

GitHub Copilot App: From Doing the Work to Directing It

Dr. Matthias Liebeck

Azure Düsseldorf Meetup · 06. Oktober 2026

Kurz zu mir

- Senior Solution Architect in Düsseldorf, Schwerpunkt .NET, C# und Azure
- Software-Entwickler seit 2009, C# seit 2006
- 2009 – 2015: Studium Informatik & Mathematik
- 2018: Promotion in Informatik über Natural Language Processing / KI
- Organisator des Azure Düsseldorf Meetups, Speaker über Azure & agentic coding
- GitHub Copilot Newsletter: ghcp.liebeck.io



Wer weiß, was agentic coding ist?

Du beschreibst das Ziel einer Aufgabe als Prompt in einem Chatfenster. Ein KI-Agent plant die Schritte, liest deinen Code, ändert Dateien, führt Builds und Tests aus und meldet sich, wenn er Feedback braucht. Nicht Autovervollständigung, sondern delegierte Arbeit.

GitHub Copilot GitHubs Coding-Agent – in IDE, CLI und eigener App

Claude Code Anthrops Coding-Agent – Terminal, IDE-Anbindung und Desktop

Codex OpenAIs Coding-Agent – Cloud, CLI und IDE-Extension

👉 Wer nutzt agentic coding?

👉 Und womit?

GitHub Copilot

Claude Code

Codex

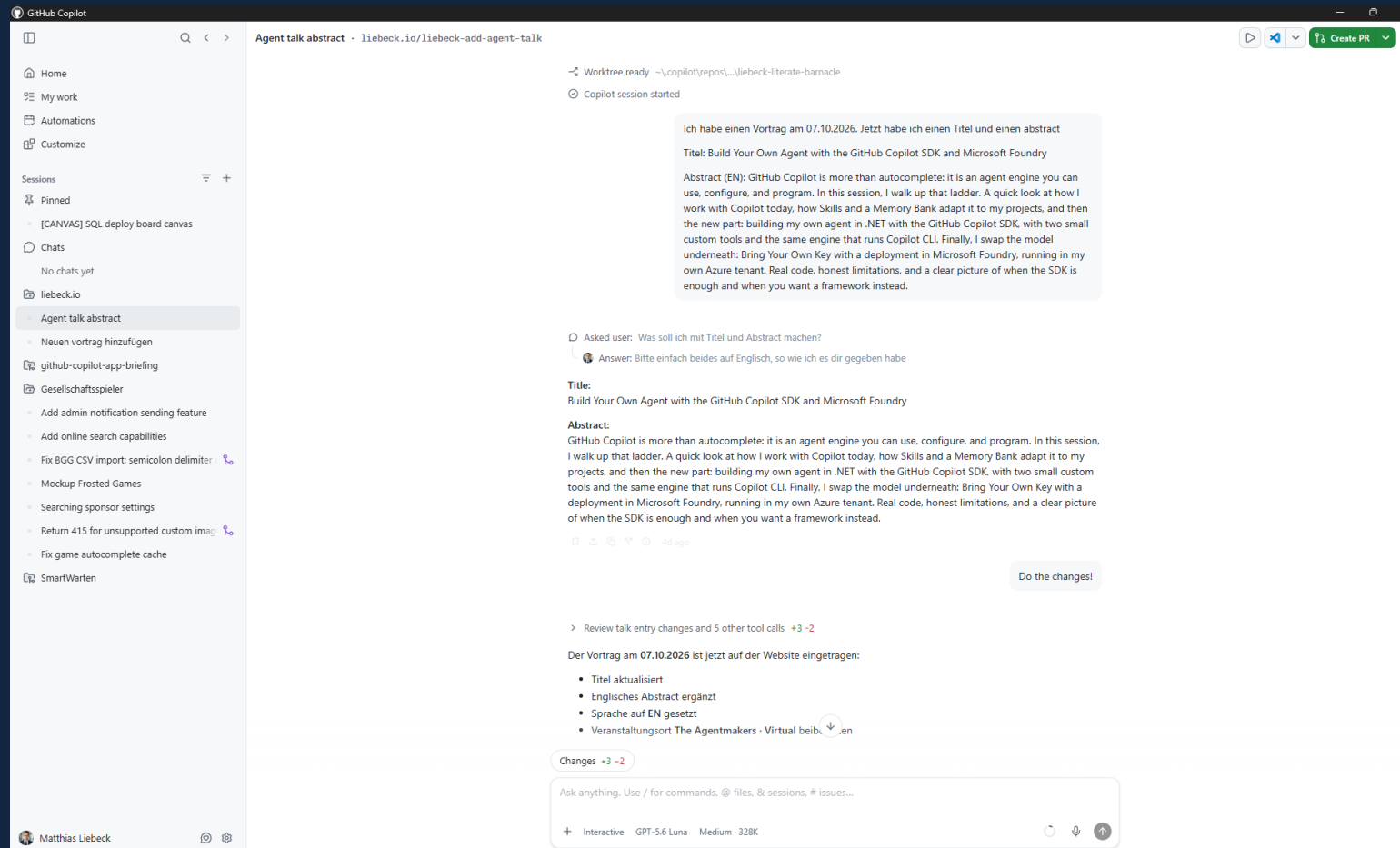
etwas anderes

Der Flugplan für heute

- 1 Einleitung
- 2 Cockpit Basics: Sessions, Workspaces, parallele Agenten
- 3 Canvases: Gemeinsame Arbeitsflächen für Mensch und Agent, mit Live-Demo
- 4 Bring Your Own Key
- 5 Automations
- 6 Fazit

Meine neue Entwicklungsumgebung

Die GitHub Copilot App ist mein Cockpit für die tägliche Arbeit geworden. Visual Studio öffne ich nur noch, um den Debugger zu starten.



Schnelle Entwicklungszyklen bei GitHub

- Technical Preview seit Mai 2026, General Availability seit dem 17. Juni 2026
- Neue Features erscheinen wöchentlich, das Changelog ist Pflichtlektüre
- Wenn heute Abend etwas anders aussieht als auf den Folien: Die Doku gewinnt

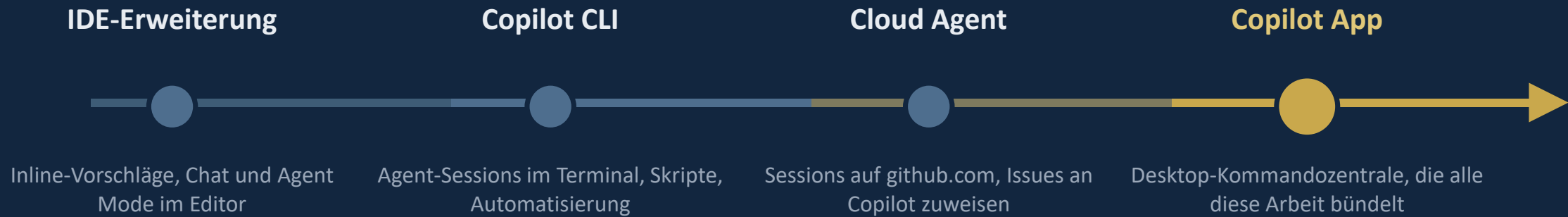
Öffentliches Repo für Issues, Feature-Wünsche und Diskussionen

– github.com/github/app

Wo die App im Copilot-Universum steht

Ich schreibe, Copilot assistiert

Ich dirigiere, Agenten arbeiten



- Auch der Agent Mode in VS Code arbeitet agentisch, aber in deinem einen offenen Workspace
- Die App ersetzt die anderen nicht: Sie dirigiert viele isolierte Sessions parallel, als Schaltzentrale darüber

Ein Desktop-Zuhause für Agenten-Arbeit

- Eigenständige Desktop-App für macOS, Windows und Linux
- Kein VS-Code-Fork, keine IDE-Erweiterung, kein Web-Wrapper um github.com
- Baut auf dem Copilot SDK auf: gleiche Runtime, gleiche Sessions, gleiche Skills wie Copilot CLI
- Native GitHub-Integration: Repos, Branches, Issues, PRs und CI-Checks funktionieren ohne zusätzliche Konfiguration
- Funktioniert mit jedem Copilot-Plan, sogar ganz ohne Plan per Bring Your Own Key
- Wer Claude Desktop mit Claude Code kennt: die vergleichbare Idee, aber nativ im GitHub-Ökosystem verankert

Du hörst auf, zuerst Code zu schreiben. Du fängst an, Arbeit zu dirigieren.



Cockpit Basics

◆ COCKPIT BASICS

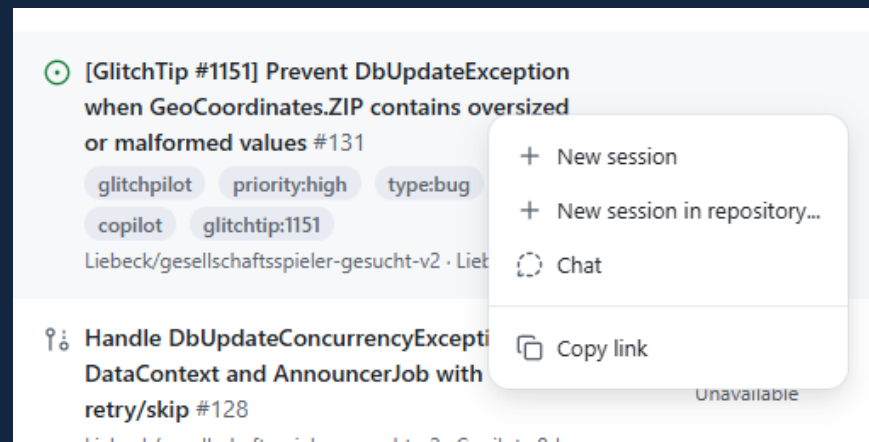
Ein Posteingang für alles, was in Bewegung ist

- My Work ist die Startansicht der App: alles Aktive auf einen Blick
- Laufende und pausierte Agent-Sessions, auch die aus der Copilot CLI
- Offene Issues und Pull Requests aus allen verbundenen Repositories
- Hintergrund-Automations und deren letzte Läufe
- Von hier startest du direkt die passende Session
- Kein Wechsel mehr zwischen Terminal, IDE, Browser-Tabs und github.com

The screenshot displays the Cockpit application interface. On the left, the 'My work' section shows a list of issues and pull requests. The first issue is 'Fix BGG CSV import: semicolon delimiter detection and correct header mapping #130' by Liebeck, with 32 additions and 6 deletions. Other issues include 'BoardGameGeek CSV import fails due to semicolon delimiter #129', 'Handle DbUpdateConcurrencyException in DataContext and AnnouncerJob with retry/skip #128', and 'Handle DbUpdateConcurrencyException in DataContext and AnnouncerJob with retry/skip #127'. The right panel shows the details of issue #130, including a description of the problem (BGG CSV exports use semicolons as delimiters), a list of changes (updated [Name] attributes, non-seekable stream support, auto-delimiter detection, header normalization, and structured error handling), and a code snippet for the C# implementation. The interface also shows reviewers (Liebeck), assignees (2 people), and labels.

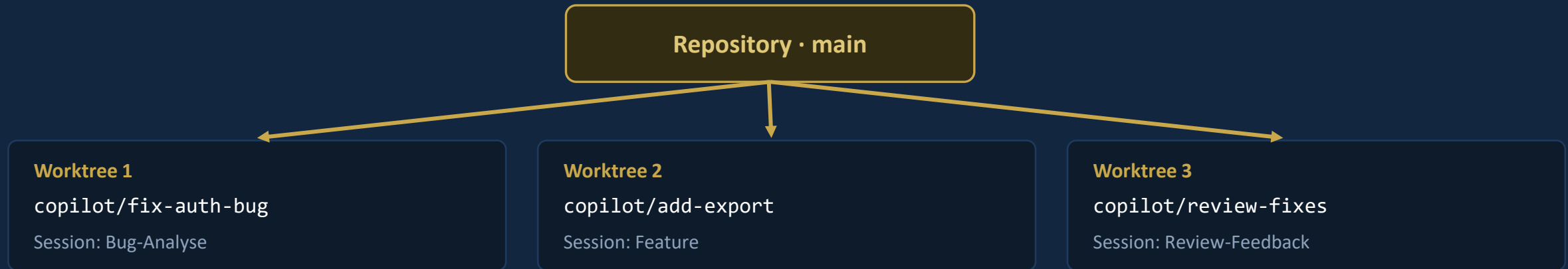
Sessions beginnen bei echter Arbeit

- Aus einem Issue: Titel, Beschreibung und Kommentare wandern in den Kontext
- Aus einem Pull Request in Arbeit: Review-Kommentare und Checks kommen mit
- Aus einem freien Prompt: leerer Workspace, Aufgabe in eigenen Worten
- Aus einer früheren Session: dort weitermachen, wo aufgehört wurde
- Der Kontext reist automatisch mit: Issue-Details, Repo-Zustand, Review-Kommentare, CI-Checks



Kein Kontext-Kopieren. Die Arbeit bringt ihren Kontext selbst mit.

Jede Session bekommt ihren eigenen Worktree



- Ein git worktree ist ein echter, isolierter zweiter Checkout des Repos in einem eigenen Ordner
- Parallele Sessions kommen sich nie in die Quere – die App legt Worktrees an und räumt sie wieder auf
- **Bei mir laufen bis zu 5 Sessions parallel – ich wechsele, sobald eine nach Feedback fragt**
- Dasselbe Prinzip gibt es in Claude (Code & Desktop) und bei Codex – wichtig ist, dass ihr wisst, dass es das gibt

Eine Session, ein Worktree, ein Branch. Das Jonglieren übernimmt die App.

Autonomie ist ein Regler, kein Schalter

Interactive

- Du und der Agent arbeiten Schritt für Schritt zusammen
- Jede Aktion ist sichtbar, du steuerst laufend nach

Plan

- Der Agent erstellt zuerst einen Plan
- Du prüfst und gibst frei, erst dann wird ausgeführt

Autopilot

- Der Agent arbeitet autonom bis zum Ergebnis
- Du prüfst das Resultat, nicht jeden Zwischenschritt

- Modus, Modell und Reasoning Effort werden pro Session gewählt
- Dasselbe Repo kann gleichzeitig eine Interactive- und eine Autopilot-Session haben

Nicht jede Aufgabe verdient denselben Grad an Vertrauen.

Sessions überleben deinen Arbeitstag

- Session verlassen und später genau dort weitermachen, wo du warst
- Branch, Dateien, Konversation und Task-Zustand bleiben erhalten
- **/chronicle: Fragen an die eigene Session-Historie stellen**
 - „In welcher Session habe ich das Auth-Modul umgebaut?“
- Die Historie umfasst auch Sessions aus CLI und Cloud Agent: eine gemeinsame Quelle der Wahrheit
- Sessions lassen sich optional mit dem Team teilen (nur Lesezugriff)

/chronicle View session history and activity [chronicle](#).

/chronicle tips Get personalized tips based on usage patterns.

/chronicle search Search all session content by keyword or topic.

/chronicle standup Get a report on your work from the last day.

/chronicle improve Suggest improvements to copilot-instructions.md.

/chronicle reindex Rebuild the [Chronicle](#) session index.

/chronicle cost-tips Get personalized tips to reduce token usage and cost.

/chronicle|

+ Interactive | Claude Sonnet 5 Medium

Lokal oder remote: wo deine Session lebt

Lokale Session

- Läuft auf deinem Rechner, im lokalen Worktree
- Volle Geschwindigkeit, deine Tools, dein Setup
- Pausiert, wenn der Rechner schläft oder aus ist
- Ideal für aktives Arbeiten mit Rückfragen

Remote Session (Cloud)

- Läuft in einer isolierten, von GitHub gehosteten Umgebung
- Arbeitet weiter, wenn du den PC ausmachst, auch über Nacht und übers Wochenende
- Von jedem Gerät wieder aufnehmbar, auch Tage später

Dein Rechner geht aus. Die Remote-Session nicht.

Deine Konventionen reisen mit

- Globale Instructions in den App-Einstellungen: gelten für jede Session
- Repository-Instructions pro Projekt: Coding-Standards, Architekturregeln
- Agent Skills: Ordner mit Anweisungen, Skripten und Ressourcen, die der Agent bei Bedarf lädt
 - Werden aus Repos und CLI-Setup automatisch synchronisiert
- MCP-Server: Repo-MCPs synchronisieren automatisch, eigene Server lokal oder per HTTP-Konfiguration
- Einmal konfiguriert, erbt jede neue Session das komplette Setup

Einmal einrichten, jede Session erbt es.



Demo: www.liebeck.io mit GitHub Copilot App updaten



Canvases

Fortschritt, begraben im Chat

- Je mehr ein Agent pro Session erledigt, desto mehr verschiebt sich deine Arbeit: Transkripte lesen, den relevanten Diff suchen, Kurskorrekturen wiederholen
- Chat ist gut, um Absichten zu klären und Mehrdeutigkeit auszudiskutieren
- Chat ist schlecht darin, den Zustand von Arbeit sichtbar zu machen
- Ein Plan mit 14 Schritten, von denen 9 erledigt sind: im Chat-Verlauf unsichtbar, auf einer Arbeitsfläche offensichtlich

Chat ist für Absichten. Arbeit braucht eine Fläche.

Eine gemeinsame Arbeitsfläche für Mensch und Agent

- Eine strukturierte, interaktive Fläche über einem Arbeitsobjekt
- Das Arbeitsobjekt kann vieles sein: ein Plan, ein PR, ein Terminal, eine Release-Checkliste, ein Migrations-Board, ein Incident
- Bidirektional: Der Agent aktualisiert das Canvas während der Arbeit, du editierst, sortierst um, gibst frei oder steuerst um, auf derselben Fläche
- Das Canvas ersetzt die Konversation nicht, es gibt ihr einen Ort zum Landen
- GitHub nennt das den Anfang von Agent Experience (AX): Oberflächen, auf denen Menschen und Agenten gemeinsam arbeiten

Der Agent zeigt seine Arbeit. Du steuerst sie auf derselben Fläche.

Canvas ist kein UI-Builder

Der erste Instinkt

- Dashboards für Endnutzer bauen
- Eine App im Canvas entwickeln
- Canvas als Ersatz für das eigene Frontend

Der eigentliche Zweck

- Laufende Agenten-Arbeit beobachten und steuern
- Zustand sichtbar machen: erledigt, läuft, hakt
- Entscheidungen treffen, wo die Arbeit passiert

- Ein Canvas ist eine Steuerfläche für Arbeit in Bewegung, kein Auslieferungskanal für fertige Software

/create-canvas: beschreiben statt bauen

- In einer Session /create-canvas eingeben und den Workflow beschreiben: Was sollen Menschen tun können, was der Agent
- Der Agent generiert daraus die Capabilities, also die Aktionen, die er selbst auf dem Canvas aufrufen kann
- Danach gemeinsam iterieren: Capabilities ergänzen, entfernen, umbauen
- Technisch: ein Extension-Verzeichnis mit package.json, extension.mjs für Verhalten und Capabilities, optionale JSON-Artefakte für Zustand

Project Scope

- .github/extensions: im Repo, mit dem Team geteilt
- Achtung: Jeder Worktree kopiert den State mit

User Scope

- ~/.copilot/extensions: eine Wahrheit über alle Worktrees
- Persönlich auf dem Rechner. Die Wahl fürs Demo-Board

Live: das SQL Deployment Board



Demo

- Der echte Schmerz: Migrations- und Translations-Skripte von Hand über drei Datenbanken (Stage, Docker, Live)
- Per /create-canvas entsteht ein Board: eine Karte pro Skript, Lanes als Deployment-Stationen
- Der Agent scannt master, Branches und Worktrees, fasst jedes Skript zusammen und warnt bei falscher Migrations-Reihenfolge
- Ausgeführt wird weiter von Hand: Ziehen ist Abhaken. Der Agent beobachtet, ich entscheide

Der Prompt: ein Absatz, ein Board

```
/create-canvas Create a canvas for tracking SQL script deployments in this repository. Install the extension at user scope. Scripts live in two folders: "scripts migration" (database migrations) and "scripts" (translation scripts). Only consider .sql files whose filename starts with yyyyMMdd, dated 2026-08-01 or later.
```

```
Board with five lanes: Not deployed yet, Stage, Docker, Live, Done.
```

```
Two visually distinct card types: migrations (white, blue accent) and translations (white, orange accent).  
Agent capability scan_scripts: search both folders on dev, on all local and remote branches, and in all git worktrees;  
add a card per new script with filename, type, a one-sentence summary derived from the script body, and an origin badge  
naming the branch or worktree if the script is not yet on master. Keep the returned payload compact, no full branch  
listings.
```

```
Humans drag cards between lanes; record a timestamp per station. Order rule for migrations only: warn (but still allow  
the move) if a migration is moved ahead while an older migration has not reached that lane; translations move freely.  
Cards that reached Live move to Done after seven days.
```

```
Store the board state in the user-scope extension directory, keyed by repository path so it survives reloads and new  
canvas instances.
```

```
The agent never executes SQL; execution is done manually.
```

Genau dieser Prompt läuft gleich live. Danach wird am Board weiter iteriert, per Zuruf.

Das fertige Board

SQL deployment board

C:\Users\liebeck\.copilot\repos\copilot-worktrees\github-copilot-canvas-sql-board\liebeck-symmetrical-invention

Scan
scripts

Older migration scripts
migration/20260901_AddContosoStoreCurbsidePickup.sql has not reached Docker.

Not deployed yet

**scripts
migration/20260901_Ad
dContosoStoreCurbsideP
ickup.sql**

migration

BEGIN TRANSACTION;.

**scripts/20260904_loyalty
-tier-translations.sql**

translation

DECLARE @TranslationKeys
TABLE (TranslationKey
NVARCHAR(MAX),
IndustryType INT);.

**scripts/20260905_checko
ut-error-messages.sql**

translation

DECLARE @TranslationKeys
TABLE (TranslationKey
NVARCHAR(MAX),
IndustryType INT);.

Stage

Docker

**scripts
migration/20260903_Ad
dContosoLoyaltyProgra
m.sql**

migration

BEGIN TRANSACTION;.

**scripts/20260902_curbsi
de-pickup-
translations.sql**

translation

DECLARE @TranslationKeys
TABLE (TranslationKey
NVARCHAR(MAX),
IndustryType INT);.

Live

Done

Canvases und Artifacts sind nicht dasselbe

Artifacts (Claude Desktop)

- Interaktive Flächen als Ausgabe des Modells
- Stark für Dokumente, Prototypen, Visualisierungen
- Richtung primär: Modell erzeugt, Mensch nutzt

Canvases (Copilot App)

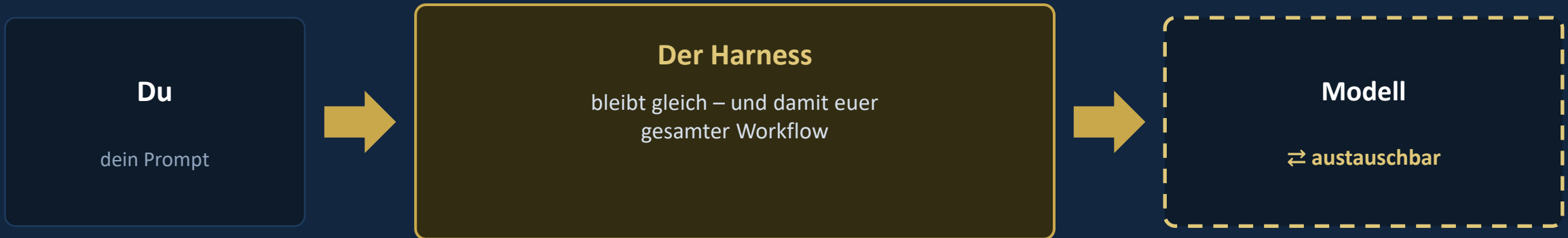
- Bidirektional: Agent und Mensch auf derselben Fläche
- Als Extension gebaut, mit agent-aufrufbaren Capabilities und persistiertem Zustand
- Die App verbindet Canvas und Artefakt und erzwingt, welche Aktionen erlaubt sind

- Vorsichtig formuliert, beide Produkte entwickeln sich schnell: Stand heute gibt es diese Form persistenter, bidirektionaler Arbeitsflächen mit Agent-Capabilities in Claude Desktop nicht



Bring Your Own Key

Warum überhaupt ein eigener Key bzw. Modell?



Der Baustein rechts ist tauschbar. Der in der Mitte bleibt – ihr lernt nichts Neues, ihr tauscht nur das Modell.

Abrechenbarkeit

Eigener Vertrag, eigene Kostenkontrolle –
Abrechnung direkt beim Anbieter

Vorhandene GPU nutzen

Lokale Modelle ohne Cloud – wenn der Speicher
mitspielt

Compliance & Standort

Nicht überall erlaubt: manche Teams dürfen nur
on-premise oder in Europa

Und gleich sehen wir: Es gibt fünf Wege dorthin.

Bring Your Own Key: mehrere Wege zum eigenen Modell

Weg 1: Anbieter direkt

- z. B. Claude per Anthropic API-Key
- Vertrag und Abrechnung beim Anbieter

Weg 2: Lokal

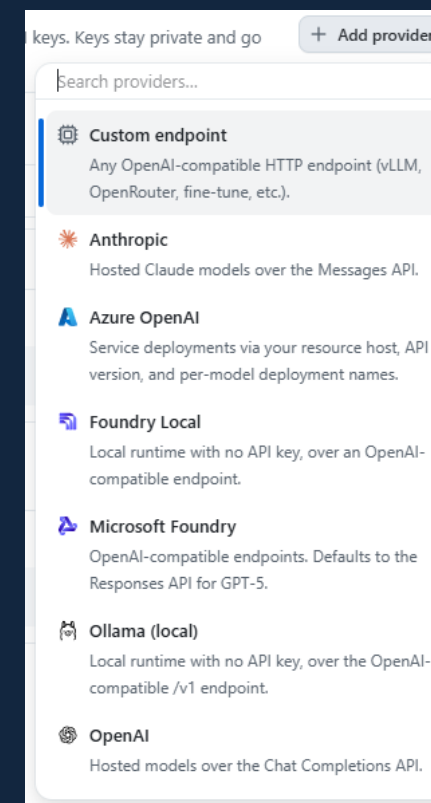
- Eigene Hardware, kein Anbieter im Spiel
- Der Haken kommt gleich: GPU-RAM

Weg 3: OpenRouter

- Ein Key, viele Anbieter und Modelle
- Der praktikable Mittelweg

Weg 4: Eigene Cloud

- Deployment in Microsoft Foundry
- Eigener Tenant, eigene Region



Das Harness bleibt gleich: Copilot orchestriert, nur das Modell dahinter wird getauscht.

Weg 2: lokales Modell

- Setup funktioniert: lokales Modell angebunden (Qwen über Ollama; alternativ Foundry Local)
- Realität: 8 GB VRAM reichen für agentische Kontextlängen nicht, Abstürze statt Antworten
- Die bindende Grenze ist der Speicher, nicht die Rechenleistung
- Gut, um die Mechanik zu zeigen. Für produktive Agenten-Arbeit nicht brauchbar
- Beim nächsten Rechner fließt mehr Geld in GPU-RAM. Falls die Preise je fallen sollten.

Weg 3: OpenRouter: ein Schlüssel, viele Modelle

- Aggregator mit einheitlicher API: ein Key, ein Guthaben, Zugriff auf Modelle vieler Anbieter
- Prepaid-Credits in US-Dollar: Kosten werden pro Request in Echtzeit abgebogen, leeres Guthaben stoppt die Calls
- Token-Preise werden 1:1 zum Listenpreis des Anbieters durchgereicht, Plattformgebühr 5,5 % beim Aufladen
- Activity-Seite zeigt Tokens und Kosten pro Request, Transparenz auf Lauf-Ebene
- In der App: als Custom Endpoint (OpenAI-kompatibel) mit Key hinterlegen, Modelle erscheinen in der Session-Auswahl
- Getestet mit qwen/qwen3-coder-next: echtes Feature auf dem eigenen Repo, stabiles Tool-Calling

Ein Guthaben, viele Modelle. Das leere Guthaben ist dein Kostenlimit.

Weg 3: OpenRouter: Risiken und die Schalter dagegen

Risiken

- Logging gegen 1 % Rabatt: Die FAQ nennt ihn noch, die Settings nicht mehr. Ablehnen
- Free-Endpoints dürfen teils speichern, trainieren oder Prompts in öffentliche Datensätze publizieren
- Zweite Verwaltungsgrenze: Prompts laufen über OpenRouter zum Provider
- Die Garantie hängt am Endpoint, nicht am Modell. Selbst erlebt: Qwen bestellt, Novita lieferte

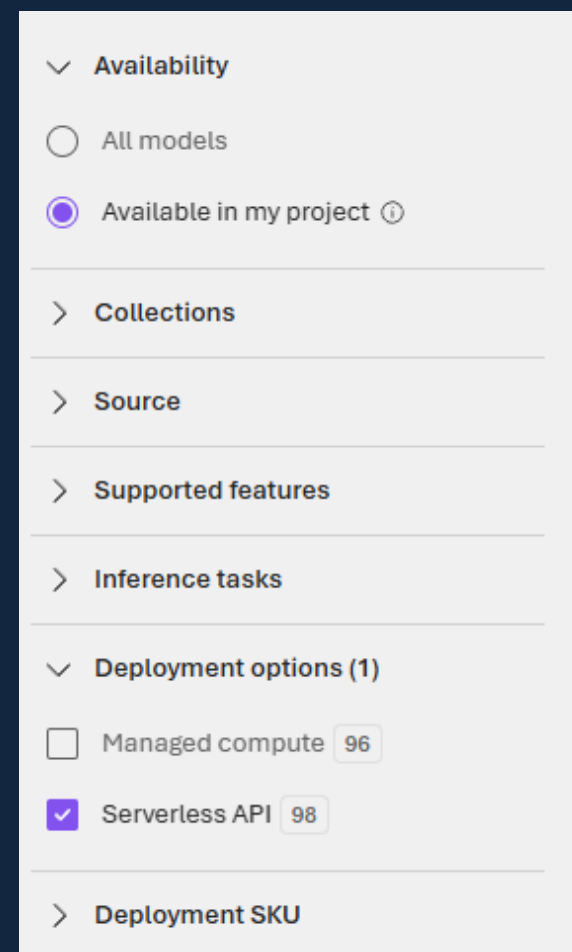
Die Schalter dagegen

- Achtung Default: Free-Endpoint-Training ist ab Werk AN. Vor dem ersten Request aktiv abschalten
- ZDR pro Anbieter-Familie schaltbar, plus Sammelschalter: alle Non-Frontier-Modelle nur über ZDR-Endpoints
- Governance: Drittanbieter-Modelle nur auf eigenen Repos, kein Kundencode

„Nicht trainiert“ und „nicht gespeichert“ sind zwei verschiedene Garantien.

Weg 4 ausprobiert: eigenes Modell in Microsoft Foundry

- DeepSeek-V4-Pro als Serverless-Deployment im eigenen Tenant, Endpoint plus Key in der App
- Wichtigster Filter im Katalog: Serverless API (pay per token). Managed Compute mietet GPUs pro Stunde
- Qwen gäbe es hier nur als GPU-Miete: Der Speicher bleibt die Grenze, auch wenn er Microsoft gehört
- Enterprise-Winkel: Daten bleiben in Tenant und Region, Abrechnung als eigener Azure-Verbrauch
- Getestet: echtes Feature auf dem eigenen Repo, stabiles Tool-Calling, gute Rückfragen





Automations

◆ AUTOMATIONS

Automations: wiederkehrende Arbeit, einmal definiert

- Skills und Prompts in wiederholbare Aufgaben verwandeln
- Ausführung auf Zeitplan oder auf Zuruf
- Ergebnisse tauchen in My Work auf, wie jede andere Session
- Zwei Beispiele laufen bei mir produktiv: das tägliche Release-Briefing und die wöchentliche Memory-Bank-Verification

The screenshot shows a 'New automation' form with the following fields and options:

- Name:** Copilot App Release-Briefing
- Trigger:** Daily (dropdown)
- Hours:** 09:00 (dropdown)
- Minute:** :00 (dropdown)
- Run in the cloud:** Toggle switch is turned on. Text below: 'Run even when your computer is off, triggered by events or on a schedule.'
- + Add another trigger**
- Tools:** Select which tools the agent can use. Built-in tools are always available. A dropdown menu shows 'All tools selected'.
- Prompt:** A text area containing the instruction: 'Check the releases and changelog of the public repository github/app (GitHub Copilot App). Read the most recent comment on issue #1 in this repository to see which release was reported last. If there are new releases since then, post one comment on issue #1 summarizing them: version, date, and the 3 to 5 most relevant changes for a practitioner, grouped by feature area. Be brief and factual. If there is nothing new, do not post a comment.'
- Model:** Autopilot | Claude Sonnet 5 | Medium (radio buttons)
- Automation Name:** github-copilot-app-briefing (dropdown)
- Buttons:** Cancel and Create (with a dropdown arrow).
- Footer:** Use Create and run from the menu to test your prompt right away.

◆ AUTOMATIONS

Beispiel: das tägliche Release-Briefing

- Jeden Morgen liest ein Agent Releases und Changelog von github/app
- Das Gedächtnis ist ein Issue: Der Agent liest seinen letzten Kommentar und berichtet nur Neues
- Ergebnis: ein Kommentar pro Tag mit Version, Datum und den relevanten Änderungen
- Läuft als Cloud Automation, der Rechner darf aus sein
- Das Briefing liegt morgens in My Work, ohne einen einzigen Klick vorher
- Einrichtungsaufwand: ein Prompt, ein Zeitplan, ein Testlauf

Your automations

Use agents to handle recurring work on a cadence you choose or triggered by events.

Automation Name	Frequency	Description	Owner	Status	Time
Memory Bank Verification (...)	Weekly	Run the "memory-bank-verification" skill located at skills/memory-bank-verificatio...	malask-git/smartwarten	✓	4h ago
Copilot App Release-Briefing	Daily	Check the releases and changelog of the public repository github/app (GitHub Cop...	Liebeck/github-copilot-app-...	✓	3h ago

Recent runs

Today

Automation Name	Frequency	Status	Time	Owner
Copilot App Release-Briefing		✓	9:08	Liebeck/github-copilot-app-briefing
Memory Bank Verification (weekly)		✓	7:41	malask-git/smartwarten
Copilot App Release-Briefing		✓	7:03	Liebeck/github-copilot-app-briefing

Skill wird Automation: Memory Bank Verification

- Der bestehende Agent Skill wird per / direkt in den Automation-Prompt eingebunden
- Wöchentlicher Lauf: Memory Bank gegen den aktuellen Code-Stand prüfen
- Befunde landen als Issue, nichts wird ungefragt geändert
- Erster Lauf: 10 echte Diskrepanzen in 5 Memory-Bank-Dateien gefunden
- Ausbaustufe: Der Agent darf den Drift später per Pull Request selbst beheben
- Einmal als Skill geschrieben, wiederverwendet als Automation

Loaded skill memory-bank-verification

Issue #4 "Memory bank drift" has been opened with 10 discrepancies found across 5 memory-bank files:

#	File	Area	Finding
1	techContext.md	EF Core packages	Version 10.0.3 → actual 10.0.10
2	techContext.md	Hangfire packages	Version 1.8.23 → actual 1.8.24
3	techContext.md	Playwright package	Version 1.58.0 → actual 1.61.0
4	techContext.md	Migrations note	"Recent" migrations are from May 2026; latest are 20260730...
5	prompt-suppression-feature.md	Migration status	Claims migration not generated; 20260708163356_AddPromptSuppression.cs exists
6	progress.md	Migration status	Same — "NOT YET GENERATED" note is stale
7	activeContext.md	Migration status	AddLocationTrailingTicketSampleSize marked "MIGRATION NOT YET GENERATED"; 20260705184939... exists
8	(missing)	New Location columns	ShowIssueCategoryLabel + LanguagePickerFlagsOnly (migration 20260730044259) undocumented
9	(missing)	New entity	LocationCustomHtml + CustomHtmlSlot (migration 20260730073554) undocumented
10	(missing)	Schema constraint	TenantTranslations.CultureKey narrowed to nvarchar(20) (migration 20260713074309) undocumented

◆ AUTOMATIONS

Läuft auch, wenn der Laptop zu ist

- Cloud Automations: Agenten laufen auf Zeitplan in der Cloud, unabhängig davon, ob dein Rechner wach ist
- Reagieren auf GitHub-Events, öffnen Issues, hinterlassen Kommentare
- Cloud Sessions: eine Agent-Session direkt aus der App in einer isolierten, von GitHub gehosteten Umgebung starten
- Von jedem Gerät wieder aufnehmbar, der Zustand liegt nicht auf deinem Rechner
- Isolation und Kontrolle: Umgebungen sind pro Session getrennt, Admins steuern die Nutzung über Copilot-Policies

Ehrlich: was Automations noch fehlt

Kosten sichtbar machen (#2696)

- Was hat ein einzelner Lauf an AI Credits gekostet?
- Heute nur über die Gesamtrechnung rekonstruierbar
- Wunsch: Credits pro Lauf, Verlauf und Schnitt pro Automation
- Erst damit lässt sich eine Frequenz seriös wählen

Kosten begrenzen (#2698)

- Unbeaufsichtigte Läufe brauchen ein Limit pro Lauf (--max-ai-credits)
- Gestoppte Läufe müssen als gestoppt erkennbar sein, nicht als fertig, dazu ein Monatsbudget pro Automation
- Bis dahin: Frequenz konservativ wählen und Läufe stichprobenartig prüfen

Reicht Issues auf github.com ein. Die Entwickler lesen viel Feedback und setzen schnell um.



Fazit

Ehrliche Grenzen, Stand Oktober 2026

-  Zum Programmieren (mit worktrees) sehr gut geeignet, hat IDE für mich ersetzt
- Einzelne Features ändern sich schnell, Muskelgedächtnis lohnt noch nicht
- Der Review-Aufwand verschwindet nicht, er verschiebt sich: Du schreibst weniger Code und prüfst mehr Ergebnisse
- Cloud Sandboxes sind noch Public Preview, nicht GA

Wohin die Reise geht

- Agent Experience (AX) steht am Anfang: Canvases sind der erste Schritt zu Oberflächen, die für Mensch und Agent gleichzeitig gebaut sind
- Die Produktkategorie hat inzwischen einen Namen: Agentic Development Environment (ADE). Die App ist GitHubs Vertreter darin
- Das SDK öffnet die Runtime für eigene Werkzeuge und Integrationen
- Partner-Agenten docken an das Ökosystem an
- Die Grenze zwischen lokal und Cloud verschwimmt weiter: Sessions starten lokal, laufen in der Cloud weiter, oder umgekehrt
- Meine Erwartung: Es kommt noch viel, wir stehen am Anfang

Drei Sätze zum Mitnehmen

Rückblick

Die Copilot App bündelt Sessions, Worktrees, PR-Lifecycle, Canvases und Automations in einer Desktop-Kommandozentrale

Realitätscheck

Vieles ist von Claude Desktop und Codex übernommen, und die token-basierte Abrechnung ist ein Nachteil. Mit günstigen BYOK-Modellen lässt er sich kompensieren

Kernbotschaft

Starkes Tool: Canvases sind einzigartig, und die native GitHub-Integration macht es zum ersten Cockpit, das im eigenen Ökosystem lebt.



Danke



Connect on LinkedIn

linkedin.com/in/matthias-liebeck



GitHub Copilot Newsletter

ghcp.liebeck.io