

# Regeln für KI im Team: eine Vorlage

*Was der Agent nicht darf, was ihn stoppt und wer die Regeln besitzt. Für Ihr eigenes Repository an einem Nachmittag.*

Für CTOs, Tech-Leads und Entwickler. Ihr Team nutzt bereits Claude Code, Codex, Cursor oder Copilot. Bisher arbeitet jeder nach eigenen Regeln. Diese Vorlage macht daraus eine gemeinsame Ebene.

KI schreibt heute unseren gesamten Code. Für das Ergebnis bürgt ein Senior-Entwickler. Seit 2023 haben wir 15+ Produktionsprojekte ausgeliefert. Diesen Rahmen nutzen wir dabei.

## 1. Warum Textregeln nicht reichen

Sie verbieten dem Agenten, eine Datei zu öffnen. Er öffnet sie nicht. Er schreibt sich ein Skript, das den Inhalt ausgibt.

Das ist keine böse Absicht. Der Agent hat eine Aufgabe und sucht Wege. Ein Satz im Prompt ist für ihn ein Hindernis, keine Mauer. Jede Regel unten enthält eine Spalte, die den meisten „KI-Richtlinien“ fehlt. Sie zeigt, was ihn wirklich stoppt.

| Ebene                 | Was das bedeutet                                                                 | Stärke         |
|-----------------------|----------------------------------------------------------------------------------|----------------|
| Nur ein Satz          | Regel in CLAUDE.md oder AGENTS.md. Der Agent hält sich meistens daran.           | am schwächsten |
| Einstellung oder Hook | Der Befehl stoppt, bevor er startet.                                             | mittel         |
| CI                    | Änderungen erreichen den Main-Branch erst bei grünem Gate.                       | stark          |
| Lint oder Test        | Der Build bricht ab mit Fehlermeldung. Der Agent liest sie und repariert selbst. | am stärksten   |

Wurde eine Regel zweimal gebrochen, reicht ein reiner Satz nicht mehr. Stufen Sie diese Regel eine Ebene höher.

## 2. Acht Dinge, die der Agent nicht darf

| # | Der Agent darf nicht                   | Warum                                                                                                                             | Was ihn stoppt                                   |
|---|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------|
| 1 | Zur Produktionsdatenbank verbinden     | Falsche Abfragen auf Produktion lassen sich nicht rückgängig machen. Für Diagnosen reicht ein Read-only-Replikat oder Log-Export. | Einstellung (Verbot) + getrennte Zugangsdaten    |
| 2 | Secrets lesen (.env, Schlüssel, Token) | Was der Agent liest, landet im Kontext, in Logs und manchmal im Commit.                                                           | Einstellung + Secrets außerhalb des Repositories |

| # | Der Agent darf nicht                          | Warum                                                                       | Was ihn stoppt                                         |
|---|-----------------------------------------------|-----------------------------------------------------------------------------|--------------------------------------------------------|
| 3 | Produktionszugänge zur Cloud nutzen           | Gleicher Grund wie bei der Datenbank, aber mit größerer Reichweite.         | Einstellung (Verbot)                                   |
| 4 | Deployment oder Migration starten             | Was sich nicht zurückdrehen lässt, läuft zwingend über einen Menschen.      | Einstellung (Nachfragen) + Umgebungsfreigabe in der CI |
| 5 | Historie überschreiben oder auf Main pushen   | Force-Push und reset --hard zerstören unbemerkt fremde Arbeit.              | Einstellung + Hook + Branch-Schutz auf dem Server      |
| 6 | Den Test schreiben, nach dem Sie ihn bewerten | Schreibt der Agent den Test, bestimmt er seine eigene Note.                 | CI-Prüfung + Regel für den Reviewer                    |
| 7 | Die eigene Änderung reviewen                  | Dasselbe Modell für Code und Prüfung hat dieselben blinden Flecken doppelt. | Verbindliche PR-Freigabe durch einen Menschen          |
| 8 | Kundendaten in den Prompt einfügen            | Personenbezogene Daten gehören in kein Werkzeug ohne Freigabe.              | <b>nur ein Satz</b> + Teamabsprache                    |

Beim achten Punkt schreiben wir ehrlich „nur ein Satz“. Mechanisch hält diesen Punkt heute noch kein Werkzeug zuverlässig auf. Eine Regel, die ihre Schwäche zugibt, schlägt jede vorgetäuschte Stärke.

**Wie viele Ihrer Regeln sind nur ein Satz? Genau so viele Lücken haben Sie.**

### 3. Wohin welche Regel gehört

Der Agent lädt die Regeldatei bei jedem Start. Je mehr Text er liest, desto schlechter beachtet er das Wesentliche. Deshalb nutzen wir drei Ebenen:

1. **Repository-Root:** Kurze CLAUDE.md oder AGENTS.md. Klärt Zweck des Repos, Testbefehle, Verbote und Definition fertiger Arbeit.
2. **Ordner:** Modulregeln in einer Datei direkt im jeweiligen Ordner. Der Agent lädt sie erst bei der Arbeit dort.
3. **Lint oder Test:** Das Wichtigste. Niemand muss das lesen. Es stoppt fehlerhafte Änderungen automatisch.

Nutzen Sie mehrere Werkzeuge? Pflegen Sie Regeln in AGENTS.md. In CLAUDE.md verweisen Sie mit der Zeile `@AGENTS.md` darauf. Zwei Kopien driften auseinander, bevor es jemand bemerkt.

#### Dateistruktur

```

CLAUDE.md oder AGENTS.md    kurz: Repo, Befehle, Links
.claude/
  settings.json               Verbote in maschinenlesbarer Form
  hooks/push-gate.sh          was Einstellungen nicht abdecken
  rules/
    00-never.md               8 Verbote + Spalte „was ihn stoppt“
    10-task.md                was eine Aufgabe braucht, bevor der Agent startet
    20-gates.md               Lint, Typecheck, Tests: blockiert ja/nein

```

## 4. Vorlagen für das Repository

Die Beispiele gelten für Claude Code. Das Prinzip aus Verbot, Hook, CI und Lint gilt für alle Werkzeuge. Nur die Schreibweise unterscheidet sich.

### Verbote in den Einstellungen (Punkte 1 bis 5)

```
{
  "permissions": {
    "deny": [
      "Bash(psql:*)",
      "Bash(git push --force:*)",
      "Bash(git reset --hard:*)",
      "Read(.env.prod*)",
      "Read(**/*.pem)"
    ],
    "ask": [
      "Read(**/.env*)",
      "Bash(npx prisma migrate deploy:*)"
    ]
  }
}
```

Achtung: Die Regel prüft den Befehlstext, nicht die Absicht. Darf der Agent `cat` nutzen, umgeht `cat .env` das Leseverbot. Secrets gehören deshalb außerhalb des Repositories. Skripte laden sie, geben sie aber niemals aus.

### Hook gegen Push auf Main (Punkt 5)

```
#!/bin/bash
# .claude/hooks/push-gate.sh (PreToolUse, Bash)
# exit 0 = erlauben, exit 2 = stoppen
CMD=$(jq -r '.tool_input.command // empty')
[[ "$CMD" != *"git push"* ]] && exit 0
case "$CMD" in
  *" main"*|" master"*)
    echo "Push auf Main ist verboten. Eröffnen Sie einen PR." >&2
    exit 2 ;;
esac
exit 0
```

Zwei typische Fehler setzen den Hook heimlich außer Kraft. Ein `|| true` hinter dem Hook überschreibt Exit 2 zu 0. Ein `>/dev/null` verwirft die Fehlermeldung. Der Agent versteht dann nicht, warum der Befehl scheitert. Testen Sie den Hook einmal unter realen Bedingungen. Aktivieren Sie serverseitigen Branch-Schutz. Der Hook greift schließlich nur dort, wo er auch installiert ist.

## Tests und Review (Punkte 6 und 7)

- **Der Test muss zuerst fehlschlagen.** Und zwar auf dem Originalcode. Sonst bestätigt der Agent nur sich selbst.
- **Tests aus der Aufgabenstellung genehmigt ein Mensch.** Die CI warnt, wenn ein PR Tests und Code gleichzeitig anfasst. Der Reviewer entscheidet.
- **Wer geschrieben hat, reviewt nicht.** Starten Sie das Review in einem frischen Kontext, idealerweise mit einem anderen Modell. Es erhält das Diff und Ihre Regeln, keinen Chatverlauf.
- **Die CI muss blockieren.** Eine Pipeline, die nur warnt, dient bloß als Dekoration.

## 5. Woher neue Regeln kommen

Direkt aus dem Review. Ein Kommentar, der zum zweiten Mal fällt, ist kein Entwicklerfehler. Er markiert eine Lücke in Ihren Regeln.

| Häufigkeit  | Was zu tun ist                                                |
|-------------|---------------------------------------------------------------|
| Erstes Mal  | Reparieren und weitermachen                                   |
| Zweites Mal | Regel für den Agenten formulieren                             |
| Drittes Mal | Lint oder Test schreiben, der das Problem automatisch abfängt |

Die Bedingung: Reviews werden im Pull Request dokumentiert, nicht bloß mündlich im Call. Aus mündlichen Gesprächen lässt sich später keine Regel ableiten.

## 6. Verantwortung und Messung

Regeln ohne festen Verantwortlichen veralten innerhalb weniger Monate. Der Agent befolgt sie weiter, obwohl sie längst ungültig sind. Kopieren Sie folgende Struktur in `90-ownership.md`:

```
Verantwortlicher: [Name]
Neue Regel: PR in .claude/rules/, Freigabe durch den Verantwortlichen.
Bereinigung: erster Montag im Monat, 30 Minuten.
Eskalation: zweimal verletzt -> eine Stufe höher
    nur ein Satz -> Einstellung oder Hook -> CI -> Lint
Monatliche Notiz: Wie viele Regeln noch reiner Satz sind.
```

Drei Kennzahlen sollten Sie jeden Monat im Blick behalten:

- **Review-Kommentare pro PR.** Funktionieren die Regeln, sinkt diese Zahl kontinuierlich.
- **Wie oft ein Gate die Änderung gestoppt hat.** Und wie oft der Agent sie danach selbst reparierte.
- **Wie viele Regeln noch reine Sätze sind.** Das ist Ihr Arbeitsplan für den kommenden Monat.

Messen Sie weder Commit-Zahlen noch Codezeilen. Die KI bläht beides in Sekundenschnelle auf.

## 7. Arbeitsblatt für das Team (45 Minuten)

---

Ausdrucken, auf den Tisch legen und gemeinsam ausfüllen.

| #  | Agent niemals                                   | Was setzt es durch? (Satz / Einstellung / CI / Lint) | Verantwortlich |
|----|-------------------------------------------------|------------------------------------------------------|----------------|
| 1  | Produktionsdatenbank, nur Replikate oder Export |                                                      |                |
| 2  | Secrets außerhalb des Agentenkontexts           |                                                      |                |
| 3  | Produktionszugänge zur Cloud                    |                                                      |                |
| 4  | Deploy und Migration nur durch Menschen         |                                                      |                |
| 5  | Force-Push und Push auf Main                    |                                                      |                |
| 6  | Test, nach dem wir ihn bewerten                 |                                                      |                |
| 7  | Review der eigenen Änderung                     |                                                      |                |
| 8  | Kundendaten im Prompt                           |                                                      |                |
| 9  |                                                 |                                                      |                |
| 10 |                                                 |                                                      |                |

Sie sind fertig, wenn jede Zeile eine klare Durchsetzung und einen Verantwortlichen hat.

## Wollen Sie Regeln mit jemandem aufsetzen, der täglich damit arbeitet?

---

In 15 Minuten mit Filip gehen wir Ihr ausgefülltes Arbeitsblatt durch. Wir zeigen Ihnen, welche zwei Regeln wir als Erstes höherstufen würden. Keine Präsentation.

**Vereinbaren Sie 15 Minuten mit Filip:** [digitaliko.com/de/contact](https://digitaliko.com/de/contact)

Digitaliko s.r.o. · [www.digitaliko.com](https://www.digitaliko.com) · Individualsoftware in Wochen, die Ihnen ab Tag eins gehört.