Ch01 LangChain 시작하기

`LangChain` :언어 모델을 활용해 다양한 애플리케이션을 개발할 수 있는 프레임워크

이 프레임워크를 통해 언어 모델은 다음과 같은 기능을 수행할 수 있게 됩니다.

문맥을 인식하는 기능: LangChain은 언어 모델을 다양한 문맥 소스와 연결합니다.

추론하는 기능: 또한, 언어 모델은 주어진 문맥을 바탕으로 어떠한 답변을 제공하거나, 어떤 조치를 취해야 할지를 스스로 추론할 수 있습니다.

LangChain 을 활용하면 이전에 언급한 기능을 바탕으로 **검색 증강 생성(RAG) 어플리케이션 제작**, **구조화된 데이터 분석**, **챗봇** 등을 만들 수 있습니다.

설치

권장하는 파이썬 버전은 `3.11` 버전입니다.

pip 를 이용한 설치

```
pip install -r https://raw.githubusercontent.com/teddylee777/langchain-kr/main/requirements.txt
```

최소한의 기능만 설치하기 위한 **mini** 버전 (일부 패키지만 설치하는 경우)

```
pip install -r https://raw.githubusercontent.com/teddylee777/langchain-kr/main/requirements-mini.txt
```

구성

이 프레임워크는 여러 부분으로 구성되어 있습니다.

- **LangChain 라이브러리**: Python 및 *JavaScript 라이브러리*.
  다양한 컴포넌트의 인터페이스와 통합, 이러한 컴포넌트를 /체인과 에이전트로/ 결합하는 기본 런타임, 그리고 즉시 사용 가능한 체인과 에이전트의 구현을 포함합니다.

  > **컴포넌트(Components)를 결합:** AI 애플리케이션의 기본 기능 단위들을 조합한다는 뜻
  >
  > 컴포넌트를 연결하는 두가지 방식:

> - **체인:** 미리 정해진 순서대로 일련의 컴포넌트(단계)를 실행합니다.
> - **에이전트:** LLM이 스스로 다음 행동을 결정하며 도구(Tool)를 사용해 목표를 달성하는, 더 동적인 실행 흐름입니다.
>
> **기본 런타임(Basic Runtime):** 이러한 결합된 구조(체인 또는 에이전트)를 실제로 메모리에 로드하고, 입력 데이터를 처리하며, 정의된 로직에 따라 실행 결과를 만들어내는 **실행 환경** 또는 **엔진**

- **LangChain 템플릿**: 다양한 작업을 위한 쉽게 배포할 수 있는 참조 아키텍처 모음입니다.

> **참조 아키텍처(Reference Architecture)**는 특정 기술 도메인이나 산업 분야에서 **가장 바람직하고 검증된 구조와 구성**을 제시하는 청사진 또는 템플릿
>
> 1. 검증된 모범 사례 (Best Practices)
> 수많은 프로젝트를 통해 **성공적으로 구현되고 운영 효율성이 입증된** 구조와 패턴을 문서화한 것
>
> 2. 표준화 및 공통 언어 (Standardization & Common Vocabulary)
> 시스템을 구성하는 요소(컴포넌트), 이들이 상호작용하는 방식, 사용해야 할 기술 등에 대한 **표준을 제시**
>
> 3. 청사진 역할 (Blueprint or Template)
> **'어떻게' 시스템을 구성해야 하는지**에 대한 가이드라인

- **LangServe**: LangChain 체인을 REST API로 배포하기 위한 라이브러리입니다.
- **LangSmith**: 어떤 LLM 프레임워크에도 구축된 체인을 디버그, 테스트, 평가, 모니터링할 수 있게 해주며 LangChain과 원활하게 통합되는 개발자 플랫폼입니다.
- **LangGraph**: LLM을 사용한 상태유지가 가능한 다중 액터 애플리케이션을 구축하기 위한 라이브러리로, LangChain 위에 구축되었으며 LangChain과 함께 사용하도록 설계되었습니다. 여러 계산 단계에서 다중 체인(또는 액터)을 순환 방식으로 조정할 수 있는 능력을 LangChain 표현 언어에 추가합니다.

> **표현 언어= LCEL**(LangChain Expression Language)
> LangChain이 제공하는 선언적(Declarative) 방식의 **인터페이스**
> 쉽게 말해, **컴포넌트**들을 연결해서 **체인**을 만드는 데 사용하는 **파이프라인 문법**
>
> LCEL의 가장 특징적인 요소는 바로 **파이프(|) 연산자**. 이 기호는 한 컴포넌트의 출력을 다음 컴포넌트의 입력으로 전달하는 역할을 함.
>
> ```
> # 일반적인 LLM 체인 구성
> chain = prompt | model | output_parser
> ```

> - `prompt`의 출력 → `model`의 입력
> - `model`의 출력 → `output_parser`의 입력

개발 용이성✨

컴포넌트의 조립 및 통합 🔧

- LangChain은 언어 모델과의 작업을 위한 조립 가능한 도구 및 통합을 제공
- 컴포넌트는 모듈식으로 설계되어, 사용하기 쉽습니다. 이는 개발자가 LangChain 프레임워크를 자유롭게 활용할 수 있게 함

즉시 사용 가능한 체인 🚀

- 고수준 작업을 수행하기 위한 컴포넌트의 내장 조합을 제공
- 이러한 체인은 개발 과정을 간소화하고 속도를 높임

주요 모듈 📌

모델 I/O 📃

- 프롬프트 관리, 최적화 및 LLM과의 일반적인 인터페이스와 작업을 위한 유틸리티를 포함합니다.

검색 📚

- '데이터 강화 생성'에 초점을 맞춘 이 모듈은 생성 단계에서 필요한 데이터를 외부 데이터 소스에서 가져오는 작업을 담당합니다.

에이전트 🤖

- 언어 모델이 어떤 조치를 취할지 결정하고, 해당 조치를 실행하며, 관찰하고, 필요한 경우 반복하는 과정을 포함합니다.

03. LangSmith 추적 설정

LangSmith는 **LLM 애플리케이션 개발, 모니터링 및 테스트** 를 위한 플랫폼입니다.

LangChain 사용 여부와 관계없이 동급 최고의 추적 기능을 제공합니다.

추적은 다음과 같은 문제를 추적하는 데 도움이 될 수 있습니다.

- 예상치 못한 최종 결과
- 에이전트가 루핑되는 이유
- 체인이 예상보다 느린 이유

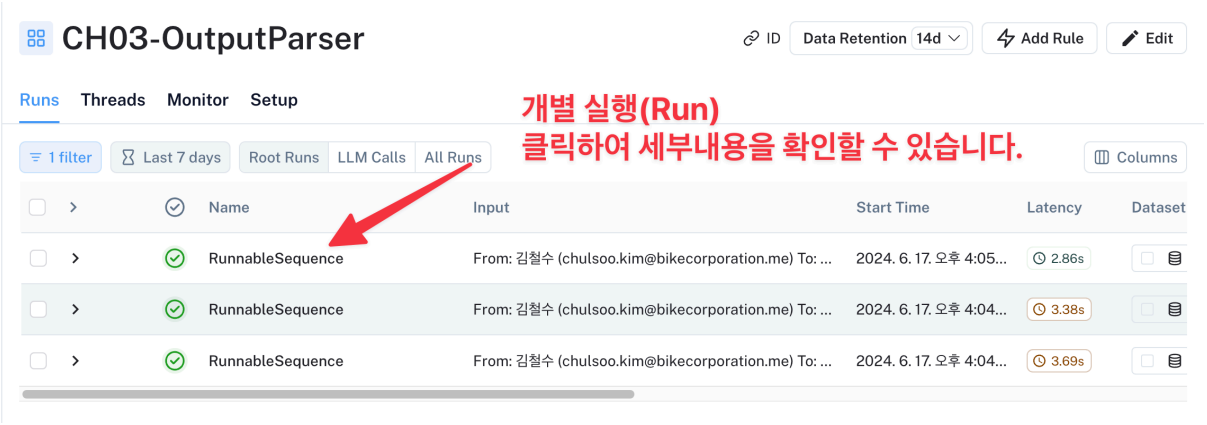
- 에이전트가 각 단계에서 사용한 토큰 수

프로젝트 단위 추적

프로젝트 단위로 실행 카운트, Error 발생률, 토큰 사용량, 과금 정보등을 확인할 수 있습니다.

Name	Feedback (7D)	Run Count (7D)	Error Rate (7D)	% Streaming (7D)	Total Tokens (7D)	Total Cost (7D)	P50 Latency (7D)
CH04-Models		22	0%	0%	6,501	$0.0091835	0.00s
CH01-Basic		48	0%	9%	189,876	$0.9631875	1.02s
llama3-agent		33	12%	94%	94,709		18.39s
CH03-OutputParser		3	0%	67%	3,332	$0.002268	3.38s
CH02-Prompt		29	3%	24%	17,389	$0.26302	3.30s

프로젝트를 클릭하면 실행된 모든 Run 이 나타납니다.



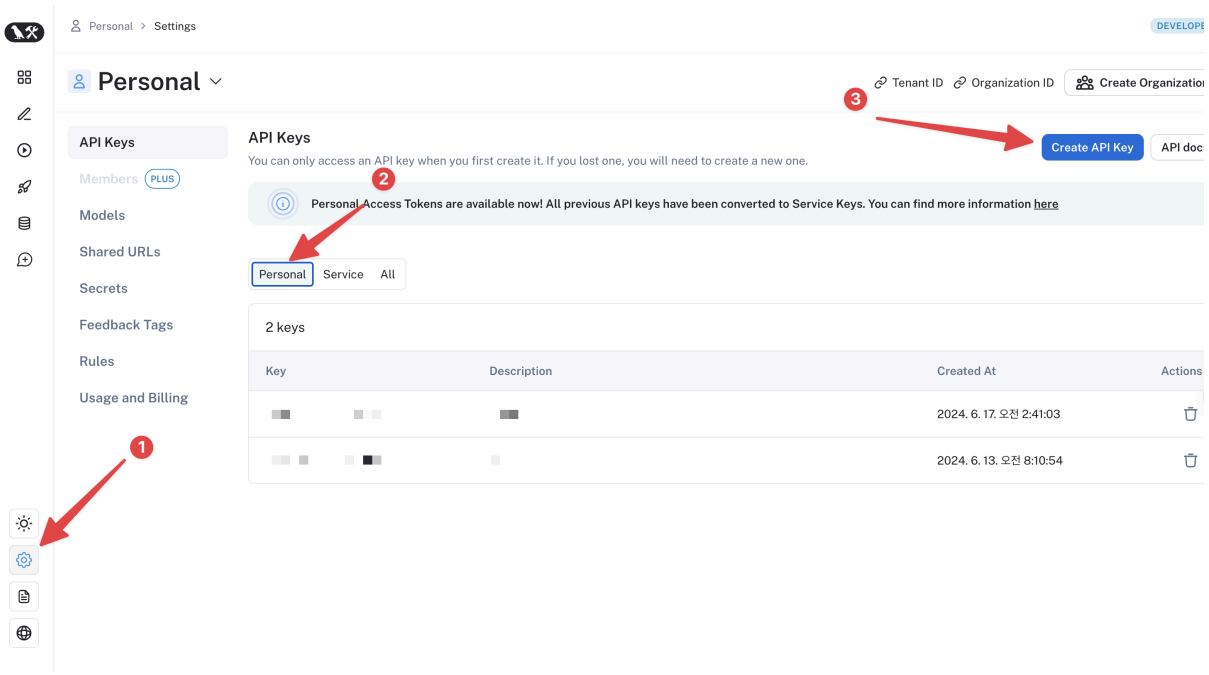
1개의 실행에 대한 세부 단계별 추적

1개의 실행을 한 뒤 retrieve 된 문서의 검색 결과 뿐만 아니라, GPT 의 입출력 내용에 대해서 자세하게 기록합니다. 따라서, 문서의 검색된 내용을 확인 후 검색 알고리즘을 변경해야할지 혹은 프롬프트를 변경해야할지 판단하는데 도움이 됩니다.

뿐만 아니라, 상단에는 1개의 실행(Run) 이 걸린 시간(약 30초)와 사용된 토큰(5,104) 등이 표기가 되고, 토큰에 마우스 호버를 하게 되면 청구 금액까지 표기해 줍니다

LangSmith 추적 사용하기

LangSmith API Key 발급

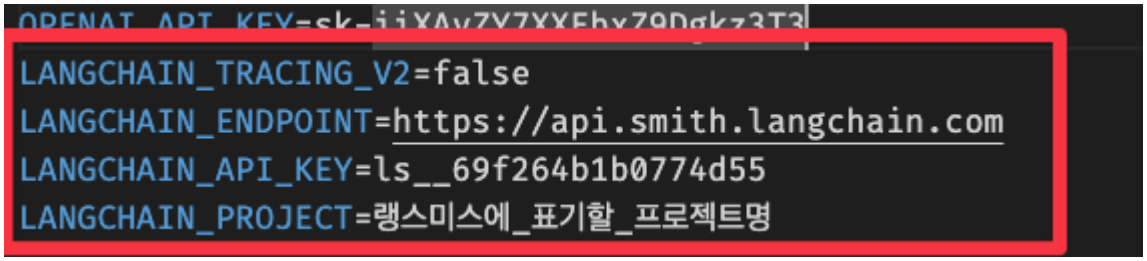


(주의!) 생성한 키를 유출하지 않도록 안전한 곳에 복사해 두세요.

.env 에 LangSmith 키 설정

먼저, `.env` 파일에 LangSmith 에서 발급받은 키와 프로젝트 정보를 입력합니다.

- `LANGCHAIN_TRACING_V2` : **"true"** 로 설정하면 추적을 시작합니다.
- `LANGCHAIN_ENDPOINT` : `https://api.smith.langchain.com` 변경하지 않습니다.
- `LANGCHAIN_API_KEY` : 이전 단계에서 **발급받은 키** 를 입력합니다.
- `LANGCHAIN_PROJECT` : **프로젝트 명** 을 기입하면 해당 프로젝트 그룹으로 모든 실행(Run) 이 추적됩니다.



05. LangChain Expression Language(LCEL)

06. LCEL 인터페이스

07. Runnable