



Lass Claude deine Lösung angreifen.

Wenn Claude an einem komplexen Projekt festhängt, kann Ultracode die Arbeit auf Agenten verteilen. Du gibst ihnen einen klaren Auftrag: finden, widerlegen, verbessern.

WAS DABEI PASSIERT



Untersuchen

Ursache und Randfälle finden.



Gegenprüfen

Befunde mit Tests angreifen.



Verbessern

Belegte Fehler beheben.

Ultracode gehört zu Claude Code. Agenten bearbeiten Teilaufgaben; ihre Befunde werden zusammengeführt. Das kann bei großen Aufgaben viele Agenten umfassen. Eine feste Zahl von Hunderten ist keine Voraussetzung.

DREI AUFTRÄGE FÜR DEIN PROJEKT

01 Einen wiederkehrenden Fehler reparieren.

02 Eine Website vor dem Launch prüfen.

03 Eine App um einen Statusfilter erweitern.

Jeder Auftrag ist vollständig kopierbar. Ersetze nur die eckigen Klammern und starte im passenden Projektordner.

[Claude Code öffnen / Produkt ansehen ↗](#)

[So funktionieren Dynamic Workflows ↗](#)



In deinem Projekt starten.

Du brauchst deine vorhandenen Website- oder App-Dateien und Claude-Code-Zugang. Dynamic Workflows sind auf bezahlten Plänen verfügbar; bei Pro aktivierst du sie in `/config` → Dynamic workflows. API-Nutzung wird separat abgerechnet.

01 / CLAUDE CODE EINRICHTEN

Noch nicht installiert? Öffne die offizielle Einrichtung und nutze den Befehl für dein System. Starte danach ein neues Terminal und prüfe mit `claude --version`. Melde dich beim ersten Start an.

[Offizielle Installation für Windows und Mac ↗](#)

02 / DEN RICHTIGEN ORDNER ÖFFNEN

Öffne im Terminal deinen Projektordner. Ersetze den Beispielpfad durch den Ordner mit deinen Dateien. Tippe anschließend:

```
cd "Pfad/zu/deinem/Projekt"  
claude
```

03 / ULTRACODE AKTIVIEREN

In der gestarteten Claude-Code-Sitzung:

```
/effort ultracode
```

Kopiere nun einen Auftrag von den nächsten Seiten. Fehlt Ultracode, prüfe Updates, Dynamic workflows und ein Modell mit xhigh-Unterstützung in `/model`.

Mit Version 2.1.284 oder neuer ausschalten: `/effort ultracode off`. Ultracode lässt sich dort getrennt vom Denkaufwand umschalten. Größere Läufe verbrauchen mehr Kontingent.

[Ultracode und aktuelle Bedienung ↗](#)



Den Fehler an der Wurzel packen.

Öffne das betroffene Website- oder App-Projekt. Ergänze im Prompt die genaue Klickfolge und den Fehlertext. Ohne diese Angaben bleibt die Ursache schwer prüfbar.

KOPIEREN / KLAMMERN ERSETZEN

Du bist mein Debugging-Team. Du arbeitest im geöffneten Projektordner. Nutze ultracode für diesen Auftrag.

Fehler: [genauer Fehlertext]. Reproduktion: [Klickfolge]. Erwartet: [richtiges Verhalten]. Bisher versucht: [Änderung oder keiner].

Reproduziere den Fehler zuerst. Lass Agenten Ursache, Zustandswechsel und Randfälle unabhängig untersuchen. Ein Gegenprüfer soll die stärkste Erklärung mit einem Gegenbeispiel oder Test zu widerlegen versuchen. Repariere nur die belegte Ursache; koordiniere Schreibzugriffe, damit Änderungen nicht kollidieren.

Erhalte alle übrigen Funktionen. Keine neuen Dienste, kein Deployment und keine Änderungen an Zugangsdaten. Führe die passenden vorhandenen Tests aus und ergänze einen Regressionstest, der vor der Reparatur scheitert und danach besteht. Teste auch den zweiten Versuch nach einer Fehlermeldung.

Korrigiere neue Fehler und prüfe erneut. Beende nach zwei Runden ohne Fortschritt mit dem konkreten Blocker. Prüfe aus Sicht eines unbeteiligten Nutzers, ob der Ablauf verständlich ist; kennzeichne dies als simulierte Perspektive. Liefere Ursache, geänderte Dateien, ausgeführte Tests mit Ergebnis und offene Punkte.



Finde, was Besucher aufhält.

Öffne die Website-Dateien. Ersetze die zwei Angaben im Prompt: dein Angebot und die gewünschte Besucheraktion. Die Dateien und Downloads müssen im Projekt vorliegen.

KOPIEREN / KLAMMERN ERSETZEN

Du bist mein Team für die Launch-Prüfung. Im geöffneten Ordner liegt meine Website. Nutze ultracode. Angebot: [konkretes Angebot]. Zielaktion: [z. B. PDF herunterladen].

Starte eine lokale Vorschau. Teile die Prüfung auf: Besucherweg und Verständlichkeit, mobile Darstellung, Links und Downloads, Tastaturbedienung. Lass einen Gegenprüfer jeden kritischen Befund am tatsächlichen Verhalten widerlegen oder bestätigen. Priorisiere Probleme, die die Zielaktion verhindern.

Korrigiere bestätigte Fehler und prüfe die betroffenen Wege erneut. Erhalte meine Gestaltung, Texte und bestehenden Funktionen, sofern ein belegter Fehler keine Änderung erfordert. Keine erfundenen Testimonials, keine neuen Trackingdienste und keine Veröffentlichung.

Teste bei 390 und 1440 Pixel Breite den Weg vom Einstieg bis zur Zielaktion, auch mit Tastatur. Öffne den heruntergeladenen Inhalt und kontrolliere, ob er zum Button passt. Nutze die vorhandenen Browserwerkzeuge; fehlen sie, melde die entsprechenden Prüfungen als offen.

Prüfe kritisch aus Sicht eines erstmaligen Besuchers, ob Nutzen und nächster Schritt klar sind; dies ist eine simulierte Perspektive. Korrigiere Abweichungen. Liefere bestätigte Befunde, Korrekturen, Testresultate und ungetestete Punkte. Stoppe nach zwei Runden ohne Fortschritt mit dem Blocker.



Neue Funktion. Alte Abläufe bleiben.

Öffne deine App mit Anfrageübersicht. Passe die Statuswerte offen, in Arbeit und erledigt im Prompt an deine App an.

KOPIEREN / KLAMMERN ERSETZEN

Du bist mein Entwicklungsteam. Du arbeitest im geöffneten Projekt mit meiner Kundenanfragen-App. Nutze ultracode.

Ergänze in der Anfrageübersicht einen Statusfilter für offen, in Arbeit und erledigt sowie Alle. Nutze die vorhandenen Statusfelder. Fehlen sie oder heißen sie anders, kläre die Zuordnung vor Änderungen.

Lass Agenten Datenfluss, Oberfläche und bisherige Tests unabhängig prüfen. Ein Gegenprüfer soll den Plan auf gebrochene bestehende Abläufe untersuchen. Setze danach die kleinste passende Lösung um; nur ein koordinierter Schreibzugriff pro betroffener Datei.

Der Filter muss mit der vorhandenen Suche kombinierbar sein. Alle zeigt alle zur Suche passenden Einträge. Bei null Treffern erscheint ein verständlicher Hinweis. Bestehende Bearbeitung und gespeicherte Daten bleiben erhalten. Keine Datenmigration, keine neuen Dienste und kein Deployment.

Nutze fiktive Testeinträge mit allen drei Statuswerten. Teste jeden Filter, die Kombination mit Suche, null Treffer und Zurücksetzen auf Alle. Führe die passenden vorhandenen Tests aus. Prüfe auch, ob ein Eintrag weiterhin bearbeitet werden kann.

Behebe Fehler und teste erneut. Prüfe aus Sicht eines neuen Nutzers, ob Filter und Ergebnis verständlich sind; kennzeichne die Perspektive als simuliert. Liefere Änderungen, echte Testresultate und offene Punkte. Stoppe nach zwei Runden ohne Fortschritt mit dem konkreten Blocker.



Wann du wirklich fertig bist.

DIE AGENTEN SIND FERTIG. JETZT KOMMT DEIN CHECK.

Öffne das Ergebnis selbst. Wiederhole die fehlerhafte Klickfolge, lade die PDF über deinen Website-Button herunter oder kombiniere Filter und Suche. Der Auftrag ist erst erfüllt, wenn dieser konkrete Weg funktioniert.

WAS IM ABSCHLUSSBERICHT STEHEN SOLL

Belegt: Ursache oder Fehler mit Datei, Klickfolge oder Test.

Geändert: betroffene Dateien und der Grund.

Getestet: tatsächlich ausgeführter Test und Ergebnis.

Offen: nicht geprüfte Geräte, Dienste oder Zugänge.

WENN NOCH ETWAS FALSCH IST

Antworte konkret: „Beim zweiten Absenden bleibt der Button gesperrt. Prüfe diese Klickfolge, korrigiere die Ursache und wiederhole den Test.“ Ein beobachteter Fehler hilft mehr als „Mach es besser“.

DEIN KONTINGENT IM BLICK

Nutze Ultracode für komplexe Aufgaben mit mehreren prüfbaren Teilproblemen. Prüfe vor und nach einem Lauf /usage. Stoppe einen ausufernden Lauf und verkleinere den Auftrag. Für Routineänderungen schalte Ultracode wieder aus.

[Nutzung und Kosten verstehen ↗](#)

Quellen und Bedienwege am 01.10.2026 geprüft. Die drei Aufträge sind eigene Praxisvorlagen; sie wurden nicht in Claude Code ausgeführt. Eine Perspektivprüfung durch KI ist kein echter Nutzertest.

[Mehr praktische KI-Anleitungen von Willi ↗](#)