF2-geschützte Sektion `manual-section`, die Download-Links zu den PDFs in `/root/dashboard/manual/` freischaltet. **Hier landet auch diese Übergabe-Doku** (siehe Teil 5 / Phase-3-Hinweis).

Teil 3 — Skills (29 Stück, komprimiert)

> Skills sind mein kodifiziertes Betriebswissen. Sie liegen in `/root/.claude/skills/<name>/SKILL.md` und werden **on-demand** geladen (über das Skill-Tool), wenn die Trigger-Phrasen einer Anfrage passen. `genesis-protocol` und `memory-discipline` sind die wichtigsten — sie definieren, WIE ich arbeite.
>
> Format hier kompakt: **Zweck · Trigger · Failure-Modus** (worauf du achten musst). Volle Details stehen jeweils im SKILL.md.

Gruppe A — Genesis-Operating (immer relevant)

Skill	Zweck	Wann triggern	Failure-Modus
genesis-protocol	Master-Betriebsprotokoll: Kommunikationsstil (DE/Du), Pre-Mortem-Pflicht, Dashboard-Auto-Update, $0-Cost, Memory-Workflow	IMMER bei Gesprächen mit Don	Wenn ignoriert: falscher Ton, vergessene Disziplin-Regeln
memory-discipline	Memory-Files verwalten: Typen, MEMORY.md-Index, Pre-Compact-Externalisierung, Brain-vs-Memory	Bei „save/merken/ Memory aktualisieren", vor `/compact`, nach Milestones	Duplikate anlegen; MEMORY.md-Index vergessen; Body in den Index schreiben
pre-mortem-bot-architecture	Risiko-Vetting VOR jedem neuen Bot/ Major-Change: Top-3-Risiken, GO/ WAIT/REVISED/DROP, Don-Approval	„neuer Bot", „wir bauen", „Architektur", „soll ich"	Bau ohne Pre-Mortem = häufigste teure Fehlentscheidung
opportunity-vetting	5-Kriterien-Scoring (0–100) für jede neue Idee (Time-to-Cash, $-Invest, Fit, Don-Zeit, Reversibilität)	„neue Idee", „lohnt sich", „sollten wir", >2h Aufwand oder >10 CHF	Ideen bauen, die Don's Zeit/ Geld nicht wert sind

revenue-discipline	Geld-Reflex: ROI-Tagging, Opportunity-Cost, Burn-Rate, Cost-Approval-Workflow	JEDE Cash-Aktion (Topup, Subscription, Pricing), jede ROI-Frage	Geld ausgeben ohne Don-Approval; Kosten verschleiern
cross-bot-synergy	6 Muster, WANN Bot A die Daten von Bot B nutzt — und wann Trennung besser ist	„Cross-Sell", „Synergie", „warum getrennt", neue Features	Bots vorschnell zusammenführen → Kopplungs-Chaos
daily-cron-orchestration	Cron-Stundenplan: Pause-Windows, Conflict-Avoidance, Watchdog-Recovery	neue Crons, „missed cron", Stundenplan-Konflikte	Cron-Slot-Kollisionen; übersehene Skips
bot-imperium-deploy	Standard-Deploy-Workflow für die Dashboards (data.json patch → deploy)	„deploy", „update dashboard", „patch data.json"	Deploy ohne Backup; ungültiges JSON pushen

Gruppe B — Business / Domain-Operations

Skill	Zweck	Wann triggern	Failure-Modus
blackdress-operations	Brand-Voice, Produkt-/Kunden-/Bestell-Workflows, Datenschutz-Boundaries für den Shop	„blackdress", „produkt anlegen", „kundensupport", „retoure", „newsletter"	Off-Brand-Ton; Datenschutz-Verstoß bei Kundendaten
customer-lifecycle	CRM für BlackDress: Acquisition, Repeat-Trigger T14/T30/T60, Win-Back, NPS — Privacy-First (SHA256+Salt, KEINE Klartextemails)	„Kunde", „Repeat", „Retention", „Win-Back", „Newsletter"	Klartextemails speichern (DSGVO-Verstoß)
image-bot-operations	Prompt-Framework, Brand-Guidelines, Quality-Gates für Bildgenerierung	„bild", „produktfoto", „instagram post", „render"	Geld für unbrauchbare/ Off-Brand-Bilder verbrennen
hr-operations	Vertrauliches, rechtssicheres HR-Framework (Audits, Recruiting, Onboarding, Feedback)	„hr-bot", „hr audit", „team", „performance review"	Vertraulichkeit verletzen; Bot selbst bauen statt vorschlagen

Gruppe C — Trading (defensiv; teils PARKED/Draft bis Don's Trading-Brief)

> Diese Skills definieren das Risk-Management des Solana-Bots. Sie sind defensiv ausgelegt. Grundregel: kein Skill aktiviert je den Live-Mode.

Skill	Zweck	Wann triggern	Failure-Modus
trading-solana-memecoins	Dach-Skill: Risk-Framework, Pre-Trade-Checks, Position-Sizing, Exit-Disziplin, Eskalation an Don	„trading-bot", „solana", „buy/sell signal", „kill switch", „v24+"	Risk-Regeln umgehen; ohne Eskalation handeln
rug-detection	Pflicht-Pre-Buy-Check, 7 Hard-Gates (Mint/Freeze renounced, Liquidity-Lock, Top-10 <30%, Holder >300, Honeypot-Sim) via rugcheck.xyz	„rug", „scam", „honeypot", „pre-buy-check"	Token ohne Check kaufen → Rug
dev-wallet-rug-detection	Erweiterung: Dev-Wallet-Historie + Rug-Streak via Helius; Hard-Gate bei 2+ Rugs/30d	„dev-wallet", „rugger", „launch-history"	Serien-Rugger übersehen

kelly-position-sizing	Fractional-Kelly (25%), Cap 3%/Floor 0.5% Wallet; bei –EV: NO-TRADE	„kelly", „position-sizing", „EV"	Überdimensionierte Positionen; –EV-Trades
smart-money-tracking	Auto-Liste von Smart-Wallets; bei Discovery: Confirm wenn ≥2 Wallets aktiv	„smart-money", „kol-wallets", „alpha-wallet"	False-Positive-Wallets folgen
live-shadow-tracking	Read-only Wallet-Tracking (Solana-RPC/ Helius), 5 Wallets, NUR Lerndaten, KEIN Copy-Trade	„shadow", „wallet-tracking", „smart-money"	Mit Copy-Trading verwechseln
trade-journaling	Brain-integriertes Trade-Journal (Pre-Hypothese, Outcome, Lesson) + Daily/ Weekly-Roll-Up	„journal", „trade-log", „post-mortem"	Lessons gehen verloren (kein Journal-Eintrag)
realistic-paper-trading	Slippage/Failed-TX/MEV-Sim für ehrlichen Paper-Modus ab v27	„v27", „Slippage", „MEV", „Realistic-Mode"	Zu optimistische Paper-Resultate (kein Realismus)
dynamic-exit-strategy	Adaptive Exits: Trailing-Stop nach MCAP, Volume-Decay, Time-Stop	„exit", „trailing-stop", „take-profit", „time-stop"	Gewinne nicht sichern / zu spät raus
mev-protection	Jito-Bundles, Dynamic-Slippage, Anti-Sandwich, Tx-Sim pre-submit	„mev", „sandwich", „jito", „priority-fee"	Sandwich-Verluste
wallet-security	Wallet-Architektur (Cold/Hot/Bot-Trennung), Seed-Disziplin, Approval-Flow für Real-Trades	„wallet", „seed", „real-money", „cold/hot-wallet"	Seed-Leak; Real-Money ohne Approval-Flow
tax-ch-crypto	CH-Krypto-Steuer-Compliance (gewerbsmäßig-Schwellen, 31.12.-Snapshot, Capital-Gains steuerfrei)	„Steuer", „Vermögen", „gewerbsmäßig", „31.12-Snapshot"	Gewerbsmäßig-Status übersehen; NICHT für Firmensteuer
trading-bot-version-iteration	Versions-Disziplin (v22→v23→…): 6-Session-Test-Zyklus, Decision-Gates (Hit-Rate ≥15%), Endreport-Format	„neue Trading-Version", „Test-Cycle", „Endreport", „Hit-Rate"	Versionen ohne Gate/ Report iterieren

Gruppe D — VERWAIST (Ziel-Bot gelöscht — als deprecated behandeln)

> Diese Skill-Dateien existieren noch (Don's Regel: Skills bleiben beim Teardown unberührt), aber **ihr Ziel-Bot ist gelöscht**. Trigger sie nur, wenn Don den jeweiligen Bot bewusst wiederbelebt. Sonst: ignorieren.

Skill	War für	Status
knowledge-bot-operations	Knowledge-Bot (RSS+BM25-Recherche-Engine)	⬭ Bot gelöscht 2026-05-23
knowledge-bot-feed-management	Knowledge-Bot Feed-Verwaltung	⬭ Bot gelöscht 2026-05-23
x-wall-monitoring	Knowledge-Imperium X-Wall (Krypto-Twitter via Nitter)	⬭ Dashboard gelöscht 2026-05-23
ideahunter-research	IdeaHunter (Daily-Trend-Scanner)	⬭ Bot gelöscht 2026-05-23

Built-in CLI-Skills (nur zur Kenntnis, nicht imperiums-spezifisch)

Claude Code bringt eigene Skills mit, die du bei Bedarf nutzt: `telegram:configure` / `telegram:access` (Telegram-Setup — nur Don im Terminal), `update-config` (settings.json/Hooks/Permissions), `verify` / `run` (App starten & Verhalten prüfen), `code-review` / `review` / `security-review`, `loop` (Intervall-Tasks), `schedule`

(Cron-Routinen), `claude-api` (Anthropic-SDK-Apps), `init`, `keybindings-help`, `fewer-permission-prompts`. Diese brauchen keine 0.5-Seite — schlag im Zweifel die Skill-Beschreibung nach.

Teil 4 — Memory-System

Das ist dein Gedächtnis über Sessions hinweg. **Ohne Memory bist du bei jedem Neustart blank** — deshalb ist Disziplin hier überlebenswichtig.

4.1 Zwei Schichten

Schicht A — Memory-Files (datei-basiert, auto-geladen)

- **Pfad**: `/root/.claude/projects/-root/memory/`
- **`MEMORY.md`** ist der Index. Er wird **bei jeder Session automatisch in deinen Kontext geladen** (max ~200 Zeilen, danach abgeschnitten). Pro Eintrag eine Zeile: `- [Titel](datei.md) — Ein-Zeilen-Hook`. **Niemals Memory-Inhalt direkt in MEMORY.md schreiben** — nur den Pointer.
- Jede eigentliche Memory ist eine eigene `.md` mit Frontmatter:

  ```
  ---
  name: kurz-kebab-slug
  description: Ein-Zeilen-Zusammenfassung (für Relevanz-Erkennung)
  type: project | feedback | policy | reference | skill | user
  ---
  ```

  Bei `feedback`/`project` folgen im Body oft **Why:** + **How to apply:**-Zeilen. Querverweise mit `[[anderer-name]]`.
- **Dateitypen**: `user_*` (über Don), `feedback_*` (Don's Korrekturen/Präferenzen, mit Begründung), `project_*` (laufende Arbeit), `policy_*` (strategische Entscheidungen), `reference_*` (Pointer auf externe Systeme), `skill_*` (on-demand-Skill-Pointer), `architecture_*` (wie ich funktioniere). Plus Unterordner `hr_proposals/`.

Schicht B — Brain (SQLite, NICHT auto-geladen)

- **Pfad**: `/root/genesis/brain/brain.db`. API: `/root/genesis/brain/tools.py`.
- 3 Tabellen: `facts` (key/value), `decisions` (topic/verdict GO|REVISED|DROP|WAIT/decided_by), `lessons` (context/outcome/lesson). Plus Tag-Tabellen + Soft-Delete.
- Zugriff on-demand: `python3 -c "from brain.tools import BrainQuery; ..."` (aus `/root/genesis/`).
- Backup-Script: `/root/genesis/brain/backup_brain.sh`. Sonntags-Snapshot als Markdown unter `/root/genesis/brain/snapshots/`.

4.2 Wann Memory-File, wann Brain?

Inhalt	Wohin	Warum
Don's Stil / Präferenzen / wer er ist	Memory-File (`feedback_`/`user_`)	wird auto-geladen
Architektur eines aktiven Projekts	Memory-File (`project_`)	sofortige Re-Orientierung

Open Loops / pending Decisions	Memory-File (`project_open_loops.md`)	überlebt `/compact` + Neustart, sofort lesbar
Einzel-Entscheidung mit Begründung (Don's GO/DROP)	Brain (`decisions`)	append-only Audit-Trail, abfragbar
Lessons aus Bugs/Fehlschlägen	Brain (`lessons`)	Cross-Session-Lookback für Pre-Mortem
Tagesnotizen	Brain (`facts`, ttl_days=14)	Auto-Cleanup

Anti-Pattern: dieselbe Info in Memory UND Brain → Drift. Pro Inhalt EINE primäre Ablage.

4.3 Updaten / Erstellen / Markieren

1. Vor dem Anlegen: `ls` im Memory-Ordner — gibt es schon eine ähnliche Datei? Dann **updaten statt neu**.
2. Neue Datei mit korrektem Typ-Präfix + Frontmatter schreiben.
3. **MEMORY.md-Index-Eintrag** ergänzen (Pflicht!).
4. **Löschen ≠ rm**: Bewährtes Muster ist `status: deleted` im Frontmatter + ein Lösch-Banner oben, **History bleibt erhalten**. So wurde der Knowledge-Bot, IdeaHunter etc. behandelt.
5. Bei Konflikt Memory vs. Realität: **Realität gewinnt.** Memory korrigieren, nicht stillschweigend an Veraltetem festhalten.

4.4 Lesereihenfolge bei Session-Start (mach das immer)

1. **`MEMORY.md`** — ist schon im Kontext (auto-geladen). Verschafft Überblick.
2. **`project_open_loops.md`** — als READ-FIRST markiert: enthält den verifizierten Gesamtstand (System, aktive Säulen, offene Decisions). **Das ist dein wichtigster Einstieg.**
3. **`architecture_how_i_work.md`** — wie du technisch funktionierst (Service, Channels, Tools, Boundaries).
4. Dann die relevanten `project_*`-Files zum aktuellen Thema.

Teil 5 — Operativer Notfall-Kasten

5.1 Service-Restart

```
systemctl status claude      # läuft der Dienst?
systemctl restart claude     # Neustart (startet tmux-Session + Claude + Telegram-Plugin)
tmux attach -t claude        # an die laufende Session anhängen (Ctrl-b d zum Lösen)
journalctl -u claude -n 50   # letzte Logzeilen
```

Bei Reboot startet `claude.service` automatisch (`enabled`, `WantedBy=multi-user.target`).

5.2 Telegram wieder verbinden

- **Ursache Nr. 1 für Telegram-Ausfälle war ein Duplicate-tmux-Session-Bug.** Der Fix steckt im ExecStart: `tmux new-session -A` (das `-A` macht den Start idempotent — hängt an eine existierende Session an, statt eine zweite zu erzeugen). Wenn Telegram tot ist, prüfe zuerst, dass

`claude.service` genau diesen ExecStart hat.

- **Reconnect-Orchestrator**: `/root/_telegram_reconnect.sh` (killt alte manuelle Session, startet den Dienst sauber, verifiziert). Verify-Log: `/root/_telegram_reconnect.log`.
- **Backup des funktionierenden Service-Files**: `/root/_archive/claude.service.bak-2026-05-23`.
- **Token**: `/root/.claude/channels/telegram/.env` (root-only). Bot `@Genesis1608_Bot`.
- **Tool-Mechanik**: Die Telegram-Tools (`react`, `reply`, `edit_message`, `download_attachment`) sind „deferred" — du musst sie pro Session einmal per `ToolSearch` laden: `ToolSearch select:mcp__plugin_telegram_telegram__react,mcp__plugin_telegram_telegram__reply`.

5.3 Backups & sauberster Restore-Punkt

- **Sauberster Restore-Punkt**: `/root/_archive/teardown-2026-05-23/` — enthält alles, was beim großen Teardown entfernt wurde (Knowledge-Stack, Trading-Dashboard-Tarballs, IdeaHunter), mit MANIFEST + SHA256SUMS.
- **Heute gelöschte Bots/Demos**: `/root/_archive/deleted-2026-05-23/` (grok-genesis-bot, bd-wellness-demo, grokbot_env).
- **Weitere Backups**: `/root/genesis/backups/` (pre-restart-Snapshots), `/root/dashboard/backups/` (data.json-Versionen vor Änderungen).
- **Host-Backups**: Hostinger macht tägliche Snapshots (von Don abonniert). Kein Off-Site-Backup (GitHub wurde diskutiert, nicht umgesetzt).

5.4 Credential-Orte (NIE im Klartext ausgeben — nur referenzieren)

Datei (root-only)	Inhalt
`/root/.cloudflare`	Cloudflare API-Token + Account-ID (für alle Deploys)
`/root/.blackdress_credentials`	BlackDress Pre-Launch-Gate-Passwort
`/root/.dashboard_credentials`	Dashboard-Passwörter
`/root/genesis-image/.env`	fal.ai-Key
`/root/.claude/channels/telegram/.env`	Telegram-Bot-Token

Wrangler-Deploys erwarten die CF-Variablen aus `/root/.cloudflare` (die Deploy-Scripts sourcen sie selbst).

5.5 Häufige Bugs & Lösungen

Symptom	Ursache	Fix
Telegram antwortet nicht; mehrere claude-Prozesse	Duplicate-tmux-Session	ExecStart auf `tmux new-session -A` stellen, Dienst neu starten
Prozess gekillt, RAM voll (OOM)	zu wenig Speicher	4 GB Swap ist eingerichtet + reboot-fest (`/root` Swap-File). Bei erneutem OOM: Swap prüfen (`swapon --show`)
Cron feuert zur falschen Zeit	TZ-Verwechslung	VOR jedem Cron: `date` + `ls -l /etc/localtime` (= Europe/Zurich) prüfen, Preview an Don
Crons nach Reboot verschwunden	session-only CronCreate-Jobs überleben Reboot nicht	SSoT pflegen: `/root/genesis/cron_renewer/desired_crons.json` — daraus werden Crons wiederhergestellt

„OpenClaw"-Fehler im Log	parallele, auth-kaputte Gateway-Instanz	ignorieren — ist NICHT Genesis (siehe Teil 1.3)

5.6 Deploy-Befehle (beide outward-facing → Don-Approval einholen)

```
bash /root/deploy_bot_imperium.sh     # Bot-Imperium-Dashboard → bot-imperium.pages.dev
bash /root/deploy_cloudflare.sh       # BlackDress → www.blackdress.ch
```

Teil 6 — Glossar

Allgemein / System

- **Bot-Imperium**: Don's Gesamtsystem aus Genesis + Sub-Bots + Dashboard.
- **Genesis**: Rolle/Name, in der Claude Code als CEO/Orchestrator für Don arbeitet (= du).
- **Sub-Bot**: spezialisierter Unter-Agent (`/root/.claude/agents/*.md`), den Genesis per Agent-Tool aufruft. Aktuell 4: blackdress, genesis-image, trading-bot, hr-bot.
- **Sub-Sub-Bot**: ein Bot unter einem Sub-Bot (Namensschema `Elternbot_Thema`). Aktuell keiner aktiv.
- **Brain**: SQLite-DB (`/root/genesis/brain/brain.db`) als zweite Gedächtnisschicht (facts/decisions/lessons).
- **Memory-File**: `.md`-Datei unter `/root/.claude/projects/-root/memory/`, auto-geladen via MEMORY.md.
- **SSoT-Crons**: „Single Source of Truth" für Crons — `/root/genesis/cron_renewer/desired_crons.json`. Schützt vor Cron-Verlust bei Reboot.
- **CronQueue** / **CronRenewer**: Mechanik aus dem „Imperium-v2"-Ausbau, die Cron-Jobs aus der SSoT wiederherstellt/abarbeitet.
- **Pre-Mortem**: Pflicht-Risikoanalyse vor jedem Bau (Skill `pre-mortem-bot-architecture`).
- **4Cs** / **3Ms**: Bewertungs-Frameworks im Dashboard (4Cs = Cadence-/Audit-Metriken; 3Ms = Übersichts-Sektion). Historisch, im Dashboard sichtbar.
- **Pause-Window**: reservierte Zeitfenster (z.B. 17:08–17:26), in denen bestimmte Crons Vorrang haben — Conflict-Avoidance.
- **Stundenplan**: der Cron-Zeitplan des Imperiums (Skill `daily-cron-orchestration`).
- **REPL-busy**: Zustand, in dem ein Cron nicht feuern konnte, weil Genesis gerade beschäftigt war.
- **Teardown**: das geplante, non-destruktive Entfernen ganzer Säulen (mit Backup nach `/root/_archive/`). Großer Teardown am 2026-05-23: Knowledge-Imperium, Knowledge-Bot, Trading-Dashboard, später grok-genesis-bot, bd-wellness-demo, IdeaHunter.
- **Imperium-v2 (Phase A/B/C)**: Ausbaustufe, in der Brain + Disziplin-Skills + CronQueue eingeführt wurden.
- **Claude Max**: Don's Anthropic-Abo ($216/Mo). Deckt Coding/Sub-Agents/Crons ab; externe APIs (xAI, fal.ai) kosten extra.
- **OpenClaw**: separate, auth-kaputte Gateway-Software auf demselben Host — NICHT Genesis.

Trading

- **„v36" (Version)**: laufende Nummer eines Strategie-Experiments des Trading-Bots. Achtung: Die Engine-Datei heißt `simulation_v29.py`, trug zuletzt aber die Strategie-Version v36 — Datei-Versionsnummer ≠ Strategie-Versionsnummer.
- **Paper-Trading**: simuliertes Trading ohne echtes Geld (aktueller Modus, der bleibt bis Don Real-Money freigibt).

- **Kill-Switch**: Datei `/root/trading-bot/.v29_kill_switch` — wenn vorhanden, ist der Bot bewusst gestoppt, Watchdog bleibt idle.
- **Anti-Gap-Protection**: Schutzlogik gegen Preis-Lücken zwischen Datenpunkten.
- **Kelly-Sizing**: Positionsgrößen-Methode (Fractional-Kelly), Skill `kelly-position-sizing`.
- **Rug** / **Rug-Detection**: Memecoin-Betrug, bei dem Liquidität abgezogen wird; Pre-Buy-Check dagegen.
- **Smart-Money**: erfolgreiche Wallets, deren Verhalten als Signal dient.
- **Realistic-Paper-Mode**: Paper-Trading mit simuliertem Slippage/MEV/Failed-TX für ehrlichere Resultate (ab v27).

BlackDress

- **Pre-Launch-Gate**: PBKDF2-Passwortschutz vor der noch nicht öffentlichen Shop-Seite.
- **MW Grosshandel**: der Dessous-Lieferant (3124 Produkte im Code).
- **Payrexx**: Schweizer Payment-Provider, für den Checkout vorgesehen (Keys noch offen).
- **D1**: Cloudflare-SQL-Datenbank (Test-DB existiert, Prod fehlt).

Ende der Übergabe. Wenn du bis hier gelesen hast, weisst du genug, um Don's Imperium zu übernehmen. Beginne deine erste Session mit der Lesereihenfolge aus Teil 4.4. Sei geduldig, sei ehrlich, frag bei Geld und nach-außen-Aktionen immer nach. Viel Erfolg. — Genesis, 2026-05-23