LangGraph

Langchain ecosystem: (Build app using langchain)

- Building standards / abstractions
- Chaining of calls, RAG implementation
- Prompt template
- Supports memory
- Decorative language
- Use Langchain workflows to build agentic platforms

LangGraph : Run at scale with Langgraph platform:

- Independent of langchain, can use langchain or others
- It is platform that focuses on stability, resiliency, and repeatability in words where you are solving problems that involve lot of interconnected processes, like agentic platform. So its an abstraction layer that allows you to organize your thinking around a workflow of different activities that could have feedback loops.
- Allows to organize, loops, humans, memory, organizing all these repeatedly things in stable way.
- Use this to design agent driven user experiences featuring things like human in loop,. Multi agent collaboration, conversation history, memory and time travel to traverse through the process and restore to a specific step in past or time.
- Deploy with tolerant scalability,

Langsmith: monitoring tooling

- Langgraph can integrate with langsmith for monitoring purpose.

LangGraph has:

- langGraph
- langGraph platform
- langGraph Studio

Article by Anthropic: https://www.anthropic.com/engineering/building-effective-agents

LangGraph terminology:

- Agent workflows are represented as **graphs**
- **State** represents the current snapshot of the application.
- **Nodes** are python functions that represent agent logic. They receive the current **State** as input, do something, and return an updated **State.**

- **Edges** are python functions that determine which **NODE** to execute next based on the **State**. They can be conditional or fixed.
- **Nodes** do the work.
- **Edges** choose what to do next.

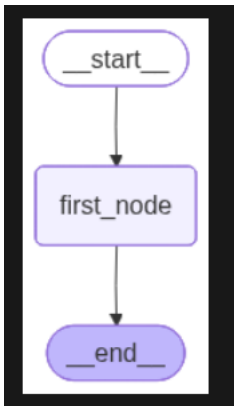
Five Steps to the first Graph:

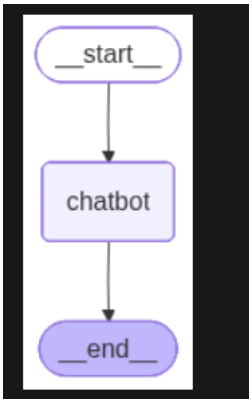
1. Define the state
2. Start the graph builder
3. Create a Node
4. Create Edges
5. Compile the graph

Lab 9: Intro to langsmith - 5 step setup without memory context:

Here lab on building graph using langgraph – with python func and with llm

But without memory or say context





More on the State:

- It is **immutable**
- For each field in your **State**, you can specify a function called a **reducer**
- When you return new **State**, LangGraph uses the **reducer** to combine this field with existing State.

Going Deeper after 5 steps:

- LangSmith
- Tools – out of the box
- Tools – custom
- Checkpointing

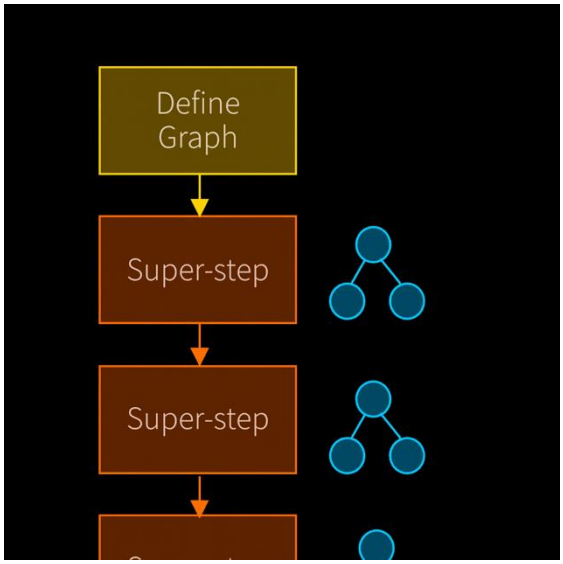
Super Step:

can be considered a single iteration over the graph nodes. Nodes that run in parallel are part of the same super-step, whiles nodes that run sequentially belong to separate super-steps.

Graph describes one super step; one interaction between agents and tools to achieve an outcome.

Every user interaction is a fresh graph.invoke(state) call.

The reducer handles updating state during a super step but not between steps.

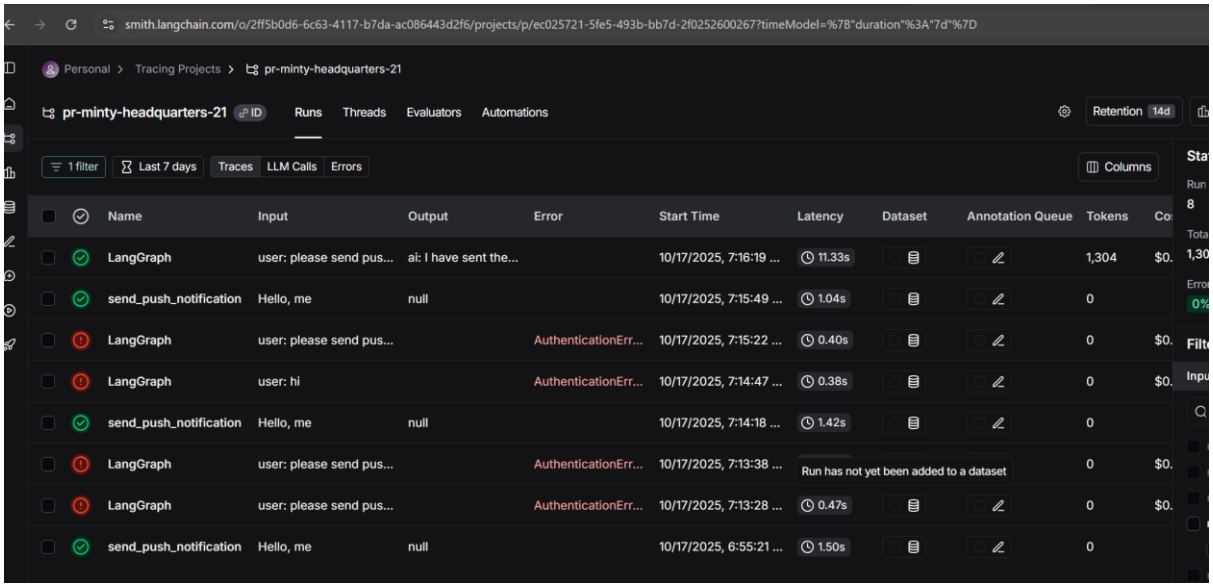

Define
Graph
Super-step
Super-step

Lab10: monitoring via langsmith, and GoogleSerperAPIWrapper with memory context – not persistent vs persistent.

LangSmith: https://smith.langchain.com/

Generate API key: https://smith.langchain.com/o/2ff5b0d6-6c63-4117-b7da-ac086443d2f6/?paginationModel=%7B%22pageIndex%22%3A0%2C%22pageSize%22%3A5%7D

pip install -U langgraph "langchain[openai]"



resource to read: https://www.anthropic.com/engineering/building-effective-agents

Serper dev tools :

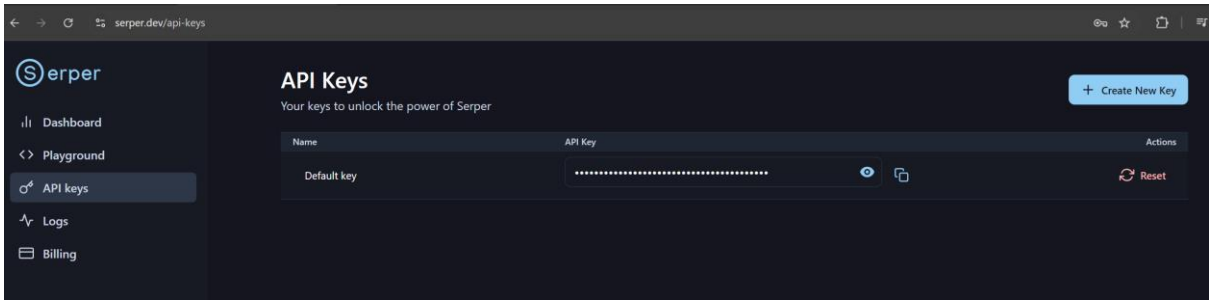
keep the key in the .env

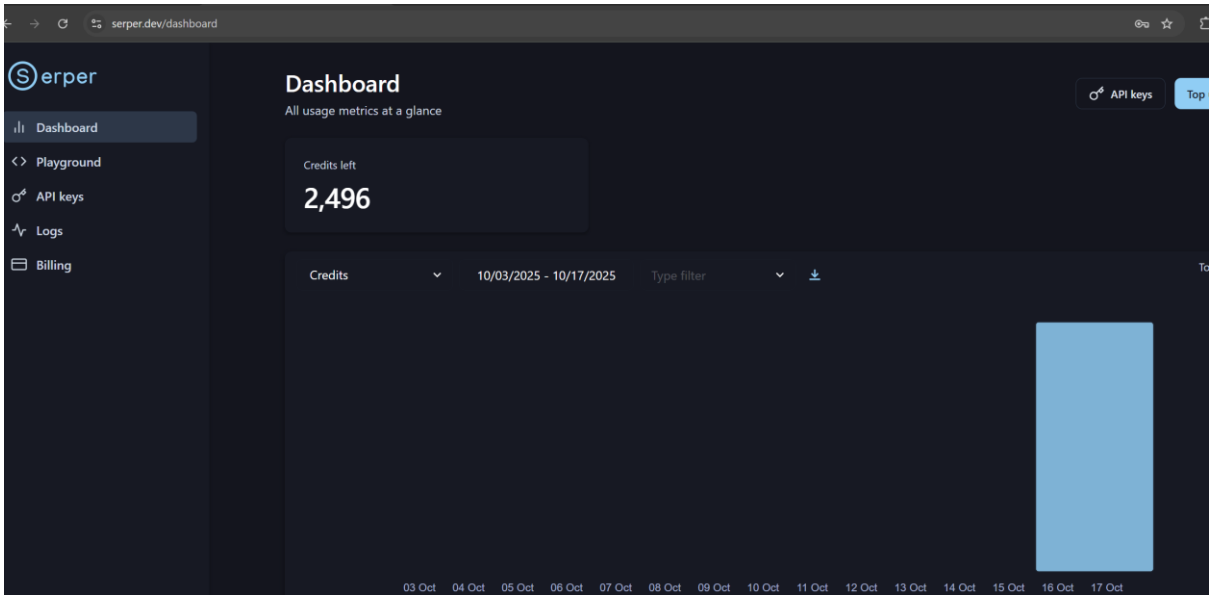
Serper.dev – cheapest google search api – 2500 free queries – no credit card required

https://serper.dev/signup

copy key and keep in .env

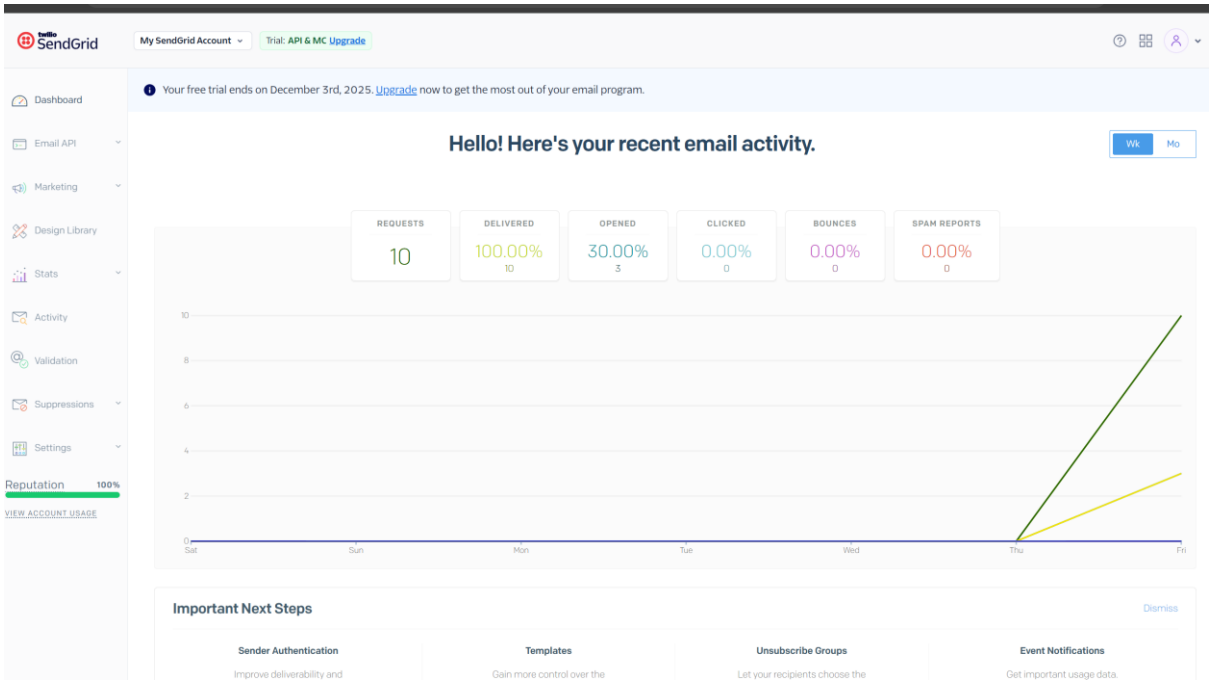
Key = SERPER_API_KEY





Send grid tool:

https://app.sendgrid.com/



Push over notification:

push over app as we did setup previously