# LLM应用概要设计

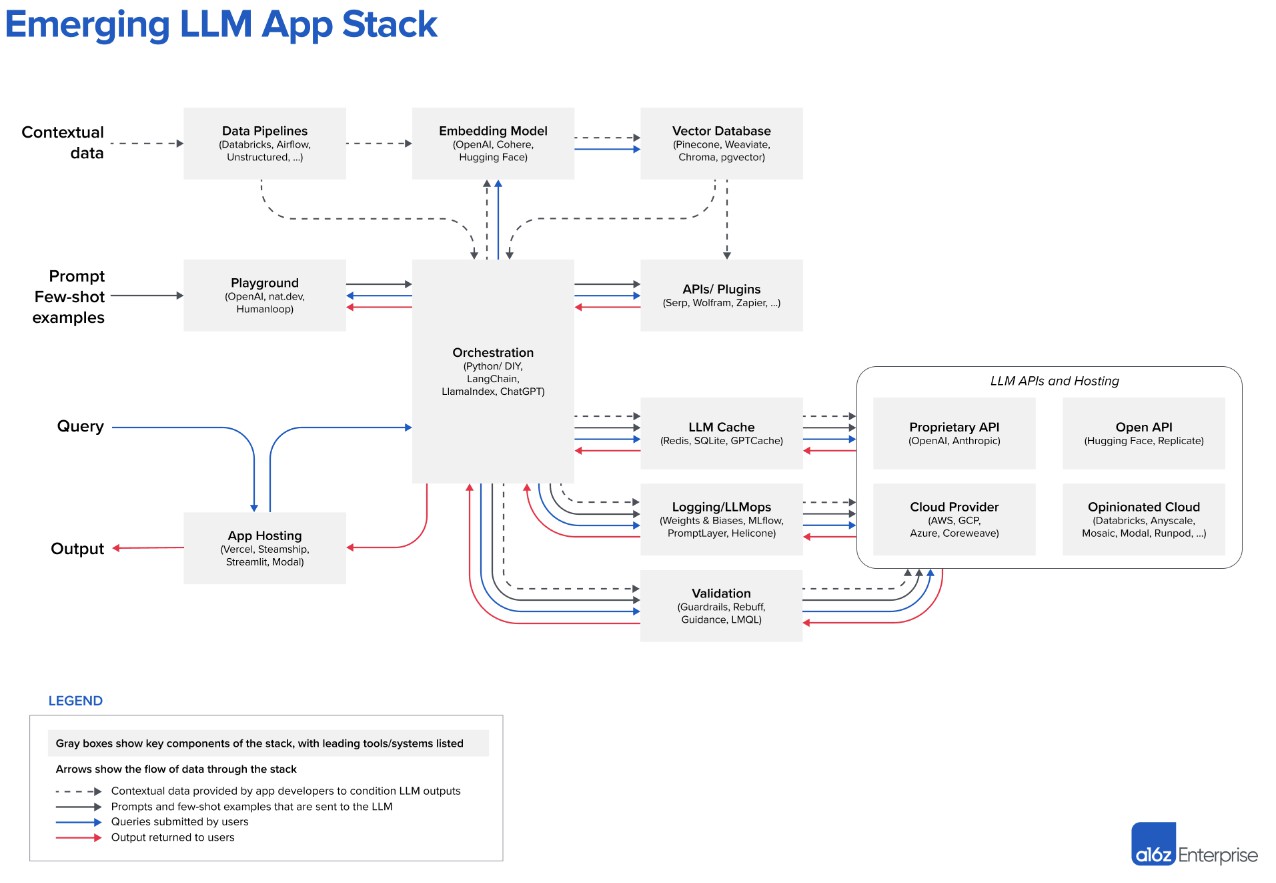
## 概述

1. 设计目标

完成LLM应用架构搭建，支持文档问答、数据分析问答、API调用的agent能力。

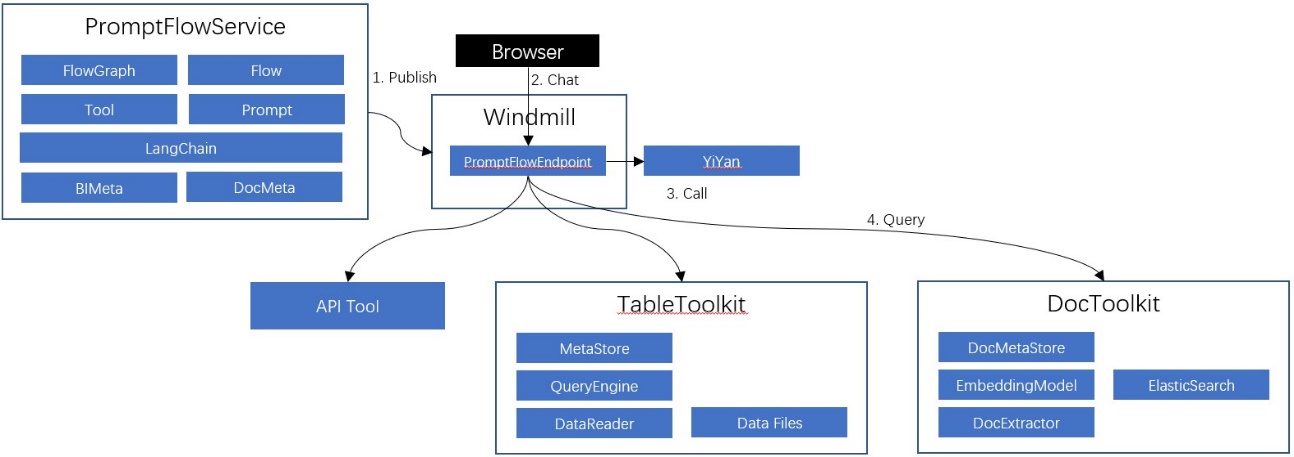
## 架构设计

架构参考a16z的LLM Stack



Orchestration选择LangChain。其中需要的Prompt、Chain、Tool等在PromptFlowService中保存和管理。LLM API支持一言和其他适合调试的开源LLM。Embedding Model使用一言。

对BI场景，由于需要支持MySQL、Oracle等数据库，以及excel、csv等数据文件，考虑尽量用纯查询引擎的方案，避免数据需要导入数据库。



系统由5部分组成：

* PromptFlowService负责管理LangChain用到的Prompt、Chain和Tool。并支持访问历史问答记录。
* Meta数据暂时放在PromptFlowService中，提供对BI依赖的数据源，和文档库元数据管理。
* 编排好的Chain，可以通过Windmill发布一个基于FastAPI的Endpoint对外提供服务。
* DocTool提供文档解析，抽取，向量构建的离线工具。
* BITool提供查询引擎，数据访问类库，数据转换和导入工具。
* API Agent提供外部API接入支持调用的功能，这是一类模块，每个接入的系统一个API Agent。这块缺少一个工作流方案选型。

参考资料

* [HumanLoop](https://humanloop.com/)
* [Vellum](https://www.vellum.ai/)
* [AzureML PromptFlow](https://learn.microsoft.com/en-us/azure/machine-learning/prompt-flow/overview-what-is-prompt-flow?view=azureml-api-2)
* [LangChain Vector stores](https://python.langchain.com/docs/modules/data_connection/vectorstores/)
* [Awesome-ai-agents](https://github.com/e2b-dev/awesome-ai-agents)

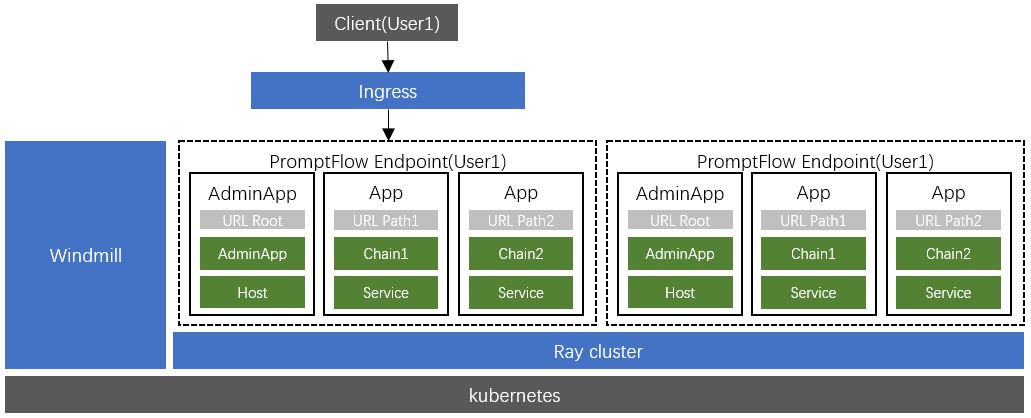
### PromptFlow Endpoint

EndpointService负责Model在线服务的部署与管理。Model、ModelServer以及部署策略通过Deployment描述，部署到Endpoint提供HTTP、GRPC等形式的在线推理服务。

一个Endpoint加载多个chain的代码，提供WebSocket形式的ChatAPI。

1. 提供LangChain服务托管，支持租户间隔离。
2. 可以创建多个Endpoint，每个Endpoint可以加载多个chain。
3. Endpoint有两种类型，对应单轮调用Flow API和对话调用ChatFlow API。

#### 服务托管



LangChain执行chain

FastAPI为chain提供HTTP hosting

RayServe+RayCluster为预测服务提供资源调度。每个App封装一个chain，通过一个PromptFlowService的actor提供对外访问，chain之间想不干扰。

Kubernetes的Service/Ingress提供流量入口

参考资料

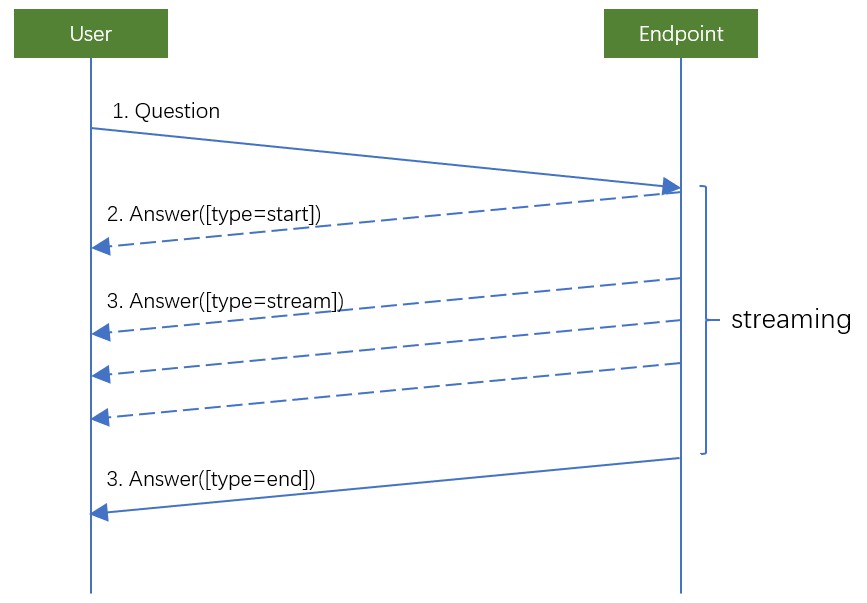
* [Langchain Deployment](https://python.langchain.com/docs/guides/deployments/)
* [Jina Langchain-serve](https://github.com/jina-ai/langchain-serve)
* [BentoML](https://github.com/bentoml/BentoML/tree/main)

#### ChatAPI设计

该API能力包括：

1. 支持多轮对话，覆盖LLM的Completion API要求
2. 支持Streaming，LLM推理是N字符/s，输出长达百个字符时，等待可能10s以上
3. 支持Human in Loop，允许用户干预/修改prompt
4. 支持结构化信息，类似Chat Card
5. 兼容一言/ChatGPT的Chat Completion API

以下是一轮用户对话的API交互，在建好WebSocket连接后，Endpoint以Streaming方式开始回复，先发一个type=start的消息，然后多轮返回answer的字符流，最后返回一个type=end的消息。



期望PromptFlowServer能兼容一言和OpenAI的ChatCompletion API，因此使用者可以用一言和OpenAI Python SDK访问服务。

|  |  |  |  |
| --- | --- | --- | --- |
| 场景 | 应用API设计 | 返回的content格式 | 执行过程 |
| 根据提问写作 | 无 | markdown格式 | 使用Chat Completion API |
| 文档搜索 | 无 | Role: Bot,  Content：markdown格式文本  Role: Card,  Content:<Card>  {  Kind: SearchDoc,  Name: SearchDoc,  Action: null,  Data: SearchResponse  } | 1. 用户提问 2. 查询向量库 3. 送给LLM写作 4. 构造Card 5. 返回 6. 前端渲染搜索文档 |
| 数据分析 | 无 | Role: Bot,  Content：markdown格式文本  Role: Card,  Content:<Card>  {  Kind: ReadTable,  Name: ReadTable,  Action: null,  Data: TableAnswer  } | 1. 用户提问 2. LLM生成SQL 3. 执行SQL 4. 返回 5. 前端渲染表格数据 |
| 数据图表 | 当图表需要注册到工作台时，转化成API调用的行为。  提供一个类似redash的图表API，HTTP接口包括3个：  CreateChart  Method：POST  URL：/chart  Body：  CreateChartRequest  获取chart  Method: GET  URL: /chart?name=<table\_answer.name>&filter=… | Role: Bot,  Content：markdown格式文本  Role: Card,  Content:<Card>  {  Kind: ReadTable,  Name: ReadTable,  Action: null,  Data: TableAnswer  } | 1. 用户提问 2. LLM生成SQL 3. 执行SQL 4. SQL解析成JSON query 5. 使用CreateChart API保存JSON query 6. 返回 7. 前端渲染表格数据 8. 前端在工具台界面提供/chart?name=<table\_answer.name>，支持刷新query |
| 发起API调用 | 复用已有API | Bot：markdown格式文本  Card：  {  Kind: InvokeAPI,  Name: ListAlarm,  Action: {  Method: GET,  URL: /alarms?filter=today  },  Data: null  } | 1. 用户提问 2. LLM意图识别 3. 构造Card 4. 返回 5. 前端调用API |
| 渲染API调用结果 | 类似Google的搜索API，HTTP接口是一个SearchDoc，返回命中的文档片段，请求如下：  Method：GET  URL：/doc/search  Query：?query=query  响应SearchResponse：  {  HitCount: 5,  Answers: []DocAnswer  } | Bot：markdown格式文本  Card：  {  Kind: ReadJSON,  Name: SearchDoc,  Action: {  Method: GET,  URL: /doc/search?query=query  },  Data: SearchResponse  } | 1. 用户提问 2. LLM意图识别 3. 后端调用API 4. 送给LLM写作 5. 构造Card 6. 返回 7. 前端渲染Data |
|  |  |  |  |

Card

* Name
* Kind
* Action
  + Method
  + URL
  + Form
  + Body
* Data
* Extra

SearchResponse

* HitCount
* []DocAnswer

DocAnswer

* Name
* URL
* Mentions
  + Mention
    - Location
    - Text
* Tags
* LastUpdateTime

CreateChartRequest

* Query

TableAnswer

* Name
* ChartType
* Data

文档搜索场景

参考资料

* [ChatGPT Chat Completions API](https://platform.openai.com/docs/guides/gpt/chat-completions-api)
* [Jina Langchain-serve HITL](https://github.com/jina-ai/langchain-serve/blob/main/examples/websockets/hitl/README.md)
* [Google Chat API Message](https://developers.google.com/chat/api/reference/rest/v1/spaces.messages#Message)
* [Microsoft Graph API ChatMessage](https://learn.microsoft.com/en-us/graph/api/resources/chatmessage?view=graph-rest-1.0)

#### ToolAPI

chain的API有两种类型，一种是单轮对话，可以用于各种agent场景，或者对话场景，API可以不至于文本输入输出，可以支持结构化输入输出。

参考资料

[AzureML PromptFlow Endpoint](https://learn.microsoft.com/en-us/azure/machine-learning/prompt-flow/how-to-deploy-for-real-time-inference?view=azureml-api-2)

### PromptFlowService

PromptFlowService是PromptFlow的控制面，提供业务上Prompt、PromptFlow、ChatHistory的核心概念的管理，以及辅助的主题域等信息管理。

目前文档管理等也放在PromptFlowService里，根据后期系统设计再考虑重构。

#### Prompt管理

功能：

* 提供Prompt的版本、分类和标签管理。
* 提供不同PromptFlow的激活、推荐Prompt查询

#### PromptFlow管理

功能：

* 提供langchain的python chain管理
* 提供langchain的json chain管理

PromptFlow与LangFlow的交互

#### ChatHistory管理

功能：

* 提供对话历史记录，相当于数据回流功能
* 对langchain提供Conversation Memory，涉及向量化

#### ChatHistorySample管理

功能：

* 针对ChatHistory的标注和人为修正记录

#### DocumentStore管理

功能：

* 提供DocumentStore级别的多租户管理
* 提供Document管理，一个DocumentStore有多个Document

### Windmill & LakeInsight相关服务

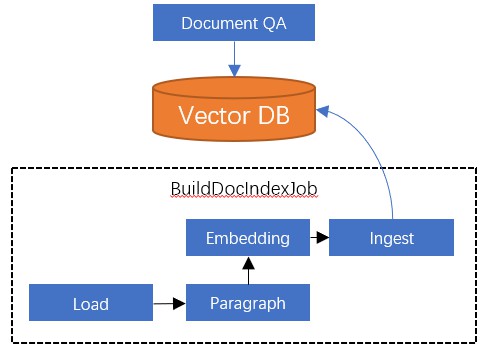
基于TrainService的BuildDocIndexJob

基于EndpointService来管理多租户的PromptFlowEndpoint

### DocToolkit

文档问答分为两个阶段

1. 离线构建基于embedding的知识库。分为文档读取，段落抽取，向量化，导入向量库四部分
2. 在线DocumentQA的chain执行向量查询。



相关模块

* DocumentQA chain，文档问答chain
* BuildDocIndexJob，文档向量化作业
* DocLoader，文档读取
* ParagraphParser，段落解析
* EmbeddingTransformer，文本向量化
* EmbeddingIngestor，向量导入到向量库

### BIToolkit

### APIToolkit

参考资料：

* [Gorilla](https://github.com/ShishirPatil/gorilla)