

LangSmith Setup Guide: Tracing Your LangGraph Applications

LangSmith is LangChain's developer platform that lets you trace, debug, evaluate, and monitor your LLM applications—including LangGraph workflows—automatically.

This guide walks you through:

You can go to this link for more details on how to generate API key:

<https://docs.langchain.com/langsmith/create-account-api-key>

1. Creating a LangSmith account.
2. Getting your ****API key****.
3. Configuring your .env file.
4. Enabling automatic tracing for your LangGraph app.

Step 1: Create a LangSmith Account

1. Go to <https://smith.langchain.com>
2. Click "Sign Up" (you can use Google, GitHub, or email)
3. Complete registration and verify your email
4. You'll be redirected to the LangSmith dashboard

The screenshot shows the LangSmith dashboard interface. On the left is a sidebar with navigation links: Home, Observability, Tracing Projects (2), Monitoring (0), Evaluation, Datasets & Experiments (0), Annotation Queues (0), Prompt Engineering, Prompts (0), and Playground. Below these are sections for LangGraph Platform: Deployments (0), Settings, and More. At the bottom of the sidebar is a Personal section. The main area is titled "Personal" with an ID placeholder. It features a "Get Started" section with three buttons: "Set up tracing", "Run an evaluation", and "Try out playground". Below this is an "Observability" section with tabs for "Tracing Projects" (2) and "Custom Dashboards" (0). A table lists two tracing projects: "extraction-op" and "default".

Name	Most Recent Run (7D)	Feedback (7D)	Run Count (7D)	Error Rate (7D)	P50 Latency (7D)
extraction-op	10/8/2025, 3:20:43 PM		27	7%	3.65s
default			0	0%	

Step 2: Get Your API Key

In the LangSmith dashboard, click your profile icon (top-right) → "Settings"

1. Go to the "API Keys" tab in settings
2. Click "Create API Key"
3. Give it a name (e.g., my-langgraph-dev)
4. Click "Create"
5. Copy your API Key ****(you won't see it again!)****

The screenshot shows the LangSmith API Keys management interface. On the left, there's a sidebar with navigation links: 'Back', 'Organizations' (Personal selected), 'Workspaces' (PLUS), 'Members and roles' (PLUS), 'API Keys' (selected), 'OAuth Providers', 'Models', 'Shared', 'Secrets', 'Feedback tags', 'Billing and usage', and 'Documentation'. The main area is titled 'API Keys' with a 'Personal' tab selected. It lists one API key: 'lsv2_pt_c ... 3495' (Expires Never), 'Auto created during onboarding', 'Created At 9/24/2025, 11...', 'Last Used At 10/8/2025, 3...', and an 'Actions' column with a trash can icon.

Step 3: Set Up Your .env File

In your project root create a ****.env**** file and paste your credentials

LangSmith Configuration

```
LANGCHAIN_API_KEY=your_actual_api_key_here  
LANGCHAIN_PROJECT=your_project_name_here  
LANGCHAIN_TRACING_V2=true  
LANGCHAIN_ENDPOINT=https://api.smith.langchain.com
```

Step 4: Automatic Tracing in LangGraph (Zero Code Changes!)

Once you set the environment variables above, LangChain (and LangGraph) will automatically trace every run—no code changes needed!

How It Works:

LangChain detects LANGCHAIN_TRACING_V2=true

Every LLM call, tool use, and graph node execution is sent to LangSmith

Your LangGraph state transitions, messages, and outputs appear in the dashboard

Now

Go to LangSmith Dashboard → "Traces"

The screenshot shows the LangSmith interface for a tracing project named "extraction-op". The main dashboard provides an overview of the project's performance with metrics like Run Count (27), Total Tokens (81,398 / \$0.44), Median Tokens (444), Error Rate (7%), and % Streaming (6%). Below this, a table lists individual runs with columns for Name, Input, Output, Error, and Start Time. The "Threads" section shows a trace for the "extract_structured" node, detailing its execution time (1.10s) and input file (Fail-6.pdf). The "Input" section shows the file being processed by various nodes like "extract_chunks" and "convert_to_markdown". A detailed view of the "extract_structured" node's input parameters is shown on the right, including Output Json Path (madhu.json), Output Md Path (madhu.md), Pdf Path (Fail-6.pdf), Markdown Chunks, and Text Chunks.

You'll see your full LangGraph execution with:

Each node (extract_chunks, convert_to_markdown, etc.)

Input/output state

LLM calls with prompts & responses

Timing and tokens used

For this you can run main.py or directly run the run_pipeline function in the main.graph.py with parameters and you can check the tracing for test purpose.

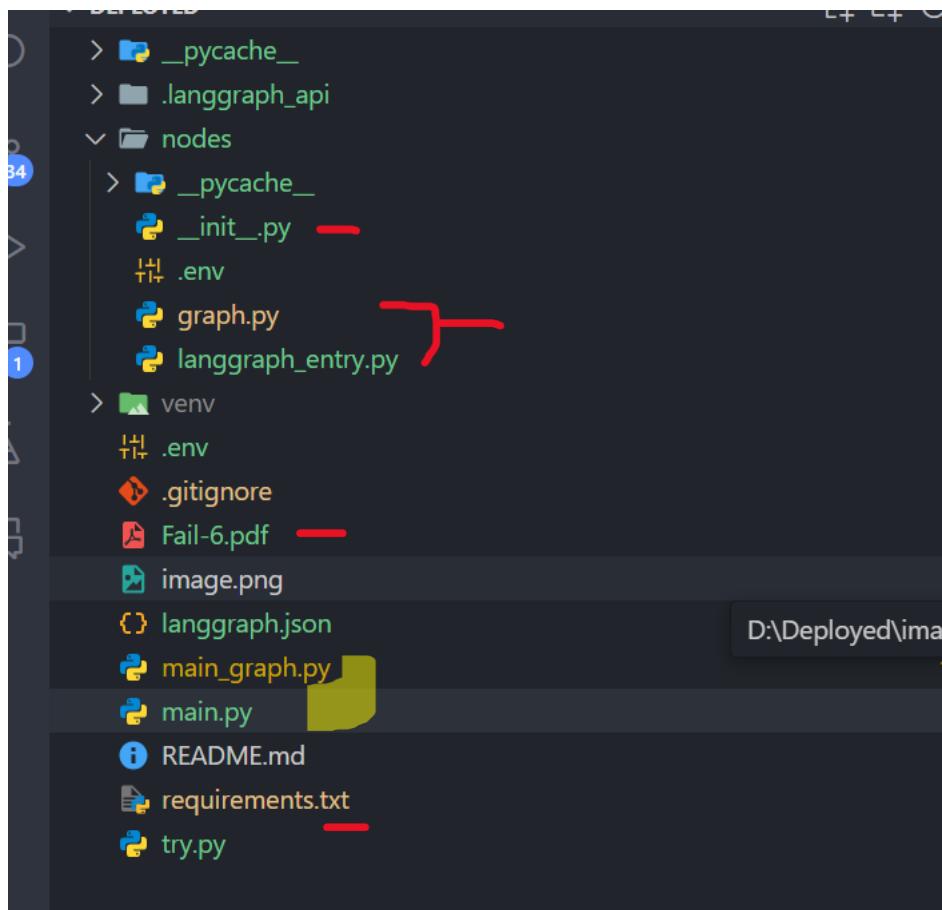
Now we would want to deploy the graph in LangGraph Studio in order to visualize it and make it accessible through an endpoint.

Deploying Your LangGraph App Locally with LangGraph Studio

****Organize your folder structure like this****

Deployed/

```
|   └── .env  
|  
|   └── langgraph.json ← LangGraph deployment config  
|  
|   └── nodes/  
|       |   └── langgraph_entry.py ← Entry point for the server  
|       |  
|       └── graph.py ← Contains your graph  
|  
└── run_server.py ← manual server script
```



langgraph.json -> tells the LangGraph server where to find the Graph.

```
{  
  "graphs": {  
    "default": {  
      "path": "nodes.langgraph_entry:graph"  
    },  
    "dependencies": [  
      "langgraph>=0.1.0",
```

```
"langchain-openai",
"python-dotenv",
"pdfplumber",
"pymupdf",
"pillow"
]
}
```

It is also having the dependencies that are required to run the graph.

Now we need to have ****langgraph-cli**** installed in the environment.

use langgraph dev to run from the root folder

```
langgraph dev
```

It will start the development server and take you to Langgraph studio hosted on
<http://127.0.0.1:2024>



****We can check on the Langgraph Studio to run it and it's like a playground where we can see our graphs and agents working ****

LangSmith

- Home
- Observability
- Tracing Projects (2)
- Monitoring (0)
- Evaluation
- Datasets & Experiments (0)
- Annotation Queues (0)
- Prompt Engineering
- Prompts (0)
- Playground
- LangGraph Platform
- Deployments (0)
- Settings
- More

Personal tomekotarup@gmail.com

Not seeing LangSmith runs?
It looks like your LangSmith API key is missing. Please make sure to add `LANGSMITH_API_KEY` to your local server's `.env` file.

← LangGraph Studio / default Graph Chat ⚙️ New Thread ⓘ Interact Trace Run exp

Memory Interrupts

```
graph TD; A[extract_chunks] --> B[convert_to_markdown]; B --> C[extract_structured_data]; C --> D[save_json]
```

New Thread

Submit your input to run the assistant

Input

Manage Assistants Submit

As you can see below we have the parameter or the data of state of graph

The screenshot shows the LangSmith interface. On the left is a sidebar with various sections like Home, Observability, Tracing Projects, Monitoring, Evaluation, Datasets & Experiments, Annotation Queues, Prompt Engineering, Prompts, Playground, LangGraph Platform, Deployments, Settings, More, and Personal. The main area has tabs for Graph, Chat, and a help icon. It shows a warning message: "Not seeing LangSmith runs? It looks like your LangSmith API key is missing. Please make sure to add LANGSMITH_API_KEY to your local server's .env file." Below this are sections for Memory and Interrupts. A large "Input" section contains fields for Pdf Path, Output Md Path, Output Json Path, Text Chunks, Markdown Chunks, and Final Metadata, all marked as Required. To the right is a "New Thread" button with a star icon and a placeholder "Submit your input to run the assistant". At the bottom is a "Submit" button.

****Now using try.py we will acces the above endpoint and send our files and run trace it in langsmith server****

When running locally give assstant_id as default

LangSmith