

Erst fragen, dann bauen: mein *Grill Me*

Mein Grill-Me-Skill zum Kopieren, wie du ihn bei jedem neuen Projekt in Claude, Codex, Copilot, Cursor oder einem anderen Chat startest, und mein Ablauf vom ersten Gespräch bis zum passenden Skill.

Kevin Coo

cooexistenz.de

Danke fürs Folgen ✓ · Du hast kommentiert: GRILL ✓

© 2026 cooexistenz · Kevin Coo · cooexistenz.de · Persönliches Exemplar, Weitergabe und Verkauf nicht gestattet · ID CX-20261102-AB9F6053

// 01

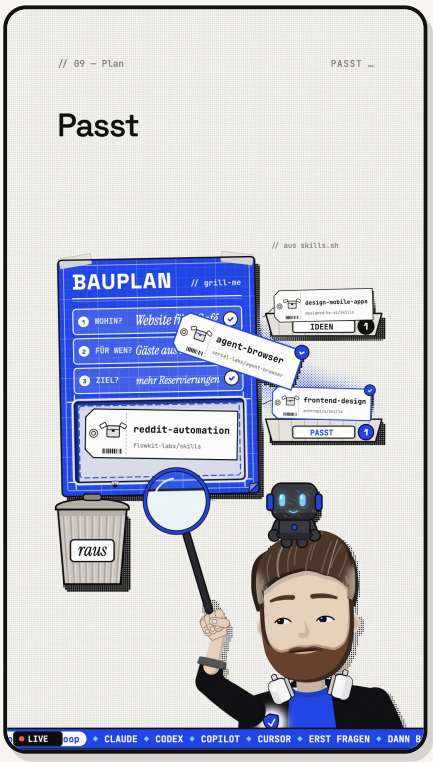
Auf einen Blick

FRAGE	KURZ GESAGT
Was ist das?	Ein Skill, der dich zu deinem Plan ausfragt, bevor die KI irgendwas baut. Ein Skill ist eine Textdatei mit Anleitung (`SKILL.md`), die deine KI bei Bedarf liest.
Von wem?	Original von Matt Pocock (Sammlung mattpocock/skills, MIT-Lizenz). Meine Version ist eine eigene deutsche Bearbeitung.
Wie beliebt?	Platz 2 auf skills.sh nach Installationen, rund 1,5 Mio. (Stand 02.11.2026).
Was kostet es?	Nichts. Es ist Text. Du brauchst nur deinen KI-Zugang.
Wo läuft es?	Claude Code, claude.ai, Codex, GitHub Copilot, Cursor. Überall sonst als erster Prompt.
Was kommt raus?	Klare Entscheidungen statt Raten. Meine Version schreibt sie in eine Datei `decisions.md`.

// AUS DEM VIDEO







© 2026 cooexistenz · Kevin Coo · cooexistenz.de · Persönliches Exemplar, Weitergabe und Verkauf nicht gestattet · ID CX-20261102-AB9F6053

// 02

Was das Video nicht sagt

„Zweitbeliebtester Skill der Welt“ heißt genau: Platz 2 in der Rangliste „All Time“ auf skills.sh. Die Rangliste zählt nur Installationen über das Kommandozeilen-Programm `npx skills`. Wer Grill Me als Plugin oder per Upload nutzt, wird nicht mitgezählt. Platz 1 ist find-skills, ein Skill zum Finden von Skills.

Das Original stellt mehrere Fragen auf einmal. Matts `grill-me` ist nur ein Startknopf. Die eigentliche Arbeit macht ein zweiter Skill namens `grilling`. Der fragt in Runden: alle Fragen, die gerade beantwortbar sind, nummeriert, jede mit einem Vorschlag. Sagst du „ja“, nimmst du den Vorschlag an. Fakten sucht die KI selbst, die Entscheidungen bleiben bei dir. Das Original schreibt keine Datei.

Meine Version ist anders gebaut. Sie stellt immer nur eine Frage, gibt nach deiner Antwort ihre eigene Empfehlung, klopft feste Winkel ab (Annahmen, Risiken, was rein und was raus kommt) und schreibt am Ende alles in `decisions.md`. Das brauche ich für Schritt 3 meines Ablaufs: Jede Skill-Idee lege ich gegen diese Datei.

Es gibt viele Skills, die „grill-me“ heißen. Auf skills.sh findest du mindestens fünf weitere mit genau diesem Namen, von anderen Leuten. Achte beim Installieren auf den Absender `mattpocock/skills` oder nimm meine Version unten.

Grill Me ist nicht nur für Code. Matt schreibt selbst, dass Leute ihn für Geschäftsentscheidungen, Texte und die Frage „Was mache ich als Nächstes?“ nutzen.



DER MENTOR SAGT

Die KI ist nur so schlau wie das, was du ihr über deinen Plan sagst. Grill Me sorgt dafür, dass du es ihr sagst.

© 2026 cooexistenz · Kevin Coo · cooexistenz.de · Persönliches Exemplar, Weitergabe und Verkauf nicht gestattet · ID CX-20261102-AB9F6053

// 03

Mein Grill-Me-Skill zum Kopieren

Das ist meine Version, komplett. Kopier beide Kästen nacheinander in eine Datei mit dem Namen `SKILL.md` und leg sie in einen Ordner `grill-me`. Hinweis: In der Vorlage für `decisions.md` stehen hier `~~~` statt drei Backticks, das funktioniert in Markdown genauso.

~~~
---
name: grill-me
description: Befrage den Nutzer schonungslos zu jedem Aspekt eines Plans oder
Designs, bis ein echtes gemeinsames Verständnis erreicht ist, und löse dabei jeden
Ast des Entscheidungsbaums einzeln auf. Einsetzen, wenn der Nutzer einen Plan oder
ein Design auf Schwachstellen prüfen, blinde Flecken vor dem Commit finden will oder
"grill me" bzw. "grill mich" sagt.
---

# Grill Me

Befrage mich schonungslos zu jedem Aspekt dieses Plans oder Designs, bis wir ein
echtes gemeinsames Verständnis erreicht haben. Deine Aufgabe ist nicht, den Plan zu
bestätigen – sondern die Schwachstellen zu finden, bevor die Realität es tut.

## Bevor du startest

1. Lies den Plan, das Design-Dokument oder das Briefing vollständig. Wenn es im Code
lebt, erkunde die Codebase, um zu verstehen, wie die Dinge heute tatsächlich
funktionieren.
2. Bau dir einen Entscheidungsbaum: Liste die Entscheidungen auf, die der Plan trifft
(explizit oder implizit), die offenen Fragen, die er offen lässt, und die
Abhängigkeiten dazwischen.
3. Ordne die Fragen so, dass Entscheidungen, die andere freischalten, zuerst kommen.
Spring nicht zufällig hin und her.

## Wie du grillst

- Stelle **immer nur eine Frage auf einmal**. Warte auf meine Antwort, bevor du
weitergehst.
- Geh jeden Ast des Entscheidungsbaums durch und löse die Abhängigkeiten zwischen den
Entscheidungen einzeln auf. Wenn meine Antwort auf eine Frage einen späteren Ast
verändert, aktualisiere den Baum und sag mir das.
- Gib zu jeder Frage, nachdem ich geantwortet habe (oder wenn ich unsicher bin),
**deine eigene empfohlene Antwort** mit kurzer Begründung. Verhör mich nicht nur –
beziehe selbst Position.
- Wenn eine Frage durch Erkundung der Codebase statt durch Nachfragen bei mir
beantwortet werden kann, erkunde die Codebase und beantworte sie selbst. Lass mich
nichts nachschlagen, was du selbst finden kannst.
- Akzeptiere keine vagen oder schwammigen Antworten. Wenn ich sage "das machen wir
später" oder "das läuft schon", hak nach und zwing mich, konkret zu werden.
~~~

Fortsetzung, gehört in dieselbe Datei direkt darunter:

© 2026 cooexistenz · Kevin Coo · cooexistenz.de · Persönliches Exemplar, Weitergabe und Verkauf nicht gestattet · ID CX-20261102-AB9F6053

```
## Worauf du abklopfst

Geh jeden dieser Winkel durch, nicht nur den Happy Path:

- **Annahmen** — was setzt der Plan als selbstverständlich voraus? Welche Annahmen sprengen alles, wenn sie falsch sind?
- **Edge Cases & Failure Modes** — was passiert bei leerem, riesigem, fehlerhaftem, nebenläufigem oder bösartigem Input? Was passiert, wenn eine Abhängigkeit ausfällt?
- **Tradeoffs** — worauf wird optimiert, und was wird dafür geopfert? War dieser Tausch Absicht oder Zufall?
- **Scope & Reihenfolge** — was ist drin, was draußen, was ist die kleinste Version, die Wert liefert, und was lässt sich verschieben?
- **Umkehrbarkeit** — welche Entscheidungen sind Einbahnstraßen? Die verdienen die härteste Prüfung.
- **Betreibbarkeit** — wie wird das nach dem Launch beobachtet, debuggt, zurückgerollt und gewartet?

## Ton

Sei direkt und skeptisch, aber in guter Absicht. Du bist der harte Reviewer, der will, dass der Plan gelingt — nicht der Querulant. Wenn etwas wirklich solide ist, sag es kurz und geh weiter — erfinde keine künstlichen Einwände.

## Wann du aufhörst

Mach weiter, bis jeder Ast des Entscheidungsbaums aufgelöst oder bewusst mit dokumentierter Begründung geparkt ist. Wenn nichts mehr übrig ist, das den Plan noch sinnvoll verändern könnte, hör auf und erstelle eine **Zusammenfassung**.

## decisions.md schreiben

Wenn die Befragung abgeschlossen ist, schreibe die Zusammenfassung in eine Datei `decisions.md` im Arbeitsverzeichnis (oder neben dem Plan-Dokument, falls vorhanden). Bei einer bestehenden `decisions.md` hänge einen neuen datierten Abschnitt an, statt sie zu überschreiben. Struktur:

~~~markdown
# Entscheidungen — <Plan-Name> (<YYYY-MM-DD>)

## Getroffene Entscheidungen
- Jede aufgelöste Frage und die Antwort, auf die wir uns geeinigt haben, mit kurzer Begründung.

## Offene Punkte
- Alles Verschobene, mit Begründung warum und wann es wieder aufgegriffen werden muss.

## Risiken
- Die wichtigsten verbleibenden Risiken und ihre Mitigationen.

## Änderungen am ursprünglichen Plan
- Was sich infolge dieses Gesprächs verschoben hat.
```

© 2026 cooexistenz · Kevin Coo · cooexistenz.de · Persönliches Exemplar, Weitergabe und Verkauf nicht gestattet · ID CX-20261102-AB9F6053

```
~~~

Zeig mir die Zusammenfassung auch direkt im Chat, nicht nur in der Datei.
```

Grundlage: grill-me von Matt Pocock, mattpocock/skills, MIT-Lizenz, Copyright (c) 2026 Matt Pocock. Meine Version ist eine eigene deutsche Bearbeitung, keine offizielle Übersetzung.

// 04

So installierst du es

Wichtig: Der Ordner heißt genauso wie der Name im Kopf der Datei, also `grill-me`, und darin liegt `SKILL.md`. Jedes Programm braucht seine eigene Kopie, nichts wird automatisch abgeglichen.

Claude Code (Terminal oder Desktop-App)

1. Leg den Ordner an: `~/.claude/skills/grill-me/SKILL.md` (für alle Projekte) oder `.claude/skills/grill-me/SKILL.md` (nur für ein Projekt).
2. Starte eine neue Sitzung und tipp `/grill-me`, dahinter deinen Plan in ein paar Sätzen.
3. Prüfen: `/skills` zeigt dir, ob er geladen ist.

claude.ai im Browser oder in der Claude-App

1. Einmalig: Settings > Capabilities > „Code execution and file creation“ einschalten (im neuen Einstellungs-Layout heißt es „Create and edit files“). Ohne das laufen keine Skills.
2. Den Ordner `grill-me` (mit `SKILL.md` darin) als ZIP packen.
3. Customize > Skills > „+“ > „+ Create skill“ > „Upload a skill“, ZIP hochladen, Schalter an.
4. Starten: Schreib „Grill mich zu diesem Plan: …“. Einen Slash-Befehl gibt es im Chat laut Hilfe nicht.

Das geht in allen Tarifen, auch im kostenlosen: Die Anthropic-Hilfe nennt ausdrücklich „Free, Pro, Max, Ultra, Team und Enterprise“ (eine ältere Entwickler-Seite nennt Free noch nicht). Die Zusammenfassung steht immer im Chat; bei eingeschalteter Dateierstellung kann Claude sie zusätzlich als Dokument anlegen, das du direkt aus dem Chat herunterlädst. Matts Original lässt sich hier nicht hochladen: claude.ai erlaubt im Kopf der Datei nur sechs Felder (name, description, license, compatibility, metadata, allowed-tools), jedes andere Feld bricht den Upload mit einer Fehlermeldung ab, und Matts Datei enthält `disable-model-invocation`. Meine Version hat nur `name` und `description` und passt.

© 2026 cooexistenz · Kevin Coo · cooexistenz.de · Persönliches Exemplar, Weitergabe und Verkauf nicht gestattet · ID CX-20261102-AB9F6053

Codex, Copilot, Cursor

TOOL	WOHIN MIT DEM ORDNER	SO STARTEST DU IHN
Codex	`~/.agents/skills/grill-me/` oder im Projekt `.agents/skills/grill-me/`	`/skills` oder `$grill-me` tippen
ChatGPT	Eigene Skills nur in der Desktop-App (Seitenleiste „Skills“); im Browser und auf dem Handy nur als Teil eines Plugins	Am einfachsten: Text in die Projekt-Anweisungen oder als ersten Prompt
GitHub Copilot	`~/.copilot/skills/grill-me/` oder im Projekt `.github/skills/grill-me/`	Copilot wählt selbst nach der Beschreibung: schreib „grill mich zu …“
Cursor	`~/.cursor/skills/grill-me/` (liest auch `~/.claude/skills/`)	`/` im Agent-Chat, dann `grill-me`

Lieber Matts Original? In Claude Code als Plugin, es aktualisiert sich selbst:

```
claude plugin install mattpocock-skills@claude-plugins-official
```

Oder für jedes Tool über skills.sh (braucht Node.js 22.20 oder neuer):

```
npx skills add https://github.com/mattpocock/skills --skill grill-me
```

Matt warnt: Nimm pro Tool nur einen Weg, sonst hast du jeden Skill doppelt. Und wenn du meine und seine Version zusammen nutzt, nenn meine um (Ordner und `name:` zum Beispiel `grill-mich`), damit klar ist, welche startet.

© 2026 cooexistenz · Kevin Coo · cooexistenz.de · Persönliches Exemplar, Weitergabe und Verkauf nicht gestattet · ID CX-20261102-AB9F6053

// 05

Ohne Skill-Funktion: als ersten Prompt

In Microsoft Copilot, Gemini, ChatGPT oder jedem anderen Chat kopierst du die Regeln einfach als erste Nachricht rein, dahinter dein Plan:

```
Grill mich zu meinem Plan, bevor du irgendetwas baust oder schreibst. Regeln:
1. Stell immer nur eine Frage und warte auf meine Antwort.
2. Fang mit den Entscheidungen an, von denen andere abhängen: Ziel, für wen, was ist drin und was nicht.
3. Gib nach jeder Antwort deine eigene Empfehlung mit kurzer Begründung.
4. Was du selbst nachschlagen kannst, frag mich nicht.
5. Akzeptier keine schwammigen Antworten wie "machen wir später", hak nach.
6. Prüf Annahmen, Risiken, was schiefgehen kann und was sich später schwer ändern lässt.
7. Hör erst auf, wenn nichts mehr geraten ist. Dann fass zusammen: Entscheidungen, offene Punkte, Risiken, Änderungen am Plan.
Mein Plan: [dein Plan in 3–5 Sätzen]
```

Dauerhaft speichern, wenn du es oft brauchst:

- **ChatGPT:** im Projekt oben rechts auf „•••“ > Project settings > Instructions einfügen. Das gilt dann nur in diesem Projekt und geht vor deinen allgemeinen Anweisungen.
- **Gemini:** als Gem (Gems > New Gem > Anweisungen > Save). Ab November 2026 werden Gems bei privaten Konten automatisch zu Skills, die du mit „/“ und dem Namen startest.
- **Microsoft 365 Copilot:** Die Custom instructions (Chat settings > Personalization) gelten für jeden Chat. Für Grill Me nimm lieber den ersten Prompt, sonst wirst du bei jeder Kleinigkeit ausgefragt.

// 06

Mein Ablauf aus dem Reel

1. **Grillen.** Neuer Chat, `/grill-me`, Plan rein. Am Ende steht `decisions.md`.
2. **Ideen auf skills.sh suchen. Installiert wird nichts.** Claude sucht passende Skills und schaut rein. Ohne Installieren geht das mit `npx skills find [Suchwort]` oder `npx skills add [owner/repo] --list` (zeigt nur, was drin ist).
3. **Jede Idee gegen den Plan legen.** Hilft sie meinem Ziel aus `decisions.md`? Wenn nicht, fliegt sie raus, egal wie viele Installationen sie hat.
4. **Erst dann installieren.** Meist bleiben ein oder zwei Skills übrig. Vorher schaue ich, was in den Dateien steht.

© 2026 cooexistenz · Kevin Coo · cooexistenz.de · Persönliches Exemplar, Weitergabe und Verkauf nicht gestattet · ID CX-20261102-AB9F6053



DER MENTOR SAGT

Die Rangliste sagt dir, was viele installieren. Dein Plan sagt dir, was du brauchst.

// 07

Prompts zum Kopieren

Start mit Skill:

```
/grill-me Mein Plan: [was du vorhast, für wen, bis wann]. Ich will vorher wissen, was ich noch nicht entschieden habe.
```

Ideen suchen, ohne zu installieren:

```
Such auf skills.sh nach Skills, die zu meinen Entscheidungen in decisions.md passen. Installier nichts. Gib mir pro Idee: Name, Link, was er macht, Installationen und Audit-Ergebnis.
```

Ideen gegen den Plan legen:

```
Leg jede dieser Ideen gegen decisions.md: Hilft sie meinem Ziel, was kostet sie mich an Aufwand, was könnte stören? Sortier in "passt" und "fliegt raus", je ein Satz Begründung.
```

Vor dem Installieren reinschauen:

```
Zeig mir, welche Dateien in [owner/repo] stecken, ob Skripte dabei sind und was sie tun, bevor ich irgendetwas installiere.
```

Gegen das Abnicken:

```
Ich habe dir zuletzt nur zugestimmt. Nenn mir die zwei Entscheidungen, bei denen ich am ehesten falsch liege, und frag mich dazu noch einmal.
```

Wenn es zu groß wird:

```
Der Plan ist zu groß für eine Runde. Teil ihn in 3 bis 5 Teile und grill mich zuerst zum wichtigsten Teil.
```

© 2026 cooexistenz · Kevin Coo · cooexistenz.de · Persönliches Exemplar, Weitergabe und Verkauf nicht gestattet · ID CX-20261102-AB9F6053

Für andere verständlich machen:

```
Fass unsere Entscheidungen als kurze Projektbeschreibung für [Kollegin / Kunde]
zusammen, ohne Fachbegriffe, mit den offenen Punkten am Ende.
```

// 08

Wer was davon hat

WER	PROBLEM	WAS DU MIT GRILL ME MACHST
Büro	Die KI baut eine Excel-Auswertung, die keiner wollte	Vorher klären: Wer liest sie, welche Zahl zählt, woher kommen die Daten
Freelancer	Kunde sagt „mach mal was Modernes“	Den Auftrag vor dem Angebot grillen lassen, `decisions.md` als Grundlage fürs Angebot
Einsteiger	Du weißt nicht, was du der KI sagen sollst	Die KI fragt, du antwortest. Ihre Vorschläge zeigen dir, woran man denken muss
Selbstständige	Eine Entscheidung dreht sich seit Wochen im Kreis	Die Entscheidung grillen: Was ist umkehrbar, was nicht, was ist der kleinste Schritt

// 09

Alternativen

WAS	WIE ES ARBEITET	WANN NEHMEN
Matts Original	Fragen in Runden, mehrere auf einmal, jede mit Vorschlag, schreibt keine Datei	Schnell durch viele Fragen, überall nutzbar
brainstorming	Aus obra/superpowers: eine Frage pro Nachricht, dann 2 bis 3 Wege zur Wahl, dann ein Design-Dokument, gebaut wird erst nach deinem OK	Wenn am Ende direkt ein fertiger Plan stehen soll; kommt als ganzes Paket
grill-with-docs	Wie Grill Me, liest aber deinen Code mit und schreibt ein Glossar und Entscheidungs-Notizen	Wenn es um ein bestehendes Software-Projekt geht
to-spec	Macht aus dem Gespräch eine Spezifikation, ohne neue Fragen (hieß früher to-prd)	Direkt nach dem Grillen, im selben Chat
Plan-Modus	In Claude Code eingebaut (`Shift+Tab`): Claude liest und schlägt einen Plan vor, ändert nichts ohne dein OK	Wenn der Plan klar ist. Fürs Grillen rät Matt, ihn aus zu lassen, weil er zum schnellen Plan drängt

© 2026 cooexistenz · Kevin Coo · cooexistenz.de · Persönliches Exemplar, Weitergabe und Verkauf nicht gestattet · ID CX-20261102-AB9F6053

Einen Test, der zeigt, dass eins davon besser ist, habe ich nicht gefunden. Die Basis ist bei allen gleich: erst fragen, dann bauen.

// 10

Grenzen und Risiken

- **„Ok“ zu allem ist die größte Falle.** Matt nennt es Passivität: 40 Mal „einverstanden“ und am Ende hast du den Plan der KI abgenickt. Widersprich, sag „weiß ich nicht“, wenn du es nicht weißt.
- **Lange Gespräche kosten Tokens und Qualität.** Jede Frage und Antwort bleibt im Gespräch. Wird es sehr lang, werden die Fragen laut Matt schlechter. Bei 200 Fragen ist der Plan zu groß: erst aufteilen.
- **Manche Fragen kann man nicht beantworten, nur ausprobieren.** „Wie soll sich das anfühlen?“ klärst du mit einem schnellen Entwurf, nicht mit noch einer Frage.
- **Nur aus vertrauenswürdigen Quellen installieren.** Anthropic rät: Skills nur von dir selbst oder von Anthropic, alles andere vorher komplett lesen, wie beim Installieren von Software. skills.sh prüft Skills nicht selbst, es zeigt nur Scans von Gen, Socket und Snyk. GitHub schreibt ausdrücklich, dass es Skills nicht prüft. Anthropic: Sicherheit bei Skills
- **Gleiche Namen.** Mehrere fremde Skills heißen „grill-me“. Liegen zwei mit gleichem Namen bei dir, gewinnt in Claude Code der persönliche vor dem Projekt-Skill. Schau also, welcher wirklich startet.
- **Deine Daten.** Was du beim Grillen erzählst, geht an den KI-Anbieter wie jeder andere Chat. Kundennamen und Interna nur, wenn dein Vertrag das erlaubt.

© 2026 cooexistenz · Kevin Coo · cooexistenz.de · Persönliches Exemplar, Weitergabe und Verkauf nicht gestattet · ID CX-20261102-AB9F6053

// 11

Deine erste Stunde

- [ ] Meine Version oben in eine Datei `SKILL.md` kopieren, Ordner `grill-me`
- [ ] In deinem Tool installieren (Claude Code, claude.ai, Codex, Copilot oder Cursor) oder den Prompt-Kasten speichern
- [ ] Einen echten Plan wählen, den du diese Woche angehen willst, und in 3 bis 5 Sätzen aufschreiben
- [ ] Neuer Chat, Grill Me starten, Plan rein
- [ ] Mindestens einmal widersprechen und einmal „weiß ich nicht“ sagen
- [ ] Zusammenfassung oder `decisions.md` speichern
- [ ] Auf skills.sh nach Ideen suchen lassen, `decisions.md` legen, höchstens ein bis jede gegen zwei behalten
- [ ] Vor dem Installieren die Dateien lesen (lassen)

© 2026 cooexistenz · Kevin Coo · cooexistenz.de · Persönliches Exemplar, Weitergabe und Verkauf nicht gestattet · ID CX-20261102-AB9F6053

// 12

Mein Tipp

Fang mit etwas Kleinem an, das nichts mit Code zu tun hat: einem Angebot, einem Workshop, einer Website-Idee. Du merkst nach drei, vier Fragen, wie viel du der KI sonst verschwiegen hättest. Und die Entscheidungen bleiben bei dir, die KI liefert nur die Fragen und Vorschläge. Das Original findest du auf skills.sh.

Alle Links und Befehle am 02.11.2026 auf den offiziellen Seiten geprüft.

– Kevin Coo

Mehr Anleitungen mit Stand und Quelle, in einfachen Worten und mit Datenschutz im Blick: auf cooexistenz.de.

cooexistenz.de ↗

Kevin Coo · cooexistenz.de

GRILL

© 2026 cooexistenz · Kevin Coo · cooexistenz.de · Persönliches Exemplar, Weitergabe und Verkauf nicht gestattet · ID CX-20261102-AB9F6053