**LangSmith Documentation**

**LangSmith Introduction** :

LangSmith is a comprehensive platform designed for building, monitoring, debugging, and evaluating production-grade LLM (Large Language Model) applications. It provides detailed insights and tools to optimize the performance and reliability of LLM applications.

* Building Production-Grade Applications: LangSmith offers robust SDKs for Python and TypeScript, facilitating the integration and utilization of LLMs.
* Monitoring and Debugging: Track and debug applications with real-time insights and detailed metrics.
* Performance Evaluation: Evaluate LLM performance using various metrics, such as average latency, total tokens used, and error rates.

**Key Features :**

1. **Building Production-Grade LLM Applications**

* Development Tools: LangSmith provides robust SDKs for both Python and TypeScript, facilitating the integration and utilization of LLMs in various applications.
* API Integration: Easily integrate with popular APIs, including OpenAI, to enhance your application's capabilities.

1. **Monitoring and Debugging**

* Real-Time Monitoring: Track the performance of your LLM applications in real-time. This includes keeping an eye on critical metrics such as average latency and the total number of tokens used by each query or question.
* Detailed Debugging: Debug your applications with detailed insights provided by LangSmith. Identify and fix issues quickly to ensure smooth operation.

1. **Performance Evaluation**

* Comprehensive Evaluations: Evaluate the performance of your LLM models using built-in and custom evaluators. LangSmith helps you assess various parameters to understand how well your model is performing.
* Metrics and Analytics: Access detailed analytics that provide insights into the average latency, total tokens used, and other performance metrics.
* Thread Detection: Determine if there are any threading issues in your model, ensuring that your application can handle concurrent requests efficiently.

1. **Example Metrics**

* Average Latency: Measure the time taken to process each query, helping you identify performance bottlenecks.
* Total Tokens Used: Track the number of tokens used in each interaction to optimize cost and performance.
* Thread Presence: Detect if threading issues are present, which can impact the performance and reliability of your model.

LangSmith equips you with the tools and insights necessary to build, monitor, and evaluate high-performance LLM applications, ensuring that you can deploy with confidence and maintain optimal operation.

**Advantages :**

1. **Comprehensive Monitoring and Debugging:** Real-time insights and robust debugging tools.
2. **Performance Evaluation:** In-depth analysis with metrics like average latency and total tokens used.
3. **Ease of Integration:** SDKs for Python and TypeScript, compatible with popular APIs like OpenAI.
4. **Enhanced Productivity:** Automated tracing and user-friendly interface.
5. **Community and Support:** Active community, support channels, and extensive documentation.

**Disadvantages :**

1. **Learning Curve:** Can be complex for beginners.
2. **Dependency on External APIs:** Relies on third-party services which may affect performance and cost.
3. **Cost:** Potentially high investment for extensive use.
4. **Limited Offline Capabilities:** Requires an internet connection.
5. **Project Migration:** No easy migration between organizations.
6. **Data Sensitivity:** Additional steps needed to handle sensitive data securely.

**Langsmith Integration with LLM:**

1. **Creating an API Key :**
2. Navigate to the Settings page on the LangSmith platform.
3. Click on Create API Key to generate a new key.
4. Copy and securely store your API key.
5. **Environment Setup :**

.env : Add your Langsmith ,LLM api key in .env file

* #gemini api
* GOOGLE\_API\_KEY = '<LLM API KEY>'
* # LangSmith
* LANGCHAIN\_API\_KEY = ‘<Langsmith API KEY>’
* # LangSmith Project Name
* LANGCHAIN\_PROJECT = "Gemini Chatbot"

1. **Project Setup :**

* First import OS (Operating System)
* Then call api from .env file
* And then make trace true
* import os
* load\_dotenv()
* # Load the Google Generative AI Embeddings
* os.getenv("GOOGLE\_API\_KEY")
* genai.configure(api\_key=os.getenv("GOOGLE\_API\_KEY"))
* #Langsmith api load
* os.environ["LANGCHAIN\_TRACING\_V2"] = "true"
* os.environ["LANGCHAIN\_API\_KEY"] = os.getenv("LANGCHAIN\_API\_KEY")

**Viewing Traces :**

To view your output traces, visit the tracing section on the LangSmith platform.