Contents

[1. System architecture design 1](#_Toc206071126)

[2. Environmental preparation 1](#_Toc206071127)

[1. System configuration 1](#_Toc206071128)

[3. Dependent installation 2](#_Toc206071129)

[1. System-level dependencies 2](#_Toc206071130)

[2. Python environment 2](#_Toc206071131)

[4. Project structure 3](#_Toc206071132)

[5. System service configuration 3](#_Toc206071133)

[1. systemd service file 3](#_Toc206071134)

[6. Deployment process 4](#_Toc206071135)

[Generate random API key (32-byte HEX) 4](#_Toc206071136)

[Validate the deployment 4](#_Toc206071137)

[7. Testing 4](#_Toc206071138)

[8. Ollama management 5](#_Toc206071139)

[9. Comparison of language models 5](#_Toc206071140)

[10. Client Configuration 6](#_Toc206071141)

# 1. System architecture design

[Client] --> (REST API) --> [Ubuntu server]

├─ FAISS vector search

├─ LLM Inference

└─ Automatic knowledge base updates

# 2. Environmental preparation

## 1. System configuration

# Ubuntu 22.04 LTS base configuration // could be compatible with Ubuntu 24.04 LTS

sudo apt update && sudo apt upgrade -y

# 3. Dependent installation

## 1. System-level dependencies

# Basic tools

sudo apt install -y \

build-essential \

cmake \

git-lfs \

nvidia-cuda-toolkit \

cgroup-tools \

inotify-tools \

poppler-utils \

tesseract-ocr \

cgroups

# Performance optimization components

sudo apt install -y \

libjemalloc-dev \

libnuma-dev \

ocl-icd-opencl-dev

# ollama installation

sudo snap install ollama

ollama pull deepseek-llm:7b

ollama serve

## 2. Python environment

# Create a virtual environment

sudo apt install python3-venv # ensure venv module is installed

python3 -m venv /home/user/localkb

source /home/user/localkb/bin/activate # activate it

deactivate # exit the virtual environment

# Core dependencies

pip install \

torch==2.1.0+cu118 \

faiss-gpu==1.7.4 \ # if it is not available, use faiss-cpu

langchain==0.0.346 \

fastapi[all] \

uvicorn[standard] \

unstructured[pdf,docx] \

sentence-transformers

# System optimization library

pip install \

psutil \

python-inotify \ # no compatible version for python 3.12

numa

# 4. Project structure

/home/user/localkb/

├── app/

│ ├── config.py # System configuration

│ ├── file\_monitor.py # File change monitoring

│ ├── knowledge\_processor.py # Knowledge processing core

│ ├── main.py # Main Service Entrance

│ ├── ollama\_client.py # Ollama client package

│ ├── prompt\_builder.py # prompt configuration

│ ├── vector\_manager.py # Vector store update

│ └── utils/

├── data/

│ ├── knowledge/ # Original knowledge base file

│ └── vectors/ # FAISS vector storage

├── logs/

│ ├── application.log # API logs

# 5. System service configuration

## 1. systemd service file

# /etc/systemd/system/knowledge.service

[Unit]

Description=Knowledge Service

After=network.target ollama.service

[Service]

User=knowledge

Group=knowledge

WorkingDirectory=/home/user/localkd

ExecStart=/home/user/localkb/bin/uvicorn app.main:app --host 0.0.0.0 --port 8000

Restart=always

MemoryLimit=8G

CPUQuota=200%

IOWeight=100

LimitNOFILE=65535

[Install]

WantedBy=multi-user.target

# 6. Deployment process

## Generate random API key (32-byte HEX)

openssl rand -hex 32 | sudo tee /home/user/localkb/key/api.key

## Validate the deployment

curl -X POST http://localhost:8000/api/ask \

-H "Content-Type: application/json" \

-H "X-API-Key: $(cat /home/user/localkb/key/api.key)" \

-d '{"question":"please summarize the todo items."}'

# 7. Testing

# Manually started

uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload

# Test the API interface

curl -X POST http://localhost:8000/api/ask \

-H "X-API-Key: $(cat /etc/knowledge-service/api.key)" \

-H "Content-Type: application/json" \

-d '{"question":"How do I configure firewall rules?" }'

Run from a Windows machine

curl -X POST http://10.75.62.60:8000/api/ask -H "X-API-Key:40622223ee0fc66a6622deefdfd86b512d834bd4990779966c9a5d9ee437b247" -H "Content-Type: application/json" -d "{\"question\ ":\"summarize todo items\"}"

# Stress test

apt install siege

siege -c 100 -t 1M http://localhost:8000/api/ask

# 8. Ollama management

# Download the DeepSeek model

ollama pull deepseek-llm:7b or

ollama pull llama3.2 (3b parameters).

# Validate the list of models

ollama list

# Model operation monitoring

watch -n 1 ollama ps

# 9. Comparison of language models

| **model** | **English concentration** | **Hardware requirements** | **Applicable scenarios** |
| --- | --- | --- | --- |
| **Llama 2** | ⭐⭐⭐⭐ | medium | General Text Processing |
| **Mistral 7B** | ⭐⭐⭐⭐⭐ | low | Instruction following, real-time response |
| **Falcon 180B** | ⭐⭐⭐⭐⭐ | Extremely high | High precision and complex tasks |
| **CodeLlama** | ⭐⭐⭐⭐ | medium | Technical documentation, code mix |
| **OpenChat** | ⭐⭐⭐⭐ | low | Conversational interactions |

# 10. Client Configuration

1. Compile the C# solution using Visual Studio 2017 or later
2. Run the application, and specify the service IP and the API key that matches the one configured at the server
3. Recommend keeping the stream checkbox checked, and run it. Depending on the server resource, the response could be available in minutes

Note about color-coding in the response:

* Yellow – process message
* Blue – real response