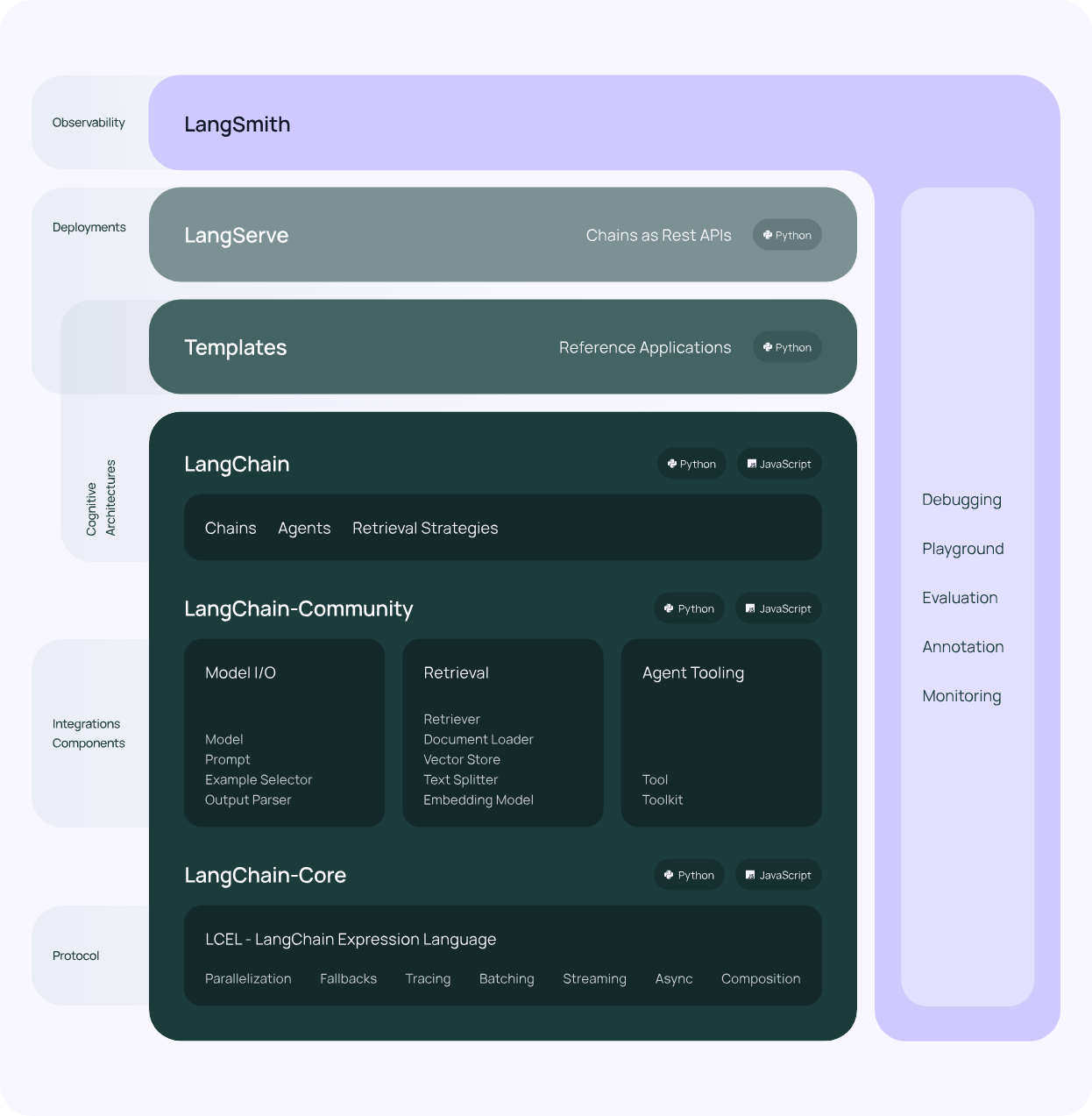
# LangChain 系列教程（一）：入门

## 介绍

**LangChain** 是一个用于开发由语言模型驱动的应用程序的框架。它使得应用程序能够：

* **具有上下文感知能力**：将语言模型连接到上下文来源（提示指令，少量的示例，需要回应的内容等）
* **具有推理能力**：依赖语言模型进行推理（根据提供的上下文如何回答，采取什么行动等）

这个框架由几个部分组成。

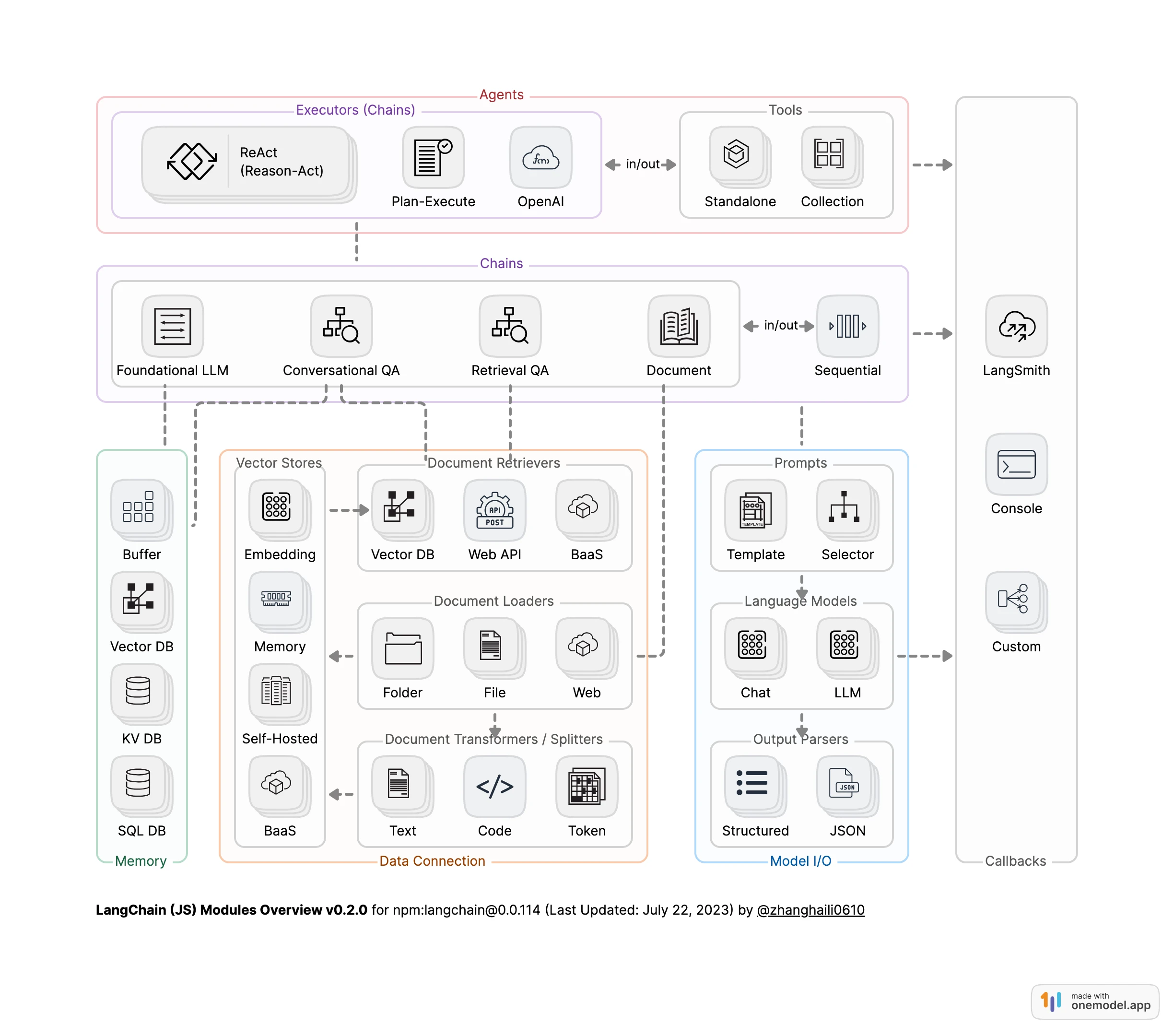
* **LangChain 库**：Python 和 JavaScript 库。包含了各种组件的接口和集成，一个基本的运行时，用于将这些组件组合成链和代理，以及现成的链和代理的实现。
* **LangChain 模板**：一系列易于部署的参考架构，用于各种任务。
* **LangServe**：一个用于将 LangChain 链部署为 REST API 的库。
* **LangSmith**：一个开发者平台，让你可以调试、测试、评估和监控基于任何 LLM 框架构建的链，并且与 LangChain 无缝集成。 

## 核心模块

LangChain由6个module组成，分别是Model IO、Retrieval、Chains、Memory、Agents和Callbacks。

* **Model I/O**：AI应用的核心部分，其中包括输入、Model和输出。
* **Retrieval**：“检索“——该功能与向量数据密切库相关，是在向量数据库中搜索与问题相关的文档内容。
* **Memory**：为对话形式的模型存储历史对话记录，在长对话过程中随时将这些历史对话记录重新加载，以保证对话的准确度。
* **Chains**：虽然通过Model IO、Retrieval和Memory这三大模块可以初步完成应用搭建，但是若想实现一个强大且复杂的应用，还是需要将各模块组合起来，这时就可以利用Chains将其连接起来，从而丰富功能。
* **Agents**：它可以通过用户的输入，理解用户的意图，返回一个特定的动作类型和参数，从而自主调用相关的工具来满足用户的需求，将应用更加智能化。
* **Callbacks**: 回调机制可以调用链路追踪，记录日志，帮助开发者更好的调试LLM模型。

这6个module具体的关系如下图所示：



## LangChain的六个模块介绍

该部分可以参考：https://mp.weixin.qq.com/s/C8pQGy-sv14MlYSQIOGPBA。写得比较详细。

## LangChain “HelloWorld”代码示例

### 配置环境

!conda create -n langchain-demo python==3.10 -y # conda环境搭建  
!pip install langchain langchain-community openai python-dotenv # 下载必要包

### 获取openai api密钥

访问 https://platform.openai.com/account/api-keys 获得官方api key。

访问 https://github.com/chatanywhere/GPT\_API\_free 获得免费版api key。

### 创建基础应用框架

# 导入openai api的base url和api key  
import os  
os.environ["OPENAI\_API\_BASE"] = "https://api.chatanywhere.tech"  
os.environ["OPENAI\_API\_KEY"] = "sk-xxx"  
  
from langchain.chat\_models import ChatOpenAI  
from langchain.chains import LLMChain  
from langchain.prompts import PromptTemplate  
  
# 初始化 ChatOpenAI 模型  
llm = ChatOpenAI(  
 model\_name="gpt-3.5-turbo", # 指定使用的模型名称  
 temperature=0.7, # 设置生成文本的随机性，范围为 0 到 2，值越高结果越随机  
)  
  
# 创建提示模板，定义模型的输入格式  
prompt = PromptTemplate(  
 input\_variables=["question"], # 定义模板中需要填充的变量  
 template="请回答下面的问题：{question}" # 定义提示的模板  
)  
  
# 创建语言模型链，将提示模板与模型结合  
chain = LLMChain(llm=llm, prompt=prompt)  
  
# 运行链，向模型提问  
response = chain.run("什么是人工智能？")  
  
# 输出模型的回答  
print(response)

人工智能是一种模拟人类智力的技术，通过计算机系统实现智能行为和决策能力。它包括机器学习、语音识别、图像识别、自然语言处理等技术，可以帮助计算机系统模拟人类的思维和行为，从而实现自动化、智能化的功能。

### 添加记忆功能

from langchain.memory import ConversationBufferMemory  
from langchain.chains import ConversationChain  
  
# 创建对话链，包含语言模型和对话记忆  
conversation = ConversationChain(  
 llm=llm, # 使用的语言模型  
 memory=ConversationBufferMemory(), # 对话记忆缓冲区  
 verbose=True # 输出详细的日志信息  
)  
  
# 进行对话  
print(conversation.predict(input="你好！"))  
print(conversation.predict(input="我们刚才说了什么？"))

​

[1m> Entering new ConversationChain chain...[0m  
Prompt after formatting:  
[32;1m[1;3mThe following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific details from its context. If the AI does not know the answer to a question, it truthfully says it does not know.  
  
Current conversation:  
  
Human: 你好！  
AI:[0m  
  
[1m> Finished chain.[0m  
Hello! 你好！How are you today?

​

[1m> Entering new ConversationChain chain...[0m  
Prompt after formatting:  
[32;1m[1;3mThe following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific details from its context. If the AI does not know the answer to a question, it truthfully says it does not know.  
  
Current conversation:  
Human: 你好！  
AI: Hello! 你好！How are you today?  
Human: 我们刚才说了什么？  
AI:[0m  
  
[1m> Finished chain.[0m  
We were just exchanging greetings and asking each other how we are. Is there something specific you would like to talk about?

### 集成外部工具

首先导入serpapi api key用于增加搜索功能，访问https://serpapi.com/并注册有免费api额度。

!pip install google-search-results # 下载相关包  
os.environ["SERPAPI\_API\_KEY"] = "xxx" # 导入 SerpAPI 密钥

!pip install numexpr # 导入numexpr提升数值计算的效率

from langchain.agents import load\_tools, initialize\_agent, AgentType  
  
# 初始化 ChatOpenAI 模型  
llm = ChatOpenAI(temperature=0.7)  
  
# 加载工具：serpapi 用于执行搜索查询；llm-math 用于处理数学计算  
tools = load\_tools(['serpapi', 'llm-math'], llm=llm)  
  
# 初始化agent  
agent = initialize\_agent(  
 tools,   
 llm,   
 agent=AgentType.ZERO\_SHOT\_REACT\_DESCRIPTION,  
 verbose=True  
)  
  
# 测试agent  
agent.run("今天的日期是多少？2025年有多少天？2的31次方减去1是多少？")

[1m> Entering new AgentExecutor chain...[0m  
[32;1m[1;3mWe need to find out the current date, the number of days in 2025, and the result of 2 to the power of 31 minus 1.  
Action: Calculator  
Action Input: Current date[0m  
Observation: [33;1m[1;3mAnswer: 2023[0m  
Thought:[32;1m[1;3mNow, let's calculate the number of days in 2025.  
Action: Calculator  
Action Input: Days in 2025[0m  
Observation: [33;1m[1;3mAnswer: 1826[0m  
Thought:[32;1m[1;3mNow, let's calculate 2 to the power of 31 minus 1.  
Action: Calculator  
Action Input: 2^31 - 1[0m  
Observation: [33;1m[1;3mAnswer: 2147483647[0m  
Thought:[32;1m[1;3mI now know the final answer.  
Final Answer: The current date is 2023. There are 1826 days in 2025. 2 to the power of 31 minus 1 is 2147483647.[0m  
  
[1m> Finished chain.[0m  
  
'The current date is 2023. There are 1826 days in 2025. 2 to the power of 31 minus 1 is 2147483647.'

### 添加文档问答能力

**example\_document.txt中内容如下：**

人与自然之间的关系自古以来便引人思考。广袤的森林、涓涓的溪流、绵延的群山，每一处景色都蕴藏着独特的气息。这些自然元素既是人类赖以生存的基础，也是滋养心灵的源泉。当我们在喧嚣的都市中感到疲惫时，往往会向往一片宁静的山林或空旷的海岸，让身心得到休憩。自然之美，不仅赋予我们视觉上的享受，更是一种精神上的慰藉。

四季的变换也凸显了人与自然之间的微妙联系。春天，万物复苏，柔嫩的花朵和新生的草木给大地披上一层绿意；夏天，蝉鸣不止，林荫小道上浓阴重重，鸟儿在枝头歌唱；秋天，金黄的落叶铺满大地，空气中弥漫着丰收的喜悦；冬天，银装素裹的世界让人感受到万籁俱寂的纯净。人们透过四季感受时间的流逝，也在与大自然的对话中体味生命的轮回。

现代社会的发展为我们带来了众多便利，却也无形中拉大了人与自然的距离。工业化、城市化的加剧，使得一些人难以直观地感受到自然的呼吸。一部分人对高楼林立的城市早已习以为常，却忘记了早晨花丛中晨露的清凉，忽略了夜晚繁星点点的浪漫。曾经轻而易举就能看见的自然美景，如今却成了假日里才得以偶尔“相遇”的稀缺风光。

然而，在忙碌的生活中，依然有人试图寻回与自然的紧密关联。他们在阳台上栽种花草，在休息日去公园和郊外漫步。他们观赏晚霞，将自己置身于微风拂面的山野间，用脚步去丈量每一寸土地。当双脚踩在泥土里，就能真切地感受到自然给予的力量——那种与土地同呼吸的踏实感，让人生出由衷的敬畏和感恩。

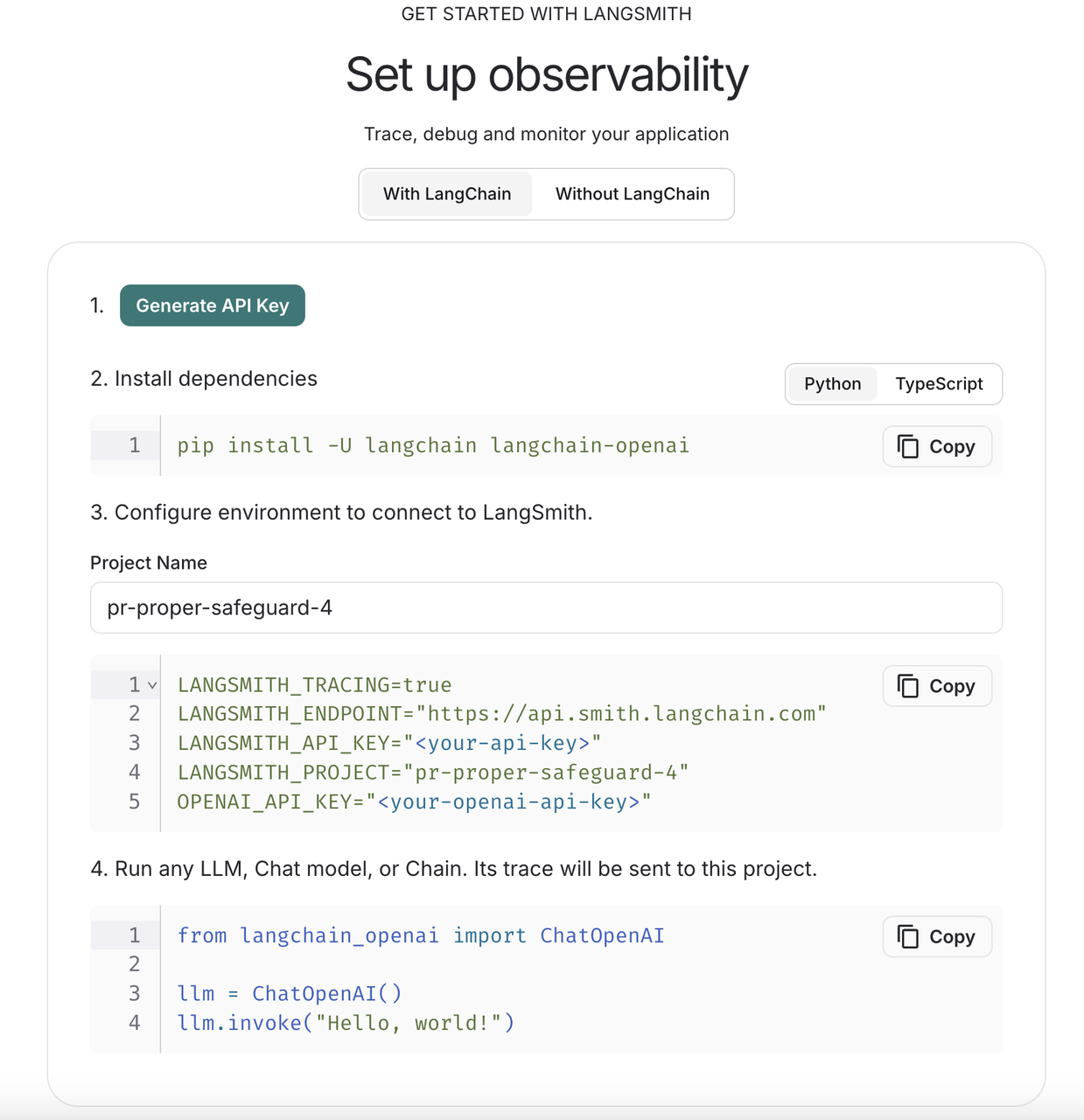
人与自然之间的情感交融，是每个人心中一处难以割舍的精神家园。自然带给我们壮阔与美丽，也孕育了我们对于世界的无限想象。我们应当以更加谦逊、更加真诚的态度，去探索和守护这一方天地。让人与自然相依相生，不仅是生态平衡的要求，也是我们心灵栖息的必经之路。愿每个人都能在自然的怀抱中，找到那份深沉而长久的宁静。

from langchain.embeddings import OpenAIEmbeddings # 或其他可用的 embedding 模型  
from langchain.document\_loaders import TextLoader  
from langchain.indexes import VectorstoreIndexCreator  
  
# 加载文档  
loader = TextLoader('example\_document.txt')  
  
# 指定 embedding 模型  
embedding = OpenAIEmbeddings()  
  
# 创建索引  
index = VectorstoreIndexCreator(embedding=embedding).from\_loaders([loader])  
  
# 查询  
response = index.query("文档中说了什么？", llm=llm)  
print(response)

/opt/anaconda3/envs/zhangjunhao/lib/python3.10/site-packages/langchain/indexes/vectorstore.py:171: UserWarning: Using InMemoryVectorStore as the default vectorstore.This memory store won't persist data. You should explicitlyspecify a vectorstore when using VectorstoreIndexCreator  
 warnings.warn(  
  
  
文档中讨论了人与自然之间的关系，描述了自然景观对人类的重要性，以及四季变换中展现的微妙联系。同时，文档也提到了现代社会发展对人与自然距离的拉大，但也强调了一些人仍然努力寻回与自然的紧密关联的重要性。整体上，文档强调了人与自然之间的情感交融以及人们应当以谦逊和真诚的态度去探索和守护自然的重要性。

### LangSmith监控

访问 https://smith.langchain.com/并注册，按照官网指示进行操作



# 导入配置  
import os  
os.environ["LANGSMITH\_TRACING"] = "true"  
os.environ["LANGSMITH\_ENDPOINT"] = "https://api.smith.langchain.com"  
os.environ["LANGSMITH\_API\_KEY"] = "xxx"  
os.environ["LANGSMITH\_PROJECT"] = "pr-puzzled-safeguard-28"  
os.environ["OPENAI\_API\_KEY"] = "sk-xxx"  
  
# 加上langsmith观测  
from langchain.chat\_models import ChatOpenAI  
from langchain.chains import LLMChain  
from langchain.prompts import PromptTemplate  
llm = ChatOpenAI(model\_name="gpt-3.5-turbo", temperature=0.7)  
prompt = PromptTemplate(input\_variables=["question"], template="请回答下面的问题：{question}")  
chain = LLMChain(llm=llm, prompt=prompt)  
response = chain.run("什么是人工智能？")  
print(response)

人工智能是一种模拟人类智能行为的技术，通过计算机系统和算法来模拟和实现人类的思维、学习、推理、感知和决策能力，从而让机器能够像人类一样处理复杂的任务和问题。人工智能包括机器学习、深度学习、自然语言处理、计算机视觉等技术领域，已经在各个领域得到广泛应用。

结果如下图所示

