

## # README (Allgemein) - KI-LAN-Party

Willkommen zur KI-LAN-Party Arbeitsumgebung! 🙌

Dieses Repository führt Sie in die Einrichtung einer sofort einsatzbereiten Demo-Umgebung für eine KI-LAN-Party mit drei Prototypen: Zwei KI-Agenten basierend auf einem Low-Code-Framework und einen Sprachassistenten.

Die Software läuft lokal auf den Teilnehmerrechnern bzw. bereitgestellter Hardware. Das Setup liefert klare Templates und Anleitungen, damit Teilnehmer selbst Experimente durchführen können.

### ### Kurzbeschreibung der verwendeten Software

- Docker: Docker ist eine Container-Plattform, die Anwendungen inklusive ihrer Abhängigkeiten in isolierten Containern verpackt.
- Dify: Low-Code-Framework zur Erstellung KI-gesteuerter Prototypen; liefert Vorlagen, Systemprompts und einfache Integrationen für KI-Agenten.
- Open WebUI: Grafische Oberfläche zur Interaktion mit lokal laufenden Modellen; ermöglicht einfachen Modellwechsel und -anpassungen.
- Ollama: Lokaler Runner zur Bereitstellung von Modellen (z. B. llama3:8b, qwen3:8b); ermöglicht die direkte Ausführung von KI-Modellen ohne Cloud-Anbindung.

### ## Setup

#### ### 1) Ollama installieren und Modell vorbereiten

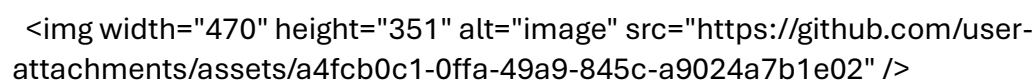
- macOS: Ollama-Installer von ollama.com / Homebrew installieren, dann -> `brew install ollama`
- Windows: Ollama-Installer von ollama.com
- Linux: curl-Install gemäß ollama.com

#### #### 1.1) Sprachmodelle über Terminal herunterladen

- (über Ollama App UI: folge Schritt 1.2)
- über Terminal-Befehl: `ollama pull llama3.1:8b` und danach `ollama pull qwen3:8b`
- Testen: `ollama list` (zeigt installierte Modelle)

#### #### 1.2) Sprachmodelle über Ollama UI herunterladen

- Starte Ollama
- Suche die gewünschten Modelle (`llama3:8b` und `qwen3:8b`):



- Sende eine Nachricht im Chat, um den Download zu starten



- Lade beide Modelle herunter (`llama3.1:8b` und `qwen3:8b`)

### 2) Docker Desktop installieren (von docker.com)

- Runterladen, installieren und starten

> \*\*Bei MacBooks kommt manchmal eine Fehlermeldung, dass Rosetta installiert werden muss. Einfach aufs Fenster klicken, dann auf Installieren gehen. Falls eine Fehlermeldung kommt, einfach jedes mal auf 'Retry' gehen. Wenn sich Docker geschlossen hat, heißt es, Rosetta wurde erfolgreich installiert. Docker noch mal manuell starten.\*\*

### 3) Dify lokal starten

- Repository klonen:

- `git clone https://github.com/langgenius/dify`

- In den Ordner wechseln, der die docker-compose.yaml enthält `cd dify/docker` (für mehr Infos, siehe Dify-README).

- neue .env Datei anlegen(oder .env.example Datei duplizieren und Namen ändern):

- falls man die Datei manuell erstellen will im Finder/Explorer, muss man erst

versteckte Dateien anzeigen lassen.

- MacBook: Command + Shift + . (Punkt)

- Windows: File Explorer öffnen mit Windows-Taste + E > Ansicht/View > Anzeige

Versteckter Inhalte klicken

- Kopiere den Inhalt von .env.example nach .env

- Start:

- `docker compose up -d`

- UI öffnen:

- http://localhost:3000 oder einfach `localhost` als URL eingeben und Enter drücken

- Admin-Konto anlegen.

> Im Hintergrund läuft ein Docker Container; hier wird Dify gehostet. Um den Container zu stoppen, einfach Docker Desktop öffnen > Containers > und den Container namens `docker` stoppen (auf den Stopp-Button klicken unter Actions).

> Um den Container wieder zu starten, einfach auf den Start Button klicken unter Actions.

---

> ## 🚦 Wähle: Vorlagen vs. Manuelles Setup

>

> \*\*Du willst den schnellsten und einfachsten Einstieg?\*\*

> 📖 \*\*Springe direkt zum [Agent 1 README](./Agent\_1/README.md) und nutze die fertigen Templates.\*\*

>

> \*\*Du möchtest alles selbst lernen und individuell anpassen?\*\*

> 📖 \*\*Lies unten weiter für eine Schritt-für-Schritt-Anleitung zum manuellen Setup.\*\*

> \*(Hinweis: Das ist fortgeschrittener und dauert länger, bietet dir aber volle Kontrolle.)\*

---

#### ### 4) Dify mit Ollama verbinden (OpenAI-API-kompatibel) 🦨

Im Dify-UI:

- Settings:



- Modellanbieter -> Suche nach "OpenAI-API-compatible" -> Modell hinzufügen



Felder ausfüllen (Beispiel: llama3.1:8b):

- Model Type: LLM
- Model Name: `llama3.1:8b`
- Authorization Name: `Authorization`
- Model display name: `Llama 3.1 8B (Ollama)` [frei wählbar]
- API Key: `ollama` [beliebiger String, wird von Ollama nicht geprüft]
- API endpoint URL:
  - macOS/Windows: `http://host.docker.internal:11434/v1`
  - Linux: `http://172.17.0.1:11434/v1` (oder das Gateway aus `docker network inspect bridge`)
- model name for API endpoint: `llama3.1:8b` [muss exakt zum Ollama-Tag passen]



#### ### Häufige Probleme:

- Manchmal kann es bei Windows in der Kombination von WSL, docker und dify zu Verbindungsproblemen kommen. In diesem Fall hilft:
  - Setze die Umgebungsvariable `OLLAMA_HOST` auf `0.0.0.0:11434`
  - Verwende die URL `http://<WSL-IP>:11434/v1` in Dify
  - Die WSL-IP findest du mit `ipconfig` in der PowerShell heraus
  - Danach Ollama neu starten

> ### Gut gemacht! Du hast die initiale Einrichtung abgeschlossen.

> ### Du kannst jetzt zum [Agent 1 README](./Agent\_1/README.md) wechseln.

> Oder, falls du noch etwas unsicher bist, lies die Nutzung- / FAQ-Bereiche unten. 📖

## ## Nutzung und Konfiguration

### ### Chat-Format

- Der KI-Agent arbeitet im Chat-Format, das bedeutet, jedes neue Chat ist wie eine neue Unterhaltung. Du kannst die 'Memory'-Funktionen in den Einstellungen aktivieren, damit der Agent sich an Dinge aus anderen Chats und Unterhaltungen erinnert, aber das verbraucht mehr Ressourcen – und manchmal möchte man einfach jedes Mal neu beginnen.

### ### Mehrere Agenten

- Dify ermöglicht es dir, mehrere Agenten zu erstellen – das bedeutet, dass du für verschiedene Anwendungsfälle je einen dedizierten Agenten erstellen kannst. Du kannst Agenten von der Haupt-Dify-Seite aus erstellen (klicke auf das Dify-Logo).

### ### Wie man Prompts schreibt (Prompt-Engineering)

- Du kannst den Prompt ändern so wie du möchtest. Der aktuelle Prompt ist nur ein Beispiel. Dein KI-Agent könnte ein Anwalt, Lehrer, Stratege oder etwas anderes sein. Bedenke, dass je detaillierter der Kontext ist, desto besser und genauer wird er sein. Stell dir vor, dies ist eine Person, die weit entfernt lebt und nichts über dich oder dein Vorhaben weiß. Und du chattest online mit dieser Person (die Experte in vielen Bereichen ist), also musst du dieser Person einen detaillierten Hintergrund des Problems geben.

### ### Platzhalter-Template-Prompt

- Falls du eine Platzhalter-Template als Starter-Prompt benötigst, kannst du das AUTOMAT-Framework verwenden, es gibt aber viele, die du online finden kannst. Hier findest du einige grundlegende Beispiele: <https://medium.com/the-generator/the-perfect-prompt-prompt-engineering-cheat-sheet-d0b9c62a2bba>

### ### Variablen

- Du kannst Variablen für den Prompt hinzufügen. Innerhalb des Prompt-Texts fügst du einfach zwei geschweifte Klammern mit einem Variablennamen dazwischen (`{{name}}`) hinzu und fügst es anschließend als Variable im Dify-Menü unter dem Prompt-Text hinzu. Abhängig davon, wie du die Variable in deinem Prompt eingebunden hast, wird der Agent diese Variable entsprechend verwenden. Ein einfaches Beispiel zum Testen: ``Nenne mich bei meinem Namen {{name}}``.

### ### Hochladen von Dokumenten

- Es ist möglich, Dokumente hochzuladen, die der KI-Agent später als Informationsquelle verwenden wird. Das wäre z. B. nützlich, wenn du eine große PDF-Datei mit vielen Informationen zu einem bestimmten Thema hast. Du könntest sie hochladen und dem KI-System konkrete Fragen zum Thema stellen. Oder das gesamte Dokument zusammenfassen. Oder Rechtschreibung und Grammatik verbessern. Es gibt viele Anwendungsfälle.

- Um dies zu tun, gehe zum Knowledge-Tab im Hauptmenü (obere Leiste) > Knowledge erstellen > Import from file > Datei auswählen/Drag&Drop > Weiter > Hochwertig > Speichern & Verarbeiten. Nachdem du es dort hochgeladen hast, kannst du in deinem

Chat in der Studio-Ansicht das Knowledge-Dropdown links (nicht oben in der Leiste) verwenden und einfach das zuvor hochgeladene Dokument auswählen. Nun kennt der KI-Agent alles, was in diesem Dokument steht.

#### ### Weitere Funktionen

- Es gibt viele weitere mögliche Funktionen, die etwas mehr Zeit für die Einrichtung benötigen (z. B. Vision-Funktion zur Analyse hochgeladener Bilder). Wenn du etwas Erfahrung hast und die zusätzliche Zeit investieren möchtest, kannst du versuchen, sie einzurichten.

#### ### Einschränkungen

- Es gibt Einschränkungen hinsichtlich der Menge an Informationen, die der KI-Agent verarbeiten kann. Wenn du lange genug mit dem Agenten chatst, fängt er an zu vergessen, worüber du am Anfang gesprochen hast, und könnte im Verlauf verwirren. Wenn du außerdem Dokumente hochlädst, die der KI-Agent als Informationsquelle verwenden soll und die Dokumente zu groß sind, könnte der Agent ebenfalls Dinge vergessen oder Kontext verlieren. Du kannst die genauen Grenzen durch Ausprobieren herausfinden.

- Die KI-Agenten sind nicht perfekt und sie "halluzinieren" oft. Du solltest die von ihnen erhaltenen Informationen daher unbedingt überprüfen, da sie dir falsche Antworten geben könnten.

### ## FAQ & Troubleshooting 🌿

- Warum funktioniert die URL `http://host.docker.internal:11434/v1`? 🔍

- Dify läuft in Docker-Containern. Damit ein Container (Dify) den Ollama-Dienst auf deinem Host-Rechner erreicht, gibt es `host.docker.internal` (auf macOS/Windows). Das ist eine spezielle Hostname-Brücke vom Container zum Host.

- Der `/v1`-Pfad: Ollama bietet eine OpenAI-kompatible API unter dem Pfad `/v1`. Dify erwartet genau dieses Schema. Deshalb muss die Base-URL auf `/v1` enden.

- Linux-Sonderfall: `host.docker.internal` ist dort oft nicht vordefiniert. Nimm stattdessen die Gateway-IP der Docker-Bridge (häufig `172.17.0.1`). Du findest sie mit: `docker network inspect bridge` und suchst nach "Gateway".

- Es gibt keinen "Chat"-Typ bei "Model Type".

- Korrekt. In Dify heißt das einfach "LLM". Das ist der Chat/Text-Generierungstyp.

- 404 Not Found bei API-Aufruf.

- Meist fehlt `/v1` am Ende der Base-URL. Nutze z. B.  
`http://host.docker.internal:11434/v1`

- Verbindung fehlgeschlagen von Dify zu Ollama.

- macOS/Windows: Nutze `host.docker.internal`, nicht `localhost`.

- Linux: Nutze `http://172.17.0.1:11434/v1` oder deinen Bridge-Gateway-Host (`docker network inspect bridge`).

- Prüfe, ob Ollama läuft: `ollama list`

- “Model not found” oder “The model does not exist”.
  - Der Eintrag “model name for API endpoint” muss exakt dem Ollama-Tag entsprechen (z. B. llama3.1:8b). Prüfe mit `ollama list` oder `curl http://localhost:11434/api/tags`
- Was trage ich bei “API Key” ein?
  - Irgendeinen nicht-leeren String (z. B. ollama). Ollama prüft das nicht, Dify will aber ein Feld.
- Ist .yml und .yaml dasselbe?
  - Ja. Beides ist YAML. Docker Compose akzeptiert beide. Falls der Dateiname abweicht, nutze `docker compose -f datei.yaml up -d`
- Responses sind langsam.
  - Kleinere Modelle nutzen.
  - In Dify die Max-Token reduzieren.
  - Genug RAM/CPU sicherstellen.
- Kann ich ein anderes Modell nutzen?
  - Ja. In Ollama den passenden Tag ziehen (`ollama pull ...`), danach in Dify dasselbe Vorgehen: Model Name und model name for API endpoint identisch zum Tag.
- “localhost” funktioniert nicht aus Dify.
  - Dify läuft im Container. localhost meint dann den Container selbst, nicht deinen Host. Daher `host.docker.internal` (Mac/Win) bzw. Bridge-Gateway (Linux).
- Linux: Wie finde ich die Gateway-IP?
  - `docker network inspect bridge` und nach "Gateway" suchen. Typisch ist 172.17.0.1
- Port-Konflikte beim Start von Dify.
  - Prüfe, ob 3000 (Dify UI) oder weitere in der Compose belegte Ports frei sind. Passe Ports in der `docker-compose.yaml` an, falls nötig, und starte neu.
- Ich habe versehentlich Knowledge/RAG aktiviert.
  - In der App die Knowledge-Sektion entfernen bzw. sicherstellen, dass kein Datenspeicher verbunden ist. Für Agent 1 brauchen wir kein RAG.
- Wie exportiere/importiere ich das Template?
  - In der App: Export Template -> Datei sichern. Import analog über Import Template.