KI-Rechner

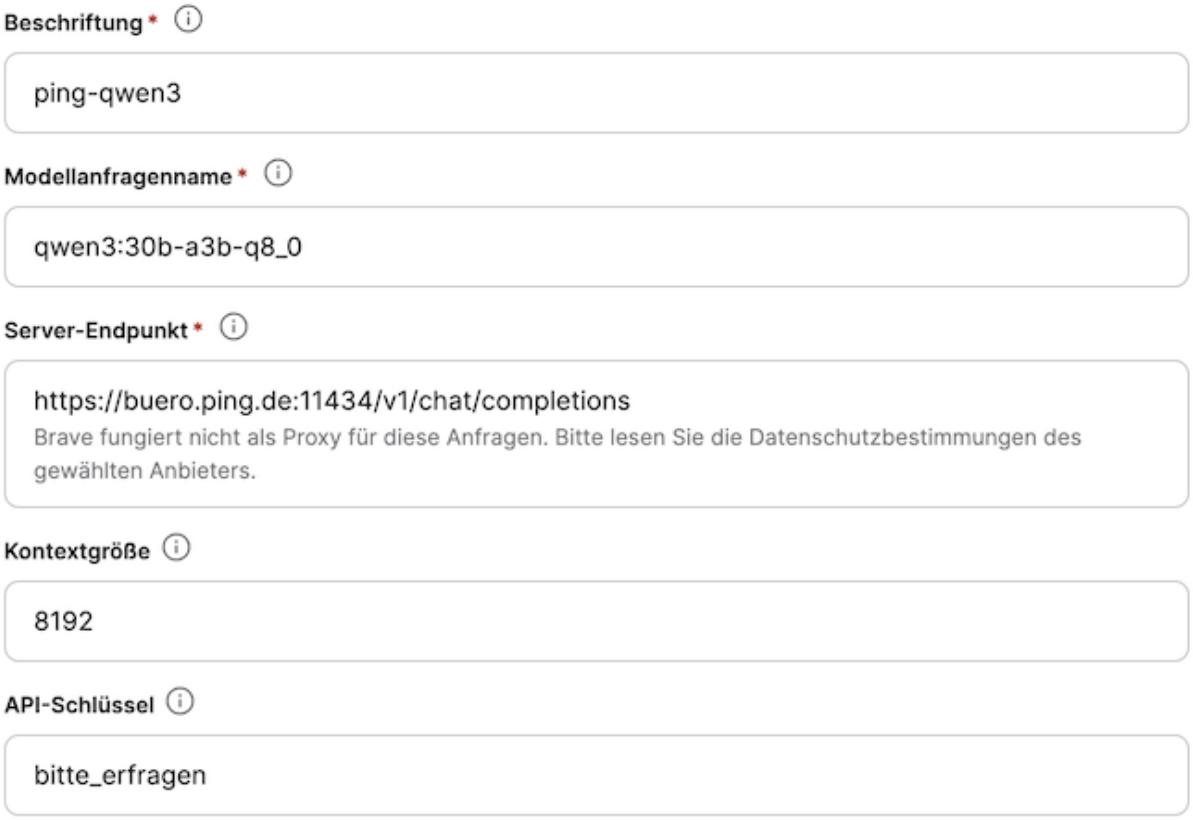
- Brave-Leo-AI mit PING KI
- Visual Studio Code und OpenCode
- Inferenz Benchmarks
- Zugriff auf vLLM via OpenAI-kompatibler API

Brave-Leo-AI mit PING KI

Konfigurieren von Leo AI im Brave Browser mit dem PING AI Server

Auf cogito (KI-Server) läuft ein Ollama-server, der einen OpenAI-API kompatiblen Endpunkt bereitstellt. Er ist unter https://ki.ping.de:8000/ erreichbar, man benötigt für den Zugriff ein Bearer Token.

1. Besorgt euch das Bearer Token, siehe API-Token Seite (für Mitglieder)
2. Im Brave Browser oben rechts auf das "Leo AI" Icon  klicken.
3. Klickt oben rechts auf die 3 Punkte übereinander " ⋮ "
4. Klickt ganz unten auf "Erweiterte Einstellungen" (mit dem Zahnrad). Ihr landet dann auf `brave://settings/leo-ai`
5. Unter "Bringen Sie Ihr eigenes Modell mit" klickt auf "Neues Modell hinzufügen"
6. Macht folgende Einstellungen:
   - Beschriftung: `ping-qwen3`
   - Modellanfragename: `qwen36-27b` (diesen Namen seht ihr so auch im open-webui oder s.u.)
   - Server-Endpunkt: `https://ki.ping.de:8000/v1/chat/completions`
   - Kontext-Größe: `65536` (mehr geht auch, hängt vom LLM und freien VRAM ab).
   - API-Schlüssel: Siehe Punkt 1. Ohne "Bearer" davor eingeben.
     ACHTUNG, Screenshot veraltet:



7. Klickt auf "Modell speichern"
8. Stellt das "Standardmodell für neue Unterhaltungen" auf `ping-qwen3`

Fertig. Wenn ihr jetzt auf das Leo-AI-Icon klickt startet eine neue Unterhaltung mit dem LLM auf dem PING Server cogito. Wenn ihr auf das "Seitenleiste anzeigen" Icon daneben klickt teilt sich das Browserfenster und ihr seht neben der Webseite das Leo AI Chatinterface, dort könnt ihr dann das LLM zur gerade aktiven Webseite befragen (zusammenfassen etc.).

Die KI von der Brave Search läuft davon unabhängig in der Cloud von Brave.

Verfügbare Modelle auflisten

Wenn ihr eine Liste aller installierten Modelle sehen möchtet, dann könnt ihr das entweder in open-webui oder es geht über die Ollama API wie folgt (ihr benötigt die Befehle `curl` und `jq`):

```
BEARER_TOKEN=siehe_oben
curl -sH "Authorization: Bearer $BEARER_TOKEN" https://ki.ping.de:8000/v1/models | jq
```

Visual Studio Code und OpenCode

Wie nutzt ihr Visual Studio Code und OpenCode mit dem PING KI Server?

Bearer Token / API key / API token

Zunächst braucht ihr das Bearer Token. Es heißt manchmal auch API Token.

PING Mitglieder finden es unter API Token.

Ermittelt welche Modelle der Server anbietet, die euch fürs Programmieren interessieren. Unter `max_model_len` seht ihr die maximale Kontext-Länge.

```
BEARER_TOKEN=siehe_oben
curl -sH "Authorization: Bearer $BEARER_TOKEN" https://ki.ping.de:8000/v1/models|jq
```

Aktuell (2026-04) gibt es nur ein Model: `qwen36-27b` mit Kontext-Länge 200.000. Es ist gut und schnell und bietet Tool calling, ist multi-modal etc. □□

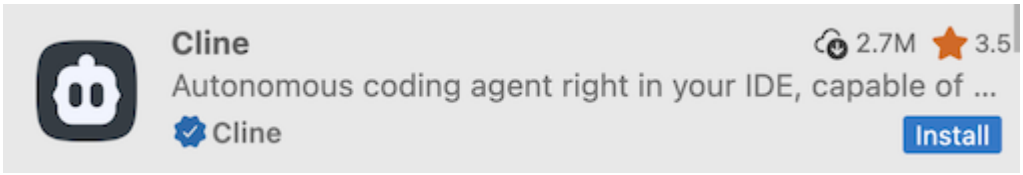
Dann geht es wie folgt:

Visual Studio Code

Für VS Code gibt es mehrere KI-Plugins die in Frage kommen. Hier findet ihr für einige exemplarische die benötigten Schritte zur Konfiguration:

Cline Plugin

1. In VS Code das Cline Plugin installieren. Achtet darauf dass es das blaue Checkmark hat:



2. *How will you use Cline?* "**Bring my own API key**"
3. API Provider:**OpenAI**
4. Custom base URL: `https://ki.ping.de:8000`
5. API key: **siehe_oben** (nutzt das Bearer Token)
6. Model: `qwen36-27b`

Fertig!

Kilo Code Plugin

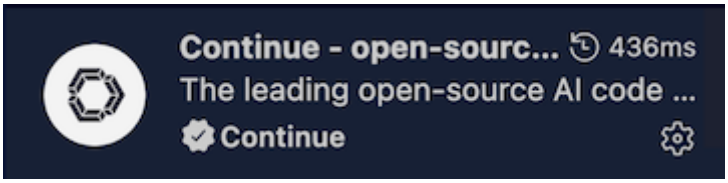
1. In VS Code das Kilo Code AI Agent Plugin installieren. Achtet darauf dass ihr kein falsches erwischt. Aktuell sieht es etwa so aus:


2. Wählt "Use your own API key"
3. API Provider: OpenAI compatible
4. Base URL: `https://ki.ping.de:8000`
5. API Key: **siehe_oben** (nehmt das Bearer Token)
6. Model Name: `qwen36-27b`

Fertig!

Continue Plugin

Installiert das Continue Plugin in VS Code.



Nehmt folgende Config Datei (ungeprüft). Tragt dort den API Key ein in der letzten Zeile. (Doku)

```
name: My Config
version: 0.0.1
schema: v1

models:
  - name: qwen36-ping
    provider: openai
    model: qwen36-27b
    apiBase: https://ki.ping.de:8000/v1
    apiKey: siehe_oben
```

Fertig!

OpenCode

Für OpenCode müsst ihr eine `opencode.json` Datei anlegen z.B. im Verzeichnis `~/.config/opencode/`

Tragt das/die Modelle in die opencode.json Datei ein, in etwa wie so:

```
{
  "$schema": "https://opencode.ai/config.json",
  "provider": {
    "ki.ping.de": {
      "npm": "@ai-sdk/openai-compatible",
      "name": "ki.ping.de",
      "options": {
        "baseURL": "https://ki.ping.de:8000/v1",
        "apiKey": "siehe_oben"
      },
      "models": {
        "qwen36-27b": {
          "name": "Qwen 3.6 (ping)",
          "modalities": { "input": ["image", "text"], "output": ["text"] },
          "limit": {
            "context": 200000,
```

```
            "output": 16000
          }
        }
      }
    }
  }
}
```

Wenn ihr das nächste Mal OpenCode startet und /connect tippt könnt ihr nach "PING" suchen und solltet den Eintrag "vLLM (PING)" angezeigt bekommen. Wenn euch dann das OpenCode nach dem API Key fragt, gebt das Bearer Token (s.o.) ein.

Inferenz Benchmarks

2026-04-18 vLLM mit cyankiwi/Qwen3.6-35B-A3B-AWQ-4bit

vLLM optionen:

```
--model cyankiwi/Qwen3.6-35B-A3B-AWQ-4bit
--tensor-parallel-size 2
--max-model-len 65536
--gpu-memory-utilization 0.85
--enable-prefix-caching
--reasoning-parser qwen3
--enable-auto-tool-choice
--tool-call-parser qwen3_coder
--max-num-seqs 32
--speculative-config '{"method":"qwen3_next_mtp","num_speculative_tokens":2}'
```

Benchmark mit `uvx llama-benchy --base-url http://cogito.buero.ping.de:8000/v1 --depth 2000 32768 63000`

model	test	t/s	peak t/s	ttfr (ms)	est_ppt (ms)	e2e_ttft (ms)
cyankiwi/Qwen3.6-35B-A3B-AWQ-4bit	pp2048 @ d2000	5463.38 ± 111.87		748.82 ± 14.93	741.48 ± 14.93	748.93 ± 14.93
cyankiwi/Qwen3.6-35B-A3B-AWQ-4bit	tg32 @ d2000	103.13 ± 22.06	112.49 ± 24.41			
cyankiwi/Qwen3.6-35B-A3B-AWQ-4bit	pp2048 @ d32768	5178.25 ± 25.55		6731.33 ± 33.06	6724.00 ± 33.06	6731.41 ± 33.05
cyankiwi/Qwen3.6-35B-A3B-AWQ-4bit	tg32 @ d32768	25.65 ± 1.43	27.93 ± 1.52			
cyankiwi/Qwen3.6-35B-A3B-AWQ-4bit	pp2048 @ d63000	4534.72 ± 42.10		14353.15 ± 133.93	14345.82 ± 133.93	14353.26 ± 133.94
cyankiwi/Qwen3.6-35B-A3B-AWQ-4bit	tg32 @ d63000	12.85 ± 3.50	14.45 ± 3.21			

Plan: P2P einschalten, da geht noch mehr...

Zugriff auf vLLM via OpenAI-kompatibler API

Für viele Tools benötigt ihr lediglich den Zugriff auf den OpenAI-kompatiblen API Endpunkt, den unser vLLM bereitstellt.

Hier findet ihr die nötigen Daten:

Base URL: `https://ki.ping.de:8000/v1`

API-Key: siehe API Token (für Mitglieder)