LangServe: Serve LangChain

LangServe Intro and Walkthrough

Press Space for next page →

《AGI 应用实践》系列教程

AGI OVERVIEW

- 101：AGI 提示工程指北
- 102：OpenAI API 浅出

LANGCHAIN IN ACTION

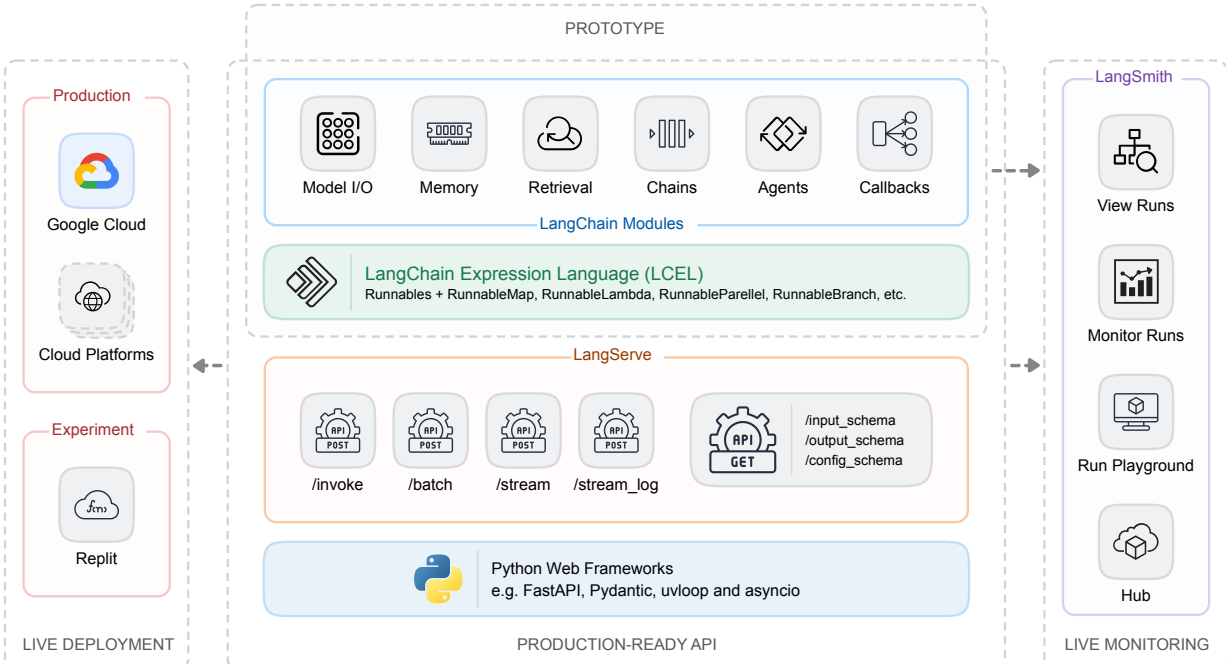
- 201：LangChain 功能模块解析（上篇）
- 202：LangChain 功能模块解析（中篇）
- 203：LangChain 功能模块解析（下篇）

- 301：LangChain 实战 - Chat LangChain 应用解析
- 302：LangServe - 部署 LangChain 最佳方式

教程大纲

- LangServe 之诞生
- 使用 LangServe 构建生产可用的 Web API
- LangServe Replit 模板
- LangServe 特性概览
- 调用 LangServe 端点接口的多种方式
- LCEL 对于 LangServe 的重要支撑

Production-Oriented LangChain APP Architecture



LangChain Universe v0.1.0
(Updated: Oct 15, 2023) by @zhanghaili0610

LangChain: https://github.com/langchain-ai
LangServe: https://github.com/langchain-ai/langserve
LangSmith: https://smith.langchain.com

您当前的浏览器不支持 HTML5 播放器

请更换浏览器再试试哦~

LangServe 之诞生

LangChain
@LangChainAI · **Follow**

🪁Introduction LangServe

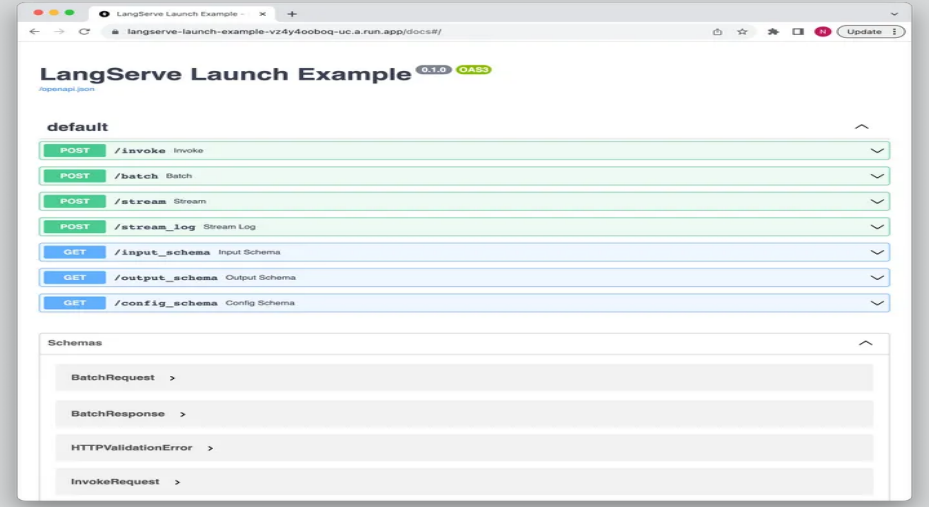
The best way to deploy your LangChains

📤Input/Output schema
📄/docs endpoint
🔤invoke/batch/stream endpoints
🎏/stream_log endpoint for streaming intermediate steps
🛠️LangSmith Integration

Used to power ChatLangChain and WebLangChain

Blog post and 🧵

LANGCHAIN BLOG



Introducing LangServe, the best way to deploy your LangChains

BY LANGCHAIN 5 MIN READ OCT 12, 2023

使用 LangServe 构建生产可用的 Web API

Building Production-Ready Web APIs with LangServe

`my_package/chain.py`:

```python
"""A conversational retrieval chain."""

from langchain.chains import ConversationalRetrievalChain
from langchain.chat_models import ChatOpenAI
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import FAISS

vectorstore = FAISS.from_texts(
    ["cats like fish", "dogs like sticks"],
    embedding=OpenAIEmbeddings()
)
retriever = vectorstore.as_retriever()

model = ChatOpenAI()

chain = ConversationalRetrievalChain.from_llm(model, retriever)
```

任意构建一个 Chain（完全可以是基于 LCEL 构建）

Build a Chain as you like (especially via LCEL)

`my_package/server.py`:

```python
#!/usr/bin/env python
"""A server for the chain above."""

from fastapi import FastAPI
from langserve import add_routes

from my_package.chain import chain

app = FastAPI(title="Retrieval App")

add_routes(app, chain)

if __name__ == "__main__":
    import uvicorn

    uvicorn.run(app, host="localhost", port=8000)
```

通过 `add_routes` 方法把 Chain 注册到 Web 服务

Register chain to web server via `add_routes`

LangServe Replit 模板

LangServe Replit Template

该模板展示了如何使用 LangServe 将 LangChain Expression Language **Runnable 部署为一组 HTTP 端点**到 Replit 上，并支持流和批处理

This template shows how to deploy a LangChain Expression Language Runnable as a set of HTTP endpoints with stream and batch support using LangServe onto Replit.

NOTICE

在运行示例前，需要通过左下角的 `Tools > Secrets` 来设置 `OPENAI_API_KEY`

Need to set an `OPENAI_API_KEY` environment variable by going under `Tools > Secrets` in the bottom left corner

LangServe 特性概览

LangServe Endpoints and Features

📚 `//docs` 通过 Swagger UI 展示和调试 API

`//docs` endpoint serves API docs with Swagger UI

📞 4 个“写”端点用于调用 Chain：`/invoke`，`/batch`，`/stream`，`/stream_log`

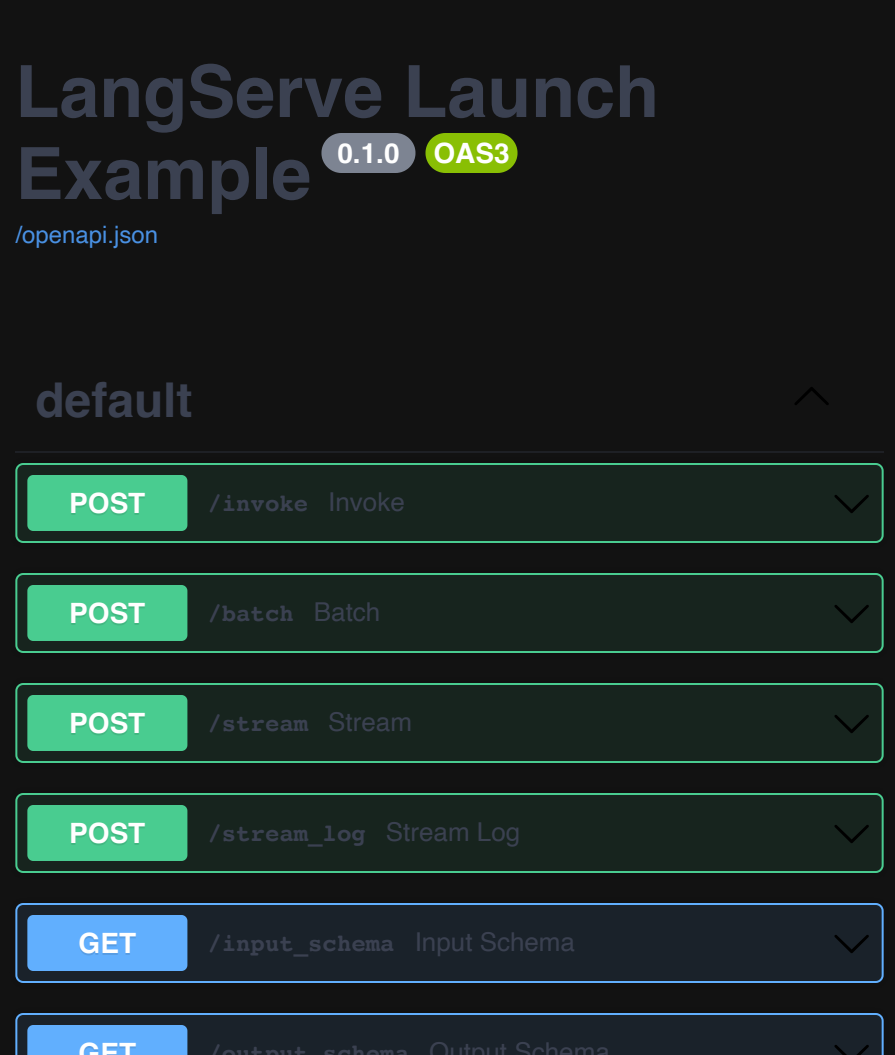
4 endpoints to call your chain in various approaches

🔍 2 个“读”端点用于获取 Chain 的输入输出结构：`/input_schema`，`/output_schema`

2 endpoints to retrieve input and output schemas (Pydantic models) auto-generated from the structure of the chain

📂 支持在同一服务的不同路径下托管多个 Chain

Support for hosting multiple chains in the same server under separate paths, e.g. `/chat/invoke`，`/say/invoke`



调用 LangServe 端点接口的多种方式

Calling hosted chain from various clients

```python
from langserve import RemoteRunnable

pirate_chain = RemoteRunnable("https://your_url.repl.co/chat/")

pirate_chain.invoke({"question": "how are you?"})
await pirate_chain.ainvoke({"question": "how are you?"})
```

```typescript
import { RemoteRunnable } from "langchain/runnables/remote"; // Introduced in LangChain.js 0.0.166+

const pirateChain = new RemoteRunnable({ url: `https://your_url.repl.co/chat/` });
const result = await pirateChain.invoke({ "question": "what did i just say my name was?" });
```

```bash
curl --location --request POST 'https://your_url.repl.co/chat/invoke' \
--header 'Content-Type: application/json' \
--data-raw '{
    "input": { "question": "what did i just say my name was?" }
}'
```

LCEL 对于 LangServe 的重要支撑

The journey to build LangServe really started when LCEL and the Runnable protocol launched

STREAMING CAPACITY

🌊 一流的流式传输支持：使用 LCEL 时，您可以获得最佳的首次 Token 时间；正在尝试支持 流式 JSON 解析器

First-class support for streaming: build your chains with LCEL you get the best possible time-to-first-token, streaming JSON parser is WIP

🔍 流式输出中间结果：添加了对 流式输出中间结果 的支持，并且在每个 LangServe 服务上都可用

Accessing intermediate results: added support for streaming intermediate results, and it's available on every LangServe server

CONTROL FLOW

⚡ 优化的并行执行：只要 LCEL 链具有 可以并行执行 的步骤，就会在同步和异步接口中自动并行执行此操作

Optimized parallel execution: whenever LCEL chains have steps that can be executed in parallel, will do it in both sync and async interfaces

🔁 支持重试和回退：为 LCEL 链的增加了 重试和回退 的支持；目前正在努力添加对重试/回退的流支持

Retries and fallbacks: added support for any part of your LCEL chain; currently working on adding streaming support for retries/fallback

参考资料

本教程在制作过程中参考和引用了以下资料（排名不分先后）的内容，特此鸣谢！

视频资料

- Short Courses | Learn Generative AI from DeepLearning.AI

图文资料

- Core Concepts | 🦜🔗 LangChain
  - JS/TS Docs, Python Docs, LangSmith Docs
  - LangChain Blog, Release Notes
- 入门：Prompts（提示词）｜通往 AGI 之路

代码资料

- openai/openai-cookbook: Examples and guides for using the OpenAI API
- datawhalechina/prompt-engineering-for-developers: 吴恩达大模型系列课程中文版
- slidevjs/slidev: Presentation Slides for Developers

感谢聆听 ❤️

webup | serviceup