

DIE MASCHINE

Das MacBook ist das Gehäuse. Der Projektzustand ist das Produkt.

WarmStack wird nicht identisch ausgeliefert. Vor Versand bauen wir eine projektspezifische Entwicklungszelle: Repo, Agentenregeln, synthetische Daten, Datenmodell, Teststrecke und ein erster lauffähiger End-to-End-Ablauf. Aufklappen heißt nicht einrichten. Aufklappen heißt weiterbauen.

- 1

Repo als Source of Truth
- 2

unabhängige KI-Entwickler
- 0

offene Konfigurationsschritte
- 1

lauffähiges Musterstück

DER MASCHINENKERN

Base Layer + Project Cartridge

Ein stabiler Unterbau - und darüber genau der Stack, den das Vorhaben braucht.

BASE LAYER · BLEIBT

Arbeitsfähig ab Boot 1

- Claude Code
- OpenAI Codex
- Git + GitHub CLI
- Node/TypeScript + pnpm
- Node/TypeScript + pnpm
- Python + uv
- Playwright / Vitest / pytest
- Shell + IDE
- Shell + IDE
- Ollama optional

Reproduzierbarer Build, Projektbefehle, Tests, Git-Zustand, Kostenlimits und Sicherheitsgrenzen sind bereits gesetzt.

PROJECT CARTRIDGE · WECHSELT

Auf Ihr Vorhaben gebaut

- Repo oder Starter passend zum Zielsystem
- synthetische Daten in Ihrer späteren Form
- erster Vertical Slice: Import → Logik → UI
- CLAUDE.md / AGENTS.md mit Projektregeln
- Tests, Seeds, Reset- und Startbefehle
- Bauanleitung: was als Nächstes zu tun ist

```
$ warmstack prepare kundenprojekt
-> repo + runtimes + tools gesetzt
-> sample data + vertical slice gebaut
-> claude build / codex review / tests grün
-> bereit für Übergabe
```

Nicht jedes Logo ist auf jeder Maschine. WarmStack installiert nicht möglichst viel - sondern genau das, was den ersten echten Arbeitstag verkürzt.

MASCHINENMANIFEST · TYPISCHE BASIS

AGENTEN

Claude Code
OpenAI Codex

bauen · prüfen · rescue

RUNTIMES

Node / TypeScript
Python / uv

startbar ohne Setup

QUALITÄT

Git · Playwright
Vitest / pytest

Diff · Test · Rollback

CARTRIDGES

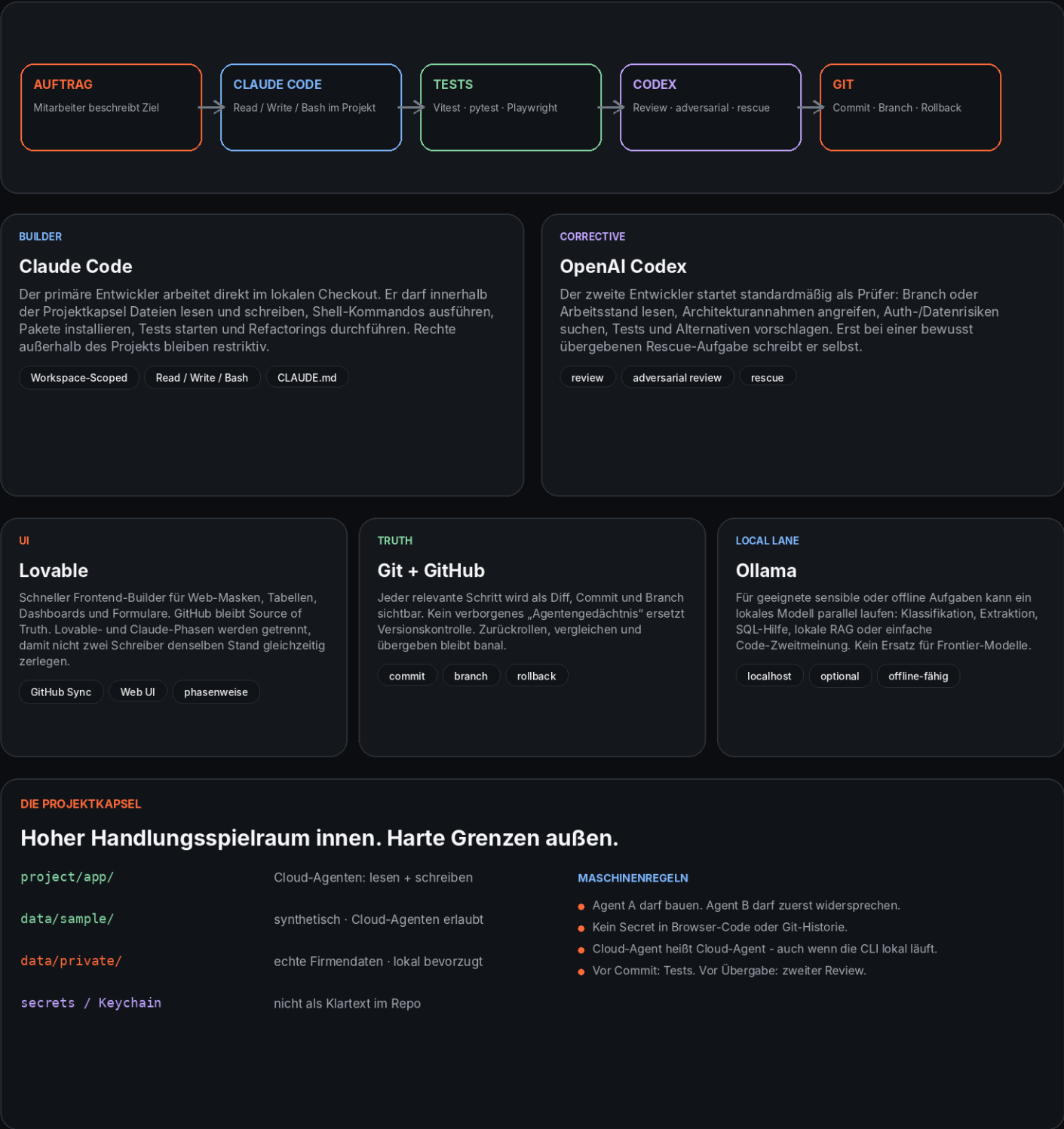
Lovable / Supabase
DuckDB / Expo / Ollama

nur wenn das Projekt sie braucht

DIE ENTWICKLER

Claude baut. Codex widerspricht.

Zwei Modelle sind nur dann besser als eines, wenn sie verschiedene Rollen haben. Auf WarmStack schreibt nicht jeder gleichzeitig überall. Der primäre Agent baut. Der zweite Agent prüft, greift Annahmen an und übernimmt nur gezielt.



PROJECT CARTRIDGES

Nicht jede Maschine ist gleich.

Ein Logistikprojekt braucht andere Werkzeuge als eine Fahrer-App. Ein Angebotsgenerator andere als ein Kundenportal. Deshalb wählen wir vor Versand den Stack, erzeugen synthetische Daten in der richtigen Form und bauen den ersten Vertical Slice.

01

Logistik / Daten

STACK

Python + uv · Polars · DuckDB · optional Postgres/Supabase · FastAPI oder TypeScript API · React/TanStack UI

DATENPFAD

CSV/XLSX → Importvalidator → Spaltenmapping → DuckDB/Postgres → API → Status-/Tourenansicht

BEREITS BEIM AUFKLAPPEN

150 synthetische Aufträge, 12 Fahrzeuge, fehlerhafte Testzeilen, Importer, Datenmodell, Fehlerprotokoll, erste Tourenübersicht und Tests.

02

Interne Web-App / Portal

STACK

React / TanStack Start oder bestehender Webstack · Lovable optional · Supabase/Postgres · Auth · RLS · Storage · Edge Functions · Zod · Playwright/Vitest

DATENPFAD

UI → Auth → API/DB → Row-Level-Regeln → serverseitige Integrationen; GitHub bleibt Source of Truth

BEREITS BEIM AUFKLAPPEN

Login, drei synthetische Kunden, Beispielaufträge, „nur eigene Daten“-Regel, Listen-/Detailsansicht, Seed- und Reset-Befehle, Browser-Test.

03

Mobile Fahrer-/Service-App

STACK

Expo + React Native + TypeScript · API-Client · lokale Queue/SQLite wenn Offline nötig · Kamera/Upload · Simulator/Device-Workflow

DATENPFAD

App → lokale Erfassung → Sync-Queue → API → Postgres/Storage; iOS-Buildpfad benötigt macOS/Xcode

BEREITS BEIM AUFKLAPPEN

20 synthetische Einsätze, Checklisten, Fotodoku-Demo, Offline-Fall, Sync-Status und ein startbarer Simulator-Build.

04

Dokument-/Angebotsautomation

STACK

Python · openpyxl · python-docx · PDF-Parser/Generator · regelbasierte Preislogik · optional lokales Modell für Klassifikation/Extraktion

DATENPFAD

CSV/Excel/Word/PDF → Parser → Regeln/Extraktion → Template → DOCX/PDF → Testvergleich

BEREITS BEIM AUFKLAPPEN

Ihre spätere Dateistruktur als synthetisches Muster: Angebotsvorlage erkannt, Staffelpreis-Regel eingebaut, Beispielangebot erzeugt, PDF-Ausgabe geprüft.

DIE AUSWAHL IST TEIL DES PRODUKTS

WarmStack verkauft nicht Lovable, Supabase, DuckDB oder Expo. WarmStack entscheidet, welche Kombination das beschriebene Vorhaben am schnellsten in einen wartbaren ersten Arbeitsstand bringt.

VOR DEM VERSAND

01

Projekt zerlegen

02

Stack wählen

03

Testdaten bauen

04

Vertical Slice

05

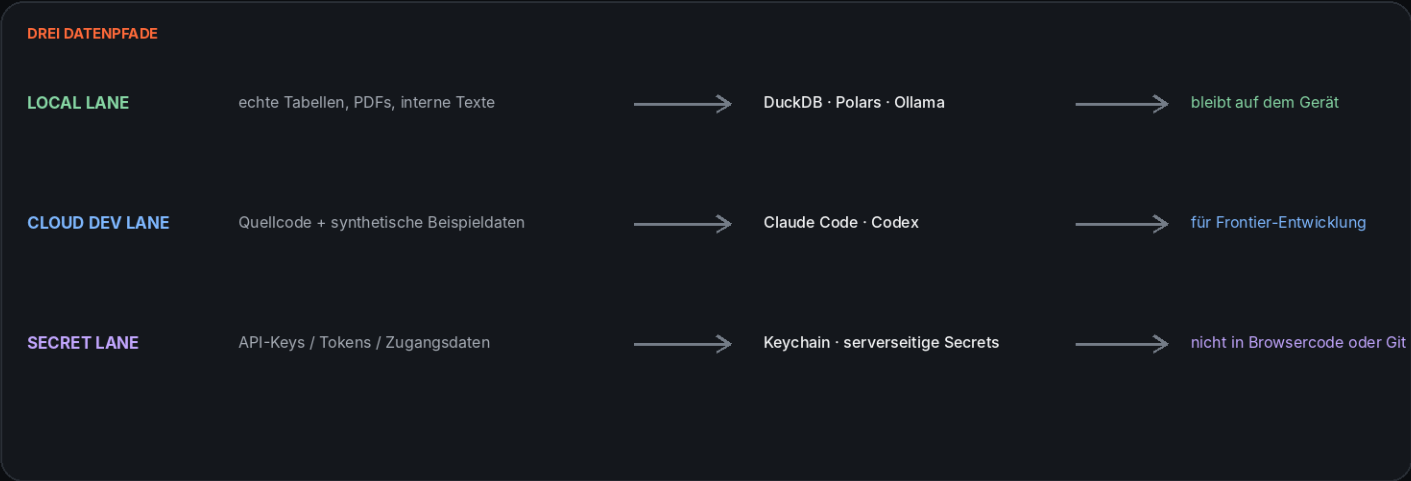
2. Agent Review

06

Kurier

Ihre Daten entscheiden, welcher Weg lokal bleibt.

WarmStack braucht für den Start keinen Zugang zu ERP, Servern oder Firmennetz. Vorinstalliert sind nur synthetische Daten. Echte Daten bringt der Kunde selbst auf das Gerät - und kann sensible Arbeit bewusst auf eine lokale Lane beschränken.



BETRIEB & SICHERHEIT

Was wir technisch fest verdrahten

- FileVault bzw. Geräteverschlüsselung aktiv.
- MDM nur für Geräte- und Umgebungsbetrieb; kein Remote-Screen als Produktfunktion.
- Keine echten Firmendaten, Logos oder ERP-Zugänge vorinstalliert.
- Git/Repo bleibt der nachvollziehbare Eigentums- und Übergabepunkt.
- Cloud-Agenten bekommen keine pauschalen Rechnerrechte, sondern Projektberechtigungen.
- Bei Rückgabe: kryptografische Löschung des Geräts und protokollierter Abschlussprozess.

HARDWARE NACH PROJEKT

Nicht „16 GB+“ für alles.

WEB / REPO	16-24 GB · 512 GB+
Claude/Codex, Webentwicklung, Tests, kleine lokale Modelle	
DATA + LOCAL AI	32-36 GB · 1 TB
DuckDB/Polars, lokale DB, gpt-oss:20b-Klasse parallel zu Browser/Devserver	
HEAVY LOCAL	48-64 GB+ · 1 TB+
größere quantisierte Coding-Modelle, Container, große Datenbestände parallel	

Mac: Unified Memory. Windows: System-RAM plus ggf. GPU-VRAM. Für native iOS-Entwicklung ist ein Mac die ehrliche Wahl.

WAS WARMSTACK NICHT BEHAUPTET

• Lokale Modelle sind nicht automatisch so gut wie Claude oder Codex. • Eine lokal laufende CLI macht einen Cloud-Agenten nicht lokal.	• Kein automatischer Zugriff auf ERP, Server oder Firmennetz. • Kein verborgenes Modell-Failover und keine magische Geräte-Synchronisation - Git ist die Wahrheit.
--	--

DER SATZ, DER BLEIBT

Das MacBook ist das Gehäuse.
Verkauft wird die komprimierte Einrichtung Ihres Projekts.

Beim Aufklappen läuft bereits etwas, das zu Ihrer echten Aufgabe passt - mit Beispieldaten, Tests, Git-Zustand und einer Bauanleitung für den nächsten Schritt.