加餐二：ChatOps 开发实战

王炜/前腾讯云 CODING 高级架构师

目录

1. Open Web UI

Open Web UI 简介

Open WebUI 是一个可扩展、功能丰富且用户友好的自托管 AI 界面，旨在完全离线运行。

它支持各种 LLM 供应商，包括 Ollama 和 OpenAI 兼容 API。

Open Web UI 核心功能

- 支持本地 Docker 部署和 K8s 部署（Helm）
- 支持细粒度的用户角色和权限配置、RBAC 访问控制
- 支持 RAG
- 支持网络搜索
- 支持浏览网页并进行聊天
- 支持 Markdown 和网页渲染

本地运行 Open Web UI

1. 以使用 OpenAI API 的方式启动

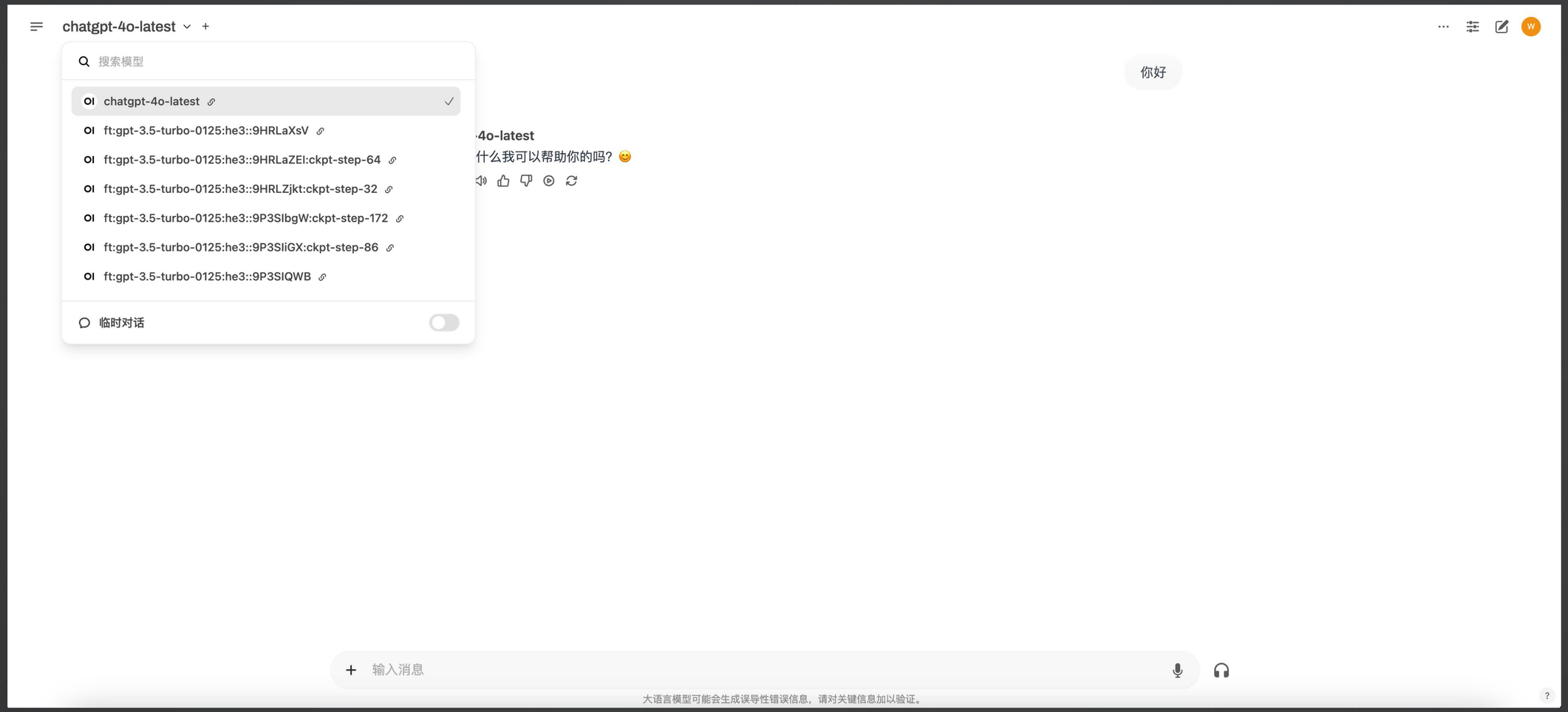
   - docker run -d -p 3000:8080 -e OPENAI_API_KEY=your_secret_key -v open-webui:/app/backend/data --name open-webui --restart always ghcr.io/open-webui/open-webui:main

2. 以同时使用 Ollama 方式启动

   - docker run -d -p 3000:8080 --gpus=all -v ollama:/root/.ollama -v open-webui:/app/backend/data --name open-webui --restart always ghcr.io/open-webui/open-webui:ollama

3. 访问 http://localhost:3000

Open Web UI

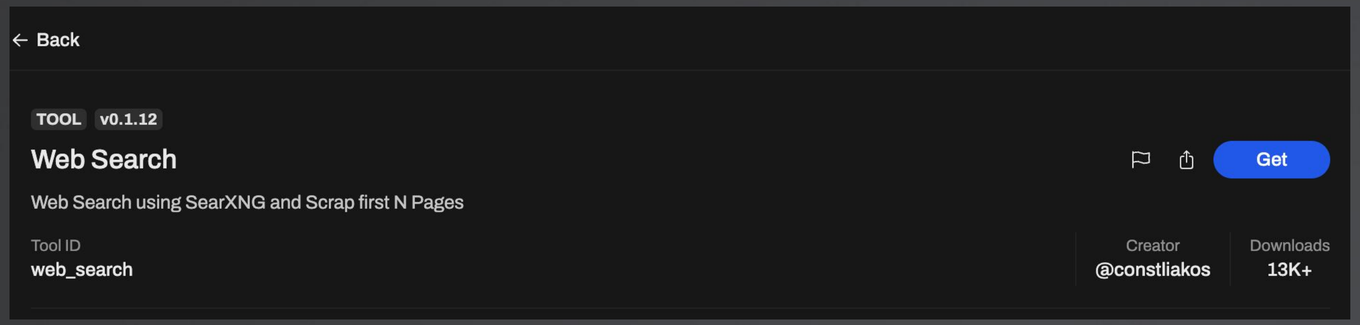

chatgpt-4o-latest
搜索模型
chatgpt-4o-latest
ft:gpt-3.5-turbo-0125:he3::9HRLaXsV
ft:gpt-3.5-turbo-0125:he3::9HRLaZEl:ckpt-step-64
ft:gpt-3.5-turbo-0125:he3::9HRLZjkt:ckpt-step-32
ft:gpt-3.5-turbo-0125:he3::9P3SlbgW:ckpt-step-172
ft:gpt-3.5-turbo-0125:he3::9P3SliGX:ckpt-step-86
ft:gpt-3.5-turbo-0125:he3::9P3SlQWB
临时对话
你好
4o-latest
什么我可以帮助你的吗?
输入消息
大语言模型可能会生成误导性错误信息，请对关键信息加以验证。

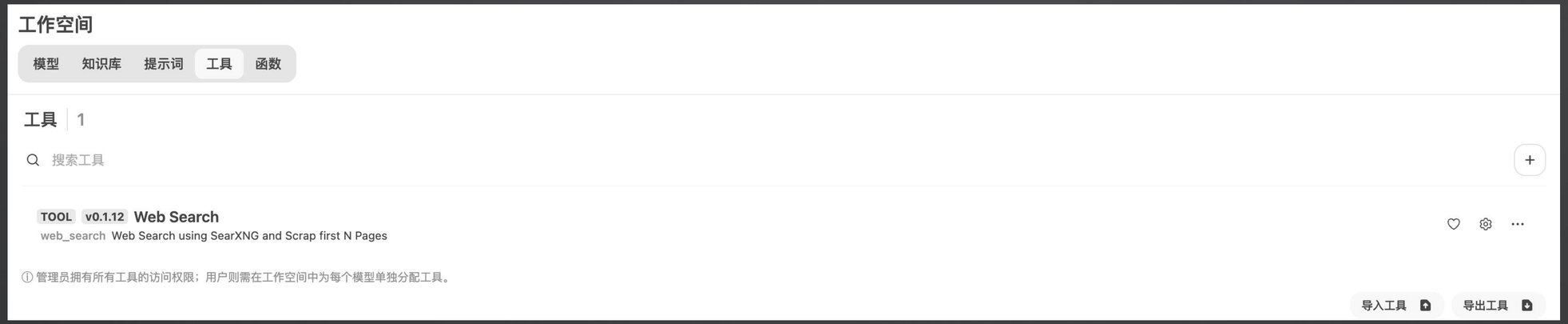
Open Web UI Tools

- Open Web UI 支持通过工具的方式增强其能力
- 需要模型支持函数调用
- https://openwebui.com/tools
- 例如 Web Search 工具

安装 Web Search

在网页中通过跳转到 Open Web UI 的方式直接安装：





如何使用 Tools?

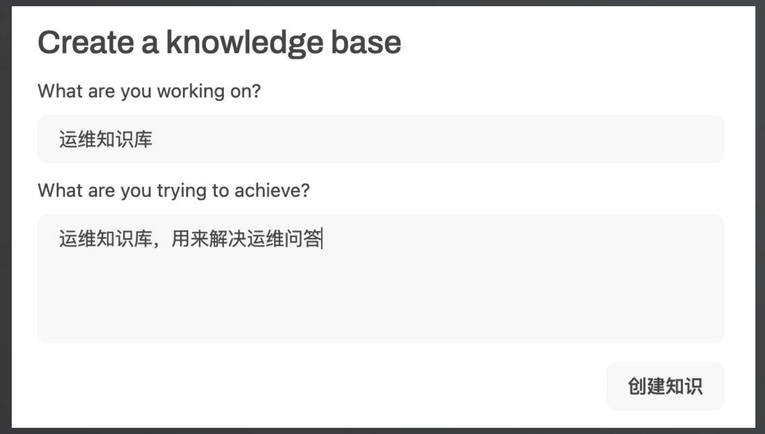
在聊天中启用 Tools，LLM 会根据上下文决定是否需要使用 Tools，以及自动填充 Tools 所需要的参数

例子：

1. 启用工具，输入提示语：访问这个链接：https://36kr.com/p/3064705365435520，并简单总结内容
2. 不启用工具，测试相同的提示语

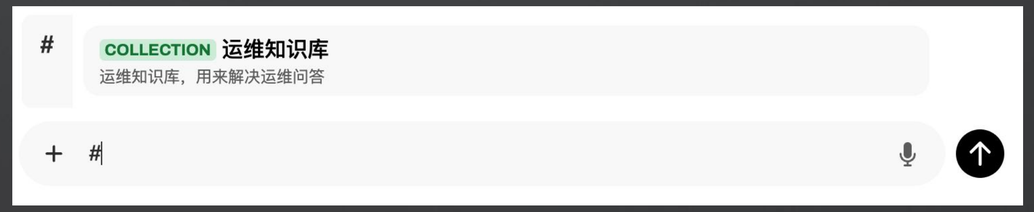
本地 RAG

1. 创建知识库



2. 上传文件

3. 在聊天中使用：通过#号引用



远程 RAG

1. 无需提前 Embedding，直接通过获取远程链接的内容进行问答

2. 通过#+网页链接启用

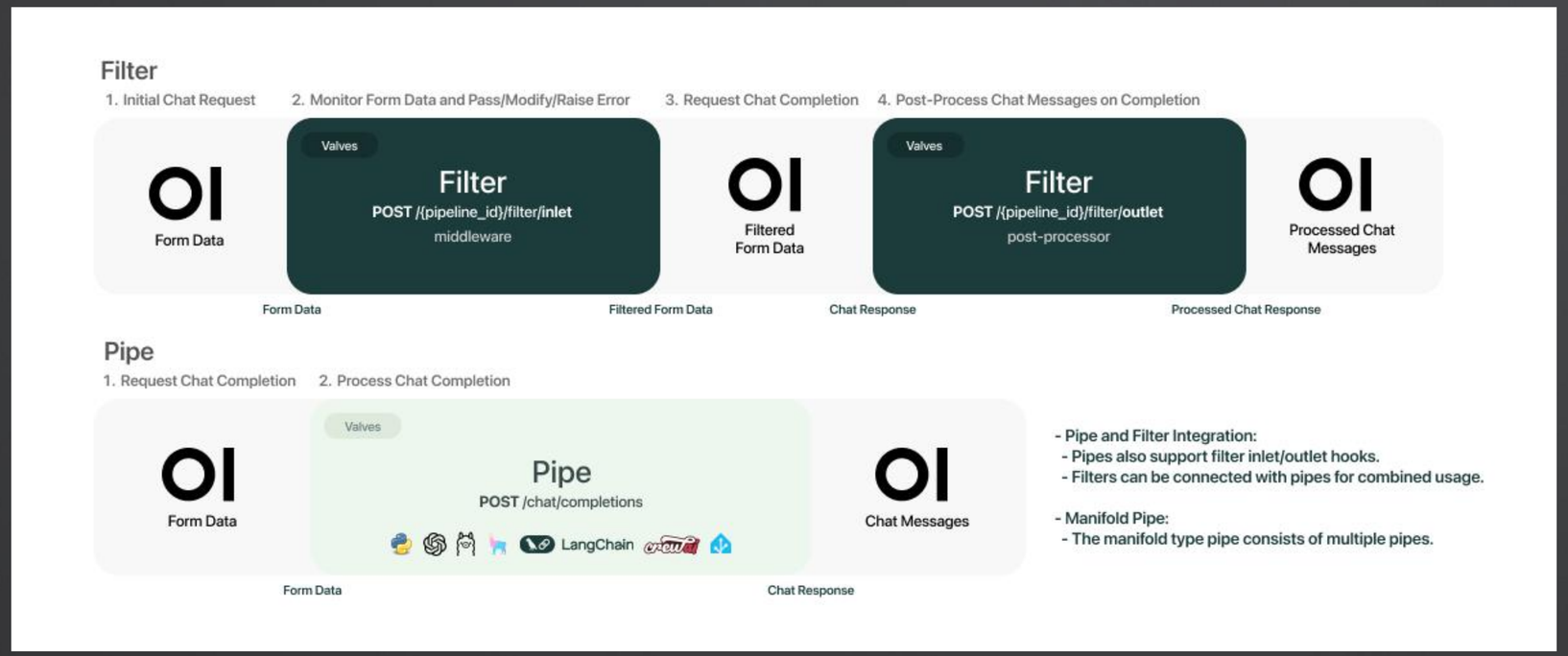
3. 缺点：相比较 Web Search Tools 爬取网页能力较弱

2. Pipeline

Pipeline 介绍

Pipeline 是扩展 Open Web UI 的方式，可以在不修改 Open Web UI 的前提下对输入和模型输出实现复杂的编排

编排能力只要是通过 Pipeline Filter 机制来实现的，允许你通过 Python 代码处理来自 Open Web UI 的输入和模型的输出

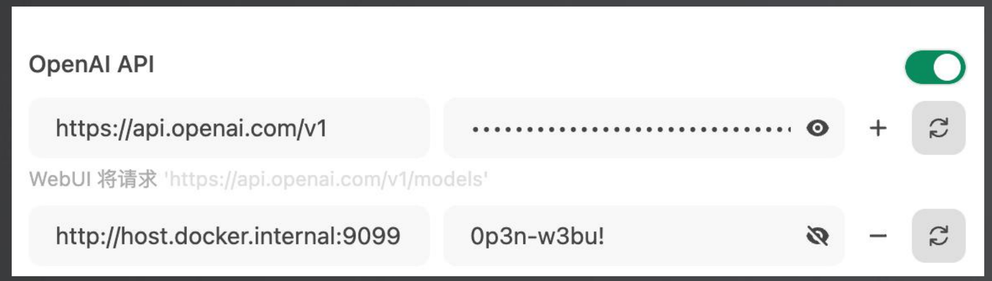


本地开发 Pipeline

1. git clone https://github.com/open-webui/pipelines.git
2. cd pipelines
3. pip install -r requirements.txt
4. sh ./start.sh

将 Open Web UI 与 Pipeline 连接

- 进入 Open WebUI 中的管理面板 > 设置 > 外部连接
- 点击 + 创建新的连接



- http://host.docker.internal:9099, 0p3n-w3bu!
  - （因为 Open Web UI 是通过 Docker 方式启动的，而 Pipeline 是通过本地源码方式启动的）
- 进入 Pipeline 模块验证

从零理解 Pipeline

- pipelines/examples/filters/function_calling_filter_pipeline.py
  - 可以自定义参数，例如：OPENWEATHERMAP_API_KEY，会作为输入框显示
  - 包含有 3 个 Tools
- 将该文件上传到 Pipeline 模块 
- 填充 openweathermap.org api key
- 保存

测试：

- 现在时间（调用 get_current_time）
- 今天 shenzhen 天气（调用 get_current_weather）
- 100*100 等于多少（调用 calculator）

工作原理

- sh dev.sh 进入开发模式（自动重载）
- 创建 .vscode/launch.json 文件，并黏贴调试配置
- 进入调试
- 在 pipelines/blueprints/function_calling_blueprint.py 文件 102 行打断点
  - Chat：现在时间
- 工作原理：将文件内定义的 Tools 进行格式化，然后放在系统提示语，让模型做出选择和填充参数
- 为什么？为了适配原生不支持 Function Calling 的模型，所以采用系统提示语进行兼容
- 缺点：如果有很多 Tools 定义会极大降低准确率

改造为 OpenAI 原生函数调用

- 修改 pipelines/blueprints/function_calling_blueprint.py 文件
- 替换为 OpenAI 原生函数调用
- 重新 Debug，在返回 tool_calls 的地方打断点
- 仍然用“现在时间”检查改造后的效果

3. ChatOps 开发实战

ChatOps Demo Application

例子：

- 示例应用部署在 K8s，每 2S 输出一批错误日志
  - 通过 NodePort 暴露服务: http://IP:32000
- 通过 Loki 采集日志信息（Grafana 查询演示）
  - 通过 NodePort 暴露 Grafana: http://IP:31000, admin, loki123
- 提供 Terraform IaC 代码一键创建环境（开通虚拟机、初始化 K8s 集群、Loki、Grafana 和示例应用）

编写 ChatOps Tools

- 复制一个 function_calling_filter_pipeline.py
- 添加工具：
  - 从 Loki 获取日志
  - 针对日志问题发送飞书消息
  - 获取集群工作负载 YAML
  - 更新（部署）工作负载
  - 部署新的开发环境
  - 创建 CVM 虚拟机

安装依赖

- 安装相关依赖
  - kubernetes==29.0.0
  - tencentcloud-sdk-python-cvm==3.0.1210
- 开通 CVM，并安装 K3s，将 config.yaml 放到 resource 目录下

AI ChatOps Demo

例子：

- 检查一下最近 app=payment 服务的日志是否有报错
- 输出日志原文
- OOM 这个问题很严重，在飞书@王炜通知他关注这个问题
- （怀疑资源配置不足）检查一下 payment 工作负载和容器的 resource 配置，输出 YAML，并给我一些建议
- （根据大模型建议）把 payment 容器的 resource 增加一倍，忽略创建过程，直接部署到生产环境
- 在新的命名空间 test-oom 拉一套新的环境，忽略创建过程直接输出访问链接，我要继续深入问题
- 帮我创建一台腾讯云2c4g的虚拟机，并输出腾讯云返回的 JSON 对象

THANKS