LoKI User Guide

Benutzerhandbuch / User Manual

Version: 1.0
Projekt: LoKI – Local AI Assistant
Autor: Knut Schneider
Datum: 06.03.2026

Inhaltsverzeichnis / Table of Contents

Deutsch

1. LoKI

LoKI ist ein Desktop-Assistent zum lokalen Ausführen von Large Language Models (LLMs). Das Programm wurde in C++ mit dem Qt-Framework entwickelt und nutzt die llama.cpp Inferenzbibliothek. LoKI lädt abhängig von der Hardware automatisch das passende Backend (CPU, CUDA, Vulkan oder OpenCL). Dadurch kann das Programm sowohl auf reinen CPU-Systemen als auch auf GPU-beschleunigten Systemen betrieben werden. Je nach dem was das Programm beim ersten Start vorfindet werden eventuell System-Treiber über ein install-Skript nach installiert.

Das AppImage enthält alle notwendigen Laufzeitkomponenten wie Python, Node-Module und verschiedene Runtime-Bibliotheken. Dadurch kann LoKI weitgehend unabhängig von der Linux-Distribution betrieben werden.

Ordnerstruktur

Beim ersten Start mit --install wird ein Arbeitsordner erstellt:

LoKI/
├─ Data (Chat-Kontext Datenbank)
├─ Docs (Dokumentation)
├─ Model (installierte LLM Modelle)
├─ ModelUseFiles (Dateizugriff für LLM's)
├─ Plugins (MCP Tools)
└─ Voice (TTS Stimmen)

Data: Dort befindet sich die sqlite3 (ist auch im AppImage)Datenbank für den Chat-Kontext. Der Chat-Kontext ist auf der Hauptseite von LoKI auf der linken Seite und führt alle geführten Chats auf. gespeicherte Chats können wieder aktiviert und fortgesetzt werden oder per rechten Mausklick gelöscht werden. Ist man in einem Chat kann man mit dem Button „+ Neuer Chat“ in einen neuen Chat wechseln.

Docs: Hier liegen die Dokumentation.

Model: Hier werden die LLM's gespeichert die per Download oder Drag&Drop erstellt werden. Beim Download kann man im Download Bereich von LoKI aus einer Liste verfügbarer LLM's (Hugging Face) auswählen. Es können reine .GGUF Modelle aber auch HF Modelle geladen werden, wobei bei den HF Modellen nach dem Download das Modell automatisch nach GGUF gewandelt wird, da nur GGUF Modelle geladen werden können.

Auf der Downloadseite ist es auch möglich ein GGUF Modell zu Requantisieren. Das bringt mehr Speed geht aber zu lasten der Genauigkeit des Modells. Weiteres in Quantisierung-Doc.

ModelUseFiles: Das ist der Ordner den das geladen Model benutzen kann um Files anzulegen und zu manipulieren. Im MCP-Modus „Local“ sind ein paar Tools die dieses ermöglichen. Der Pfad zum Ordner ist in den Settings frei wählbar und kann auch wo anders hin gelegt werden. **Achtung**: Überlegen sie genau worauf sie eine LLM zugreifen lassen!

Plugins: Dieser Ordner enthält die Ordner „local“, „stdio“ und „http“ in denen sich die Beispiel-Tools befinden. Dazu siehe „Tools-Doc“.

Voice: Hier sind ein paar Stimmen für die akustische Ausgabe. Es sind deutsche und englische Stimmen vorhanden. In diesem Zusammenhang vielen Dank an Thorsten-Voice für seine Stimme mit frei wählbaren Emotionen die man auch über die Settings einstellen kann. (Hier gilt mein Dank an Thorsten-Voice)

Es können auch andere Stimmen dort gespeichert werden. Es ist nur zu beachten das die Stimme mit ihrer .json Konfigurations-Datei gespeichert wird.

Nach dem ersten Start von LoKI mit **–install** kann man dann das Programm auch direkt aus „Anwendungen“ oder durch Doppelklick starten. Es ist keine Konsole mehr notwendig.

2. Editor

Der Editor ist ein integrierter Code- und Texteditor. Er ermöglicht das Bearbeiten von Plugin-Scripts, Konfigurationsdateien und Dokumentationen. Syntax-Highlighting erleichtert dabei die Arbeit mit JSON, Python oder JavaScript.

Der Editor kann Dokumente direkt an das aktuell laufende Modell, über die 3 unteren Buttons, übergeben, um z. B. Code zu erklären, zu überprüfen oder automatisch Dokumentation generieren zu lassen.

Tipp: wenn man in den Settings den ModelUseFiles-Pfad auf den Ordner Plugins/lokal, Plugins/stdio oder Plugins/http setzt und dann im MCP-Mode „local“ wählt kann das LLM über die File-Tools direkt auf die Tools zugreifen, lesen, mit dem richtigen Prompt erweitern und so weiter.

Hinweis: Die in den Ordner vorhandenen Plugins/Tools sind sehr einfach gehalten und sollen nur die Funktionsweise bei LoKI zeigen. Zum Beispiel ist im Local-Ordner das Tool Calc nur sehr einfach. Es eignet sich aber hervorragend für eine Erweiterung durch ein brauchbares LLM wenn man den obigen Tipp folgt.

3. Download

Der Download-Bereich ermöglicht das direkte Laden von Modellen aus HuggingFace. Sowohl GGUF-Modelle als auch vollständige HuggingFace-Repositories können geladen werden.

HF-Modelle werden nach dem Download automatisch in das GGUF-Format konvertiert, da LoKI ausschließlich GGUF-Modelle für die Inferenz nutzt. Beim HF Download wird im Model-Ordner ein Ordner mit dem Namen des Model’s erstellt der alle Dateien des HF Downloads bündelt. Dieser Ordner wird nach der Konvertierung wieder gelöscht (wegen Platzbedarf) und es steht dann das fertige Model im GGUF Format zur Verfügung.

Hinweis: Es sind nicht alle LLM’s die angezeigt werden aud ladbar. Einige LLM’s aus der Liste sind “gated” und damit gesperrt. Der Download wird dann abgebrochen. In Der Suchleiste können sie nach LLM’s ihrer Wahl suche.

Mit LoKI getestete Modelle sind: DeepSeek-R1-Distill-Llama-8B.gguf
Qwen3-8B-Q4_K_M.gguf (LoKI quantisiert)
Llama3.1-8b-f16 und Q4_K_M
gpt-oss-20b_f16 (gepackt- mehr dazu in der Quantisierungs-Doc)

4. Quantisierung

Unten im Download-Bereich ist der Bereich für die Quantisierung. Quantisierung reduziert die Speichergröße eines Modells, indem Gewichte mit geringerer numerischer Präzision gespeichert werden. Während ein Modell im F16-Format 16-Bit Floating-Point-Gewichte verwendet, nutzen Quantisierungsformate wie Q8, Q5 oder Q4 komprimierte Darstellungen.

Typische Vorteile der Quantisierung:
• geringerer RAM / VRAM Bedarf

• schnellere Inferenz
• kleinere Modell-Dateien

Typische Quantisierungsstufen:
F16 → höchste Genauigkeit, größter Speicherbedarf
Q8 → sehr gute Qualität bei moderater Speicherreduktion
Q5 → guter Kompromiss zwischen Geschwindigkeit und Qualität
Q4_K_M → sehr effizient für Consumer-Hardware

Einige moderne Modelle sind bereits intern optimiert (z. B. durch Tensor Packing oder Shared Tensors). In solchen Fällen kann eine erneute Quantisierung die Dateigröße sogar erhöhen da das packing zerstört wird. Ein Beispiel-Kandidat ist gpt-oss-20b_f16 das dadurch nur eine Größe von 12GB hat statt 30GB (geschätzt).

Wählen sie hier das LLM und die Quantisierungs-Stufe aus und starten sie die Quantisierung. Danach steht das LLM zum testen bereit.

5. Settings

Auf der Setting Seite können verschiedene Parameter von LoKI eingestellt werden. LoKI selbst kann auf **deutsch/englisch** oder **hell/dunkel** erscheinen. Dies ist zur Laufzeit einstellbar.

Speaker-Settings: Der Pfad zum Voice Verzeichnis welches die Sprachmodelle beinhaltet.
Emotions ID gilt nur für das Voice Model „Thomas“.

Download-Settings: HuggingFace Token: Man sollte sich auf der HF Web-Seite einen solchen Token generieren lassen. Dadurch gehen etwas mehr Downloads. Der Token wird verschlüsselt gespeichert.
Hier kann man auch auswählen ob nach dem Download und eventueller Konvertierung(HF Download) gleich eine Quantisierung erfolgen soll und die Quantisierungs-Stufe die in diesem Fall benutzt werden soll.

Die Settings erlauben die Anpassung wichtiger Inferenz-Parameter des LLM.

n_ctx – Kontextgröße
Bestimmt wie viele Tokens gleichzeitig im Kontextfenster gehalten werden. Größere Werte ermöglichen längere Gespräche, benötigen aber mehr RAM/VRAM.

n_batch – Batchgröße
Steuert wie viele Tokens gleichzeitig verarbeitet werden.
Höhere Werte erhöhen die Geschwindigkeit, benötigen aber mehr Speicher.

n_gpu_layers – GPU Layer
Bestimmt wie viele Transformer-Layer auf der GPU berechnet werden.
0 = CPU only
-1 = so viele Layer wie möglich auf der GPU

Temperatur
Kontrolliert die Kreativität der Antworten.
Niedrige Werte → präzise Antworten
Hohe Werte → kreativere Antworten

top_p und top_k
Diese Parameter steuern das Sampling der Token-Wahrscheinlichkeiten und beeinflussen die Vielfalt der generierten Antworten.

MCP aktivieren: Aktiviert den MCP-Modus und blendet auf der Haupseite das MCP Fenster ein.

MCP-Modus: Auswählbar ist stdio, local oder http. Http dann über den MCP-Endpunkt.

Hinweis: Die Einstellung für die KI sollten nicht bei laufender LLM geändert werden. Das hat dann keine Auswirkungen auf den laufenden Thread. Die Parameter werden beim laden der LLM benutzt.

Hinweis: Die KI Settings werden beim laden der LLM von LoKI dynamisch überwacht und eventuell, bei nicht passen, herabgesetzt. Dies wird nach dem laden des LLM dann angezeigt. Diese Herabstufung ist etwas konservativ(sicher) und man kann noch etwas nach oben gehen.

6. Tools

LoKI unterstützt MCP-Tools (Model Context Protocol). Diese Tools können lokale Programme, Web-APIs oder Automatisierungen in den Kontext des LLM integrieren.

Hier sind die Tools der einzelnen MCP Modi. Diese sind beispielhaft und streckenweise nicht ausgebaut. Dieses ist mit Absicht, damit man zu mindest die funktionsweise der Tools sieht und selber den Ausbau oder die Entwicklung neuer Tools angehen kann. Dazu mehr in der Editor-Doc.

Alle Tools verwenden das jsonrpc 2.0 Protokoll.

Im Ordner Plugins befinden sie die Ordner:

local: Hier befinden sich die Tools für den lokalen MCP-Modus. Diese sind beispielhaft und in Python geschrieben. Die Python-Tools benutzen auch das Python312-Enviroment aus dem AppImage so das keine System-Abhängigkeiten vorhanden sind. Hier sind auch Tools die auf den Ordner ModelUseFiles zugreifen und von der LLM genutzt werden kann.

stdio: Hier sind Tools die mehr System Zugriffe erlauben. Darunter sind auch Tools die den im AppImage integrierten Browser(ms-playwright) nutzen. Auch hier wird das integrierte Node-modules genutzt. Zu Beachten ist das alle Tools in diesem Ordner einen mcp-tool-loader benutzen der das scannen der Tools übernimmt. Bei eigenen Tools bitte an den schon vorhanden Tools orientieren um die Funktion zu gewährleisten.

http: Hier liegt ein beispielhafter funktionierender Server. Dieser Server wird automatisch von LoKI in diesem Modus gestartet.

Der jeweilige Modus wird über die Settings-Seite eingestellt. Die Vorwahl muss vor dem Start der LLM ausgewählt werden. Währen die LLM aktiv ist, ist keine Änderung des MCP-Modus möglich da die Plugins/Tools beim Start der LLM, je nach Modi, gescannt werden und der LLM dann im Kontext bekannt gegeben werden. Die jeweils gefundenen Plugins/Tools werden auf der Hauptseite auf der rechten Seite im Fenster „Geladene Plugins“ angezeigt.

7. System Monitor

Der System Monitor zeigt CPU-, RAM- und GPU-Auslastung in Echtzeit. Besonders wichtig ist die VRAM-Überwachung bei GPU-Inference.

Diese Seite kann man mit **F9** abdocken und auf einen anderen Monitor legen so das man während eines Chats das Verhalten des Systems bewerten kann.Mit **F9** ist er auch wieder Andockbar oder einfach über „schließen“.

Empfehlung:
Nach dem Laden eines Modells sollten mindestens 512 MB bis 1 GB VRAM Reserve vorhanden sein.

English

1. LoKI

LoKI is a desktop assistant for running large language models (LLMs) locally. The program was developed in C++ using the Qt framework and utilizes the llama.cpp inference library. LoKI automatically loads the appropriate backend (CPU, CUDA, Vulkan, or OpenCL) depending on the hardware. This allows the program to run on both pure CPU systems and GPU-accelerated systems. Depending on what the program finds when it is first started, system drivers may be installed via an install script.

The AppImage contains all necessary runtime components such as Python, Node modules, and various runtime libraries. This allows LoKI to run largely independently of the Linux distribution.

Folder structure

When starting for the first time with --install, a working folder is created:

LoKI/

├─ Data (chat context database)
├─ Docs (documentation)
├─ Model (installed LLM models)
├─ ModelUseFiles (file access for LLMs)
├─ Plugins (MCP tools)
└─ Voice (TTS voices)

Data: This is where the sqlite3 database (also included in the AppImage) for the chat context is located. The chat context is on the left side of the LoKI main page and lists all chats that have been conducted. Saved chats can be reactivated and continued or deleted by right-clicking. If you are in a chat, you can switch to a new chat using the “+ New Chat” button.

Docs: This is where the documentation is stored.

Model: This is where the LLMs created by download or drag & drop are stored. When downloading, you can select from a list of available LLMs (Hugging Face) in the download area of LoKI. Pure .GGUF models can be loaded, but also HF models, whereby HF models are automatically converted to GGUF after download, as only GGUF models can be loaded. On the download page, it is also possible to requantize a GGUF model. This increases speed but at the expense of model accuracy. Further information can be found in the quantization documentation.

ModelUseFiles: This is the folder that the loaded model can use to create and manipulate files. In MCP mode “Local,” there are a few tools that enable this. The path to the folder can be freely selected in the settings and can also be placed elsewhere. Caution: Think carefully about what you allow an LLM to access!

Plugins: This folder contains the folders “local,” “stdio,” and “http,” which contain the sample tools. See “Tools Doc” for more information.

Voice: Here are a few voices for acoustic output. German and English voices are available. In this context, many thanks to Thorsten-Voice for his voice with freely selectable emotions that can also be set via the settings. (My thanks go to Thorsten-Voice)
Other voices can also be stored there. Just note that the voice is stored with its .json configuration file.

After starting LoKI for the first time with **–install**, you can then start the program directly from “Applications” or by double-clicking. A console is no longer necessary.

2. Editor

The editor is an integrated code and text editor. It allows you to edit plugin scripts, configuration files, and documentation. Syntax highlighting makes it easier to work with JSON, Python, or JavaScript.

The editor can transfer documents directly to the currently running model using the three buttons at the bottom, for example, to explain code, check it, or automatically generate documentation.

Tip: If you set the ModelUseFiles path in the settings to the Plugins/local, Plugins/stdio, or Plugins/http folder and then select “local” in MCP mode, the LLM can access the tools directly via the file tools, read them, extend them with the correct prompt, and so on.

Note: The plugins/tools in the folder are kept very simple and are only intended to demonstrate how LoKI works. For example, the Calc tool in the Local folder is very simple. However, it is ideal for extension by a useful LLM if you follow the tip above.

3. Download

The download area allows you to download models directly from HuggingFace. Both GGUF models and complete HuggingFace repositories can be downloaded.

HF models are automatically converted to GGUF format after download, as LoKI only uses GGUF models for inference. When downloading HF, a folder with the name of the model is created in the model folder, which bundles all files from the HF download. This folder is deleted after conversion (to save space) and the finished model is then available in GGUF format.

Note: Not all LLMs that are displayed can be downloaded. Some LLMs from the list are “gated” and therefore blocked. The download is then canceled. You can search for LLMs of your choice in the search bar.

Models tested with LoKI are: DeepSeek-R1-Distill-Llama-8B.gguf
Qwen3-8B-Q4_K_M.gguf (LoKI quantized)
Llama3.1-8b-f16 and Q4_K_M
gpt-oss-20b_f16 (packed—more on this in the quantization doc)

4. Quantization

The quantization section is located at the bottom of the download area. Quantization reduces the memory size of a model by storing weights with lower numerical precision. While a model in F16 format uses 16-bit floating-point weights, quantization formats such as Q8, Q5, or Q4 use compressed representations.

Typical advantages of quantization:

• Lower RAM/VRAM requirements

• Faster inference

• Smaller model files

Typical quantization levels:

F16 → Highest accuracy, largest memory requirements

Q8 → Very good quality with moderate memory reduction

Q5 → Good compromise between speed and quality

Q4_K_M → Very efficient for consumer hardware

Some modern models are already optimized internally (e.g., through tensor packing or shared tensors). In such cases, re-quantization can actually increase the file size because the packing is destroyed. One example

candidate is gpt-oss-20b_f16, which only has a size of 12GB instead of 30GB (estimated).

Select the LLM and quantization level here and start quantization. The LLM is then ready for testing.

5. Settings

Various LoKI parameters can be set on the Settings page. LoKI itself can be displayed in German/English or light/dark. This can be adjusted during runtime.

Speaker settings: The path to the voice directory containing the speech models. Emotions ID only applies to the “Thomas” voice model.

Download settings: HuggingFace token: You should generate such a token on the HF website. This allows for slightly more downloads. The token is stored in encrypted form. Here you can also select whether quantization should take place immediately after download and possible conversion (HF download), and the quantization level to be used in this case.

The settings allow you to adjust important inference parameters of the LLM.

n_ctx – Context size
Determines how many tokens are held in the context window at the same time. Larger values allow for longer conversations, but require more RAM/VRAM.

n_batch – Batch size
Controls how many tokens are processed at the same time.
Higher values increase speed, but require more memory.

n_gpu_layers – GPU layers
Determines how many transformer layers are calculated on the GPU.
0 = CPU only
-1 = as many layers as possible on the GPU

Temperature
Controls the creativity of the responses.
Low values → precise responses
High values → more creative responses

top_p and top_k
These parameters control the sampling of token probabilities and influence the diversity of the generated responses.

Enable MCP: Enables MCP mode and displays the MCP window on the main page.

MCP mode: You can select stdio, local, or http. Http then via the MCP endpoint.

Note: The AI settings should not be changed while LLM is running. This will not affect the running thread. The parameters are used when loading the LLM.

Note: The AI settings are dynamically monitored by LoKI when loading the LLM and may be downgraded if they are not suitable. This is then displayed after loading the LLM. This downgrade is somewhat conservative (safe) and can be increased slightly.

6. Tools

LoKI supports MCP tools (Model Context Protocol). These tools can integrate local programs, web APIs, or automations into the context of the LLM.

Here are the tools for the individual MCP modes. These are examples and are not fully developed in some areas. This is intentional so that you can at least see how the tools work and can tackle the development or expansion of new tools yourself. More on this in the editor documentation.

All tools use the jsonrpc 2.0 protocol.

In the Plugins folder, you will find the following folders:

local: This is where the tools for the local MCP mode are located. These are examples and are written in Python. The Python tools also use the Python312 environment from the AppImage so that there are no system dependencies. This folder also contains tools that access the ModelUseFiles folder and can be used by the LLM.

stdio: This folder contains tools that allow more system access. These include tools that use the browser (ms-playwright) integrated in the AppImage. The integrated Node modules are also used here. Please note that all tools in this folder use an mcp-tool-loader that scans the tools. For your own tools, please refer to the existing tools to ensure functionality.

http: Here is an example of a functioning server. This server is automatically started by LoKI in this mode.

The respective mode is set via the settings page. The preselection must be selected before starting the LLM. While the LLM is active, it is not possible to change the MCP mode because the plugins/tools are scanned when the LLM is started, depending on the mode, and the LLM is then announced in the context. The plugins/tools found are displayed on the main page on the right-hand side in the “Loaded Plugins” window.

7. System Monitor

The System Monitor displays CPU, RAM, and GPU usage in real time. VRAM monitoring is particularly important for GPU inference.

This page can be undocked with F9 and placed on another monitor so that you can evaluate the behavior of the system during a chat. It can be docked again with F9 or simply by clicking “Close.”

Recommendation:
After loading a model, you should have at least 512 MB to 1 GB of VRAM reserve available.