

# Cơ bản về GraphBuilder

Trong LangGraph, bạn sẽ sử dụng một **GraphBuilder** để định nghĩa các bước (nodes), luồng xử lý (edges), điều kiện phân nhánh (routers), và điểm bắt đầu/kết thúc của một ứng dụng dựa trên LLM.

```
from langgraph.graph import StateGraph

builder = StateGraph(StateType)
```

Ở đây `StateType` là schema mô tả state (thường là TypedDict hoặc pydantic model).

Ví dụ

```
from typing import TypedDict, Literal

class DrinkState(TypedDict):
    preference: Literal["coffee", "tea", "unknown"]
```

## 1. builder.add\_node(name, node\_callable)

Thêm một bước xử lý (node) vào graph.

**Cú pháp:**

```
builder.add_node("tên_node", hàm_xử_lý)
```

Ví dụ

```
def ask_name(state):
    state["name"] = input("Tên bạn là gì? ")
    return state

builder.add_node("ask_name", ask_name)
```

## 2. builder.add\_edge(from\_node, to\_node)

Tạo liên kết (đường đi) giữa hai node.

**Cú pháp:**

```
builder.add_edge("node_nguồn", "node_đích")
```

**Ví dụ:**

```
builder.add_edge("ask_name", "ask_preference")
```

### 3. **builder.add\_conditional\_edges(from\_router\_node, conditions\_dict)**

Tạo đường đi có điều kiện (routing) từ một node tới nhiều node tùy thuộc vào kết quả của node đó.

**Cú pháp:**

```
builder.add_conditional_edges("router_node", {  
    "kết_quả_1": "node_1",  
    "kết_quả_2": "node_2",  
    ...  
})
```

**Ví dụ:**

```
builder.add_conditional_edges("router", {  
    "coffee_node": "coffee_node",  
    "tea_node": "tea_node",  
    "fallback_node": "fallback_node"  
})
```

=> Node "router" sẽ trả về một chuỗi (ví dụ "coffee\_node"), và flow sẽ đi đến node tương ứng.

### 4. **builder.set\_entry\_point(node\_name)**

Xác định điểm bắt đầu của graph.

Ví dụ

```
builder.set_entry_point("start")
```

### 5. **builder.set\_finish\_point(node\_name)**

Đánh dấu node kết thúc, tức là sau node này thì flow dừng lại.

Có thể có nhiều node kết thúc!

Ví dụ

```
builder.set_finish_point("coffee_node")
builder.set_finish_point("tea_node")
```

6. builder.compile()

Biên dịch graph của bạn thành một CompiledGraph có thể chạy được.

Ví dụ

```
app = builder.compile()
```

Sau đó

```
app.invoke({"preference": "unknown"})
```

7. Tổng kết

Hàm	Mô tả
<code>add_node(name, node_fn)</code>	Thêm một bước xử lý đơn vào graph
<code>add_edge(from_node, to_node)</code>	Tạo liên kết tuyến tính giữa 2 node
<code>add_conditional_edges(name, mapping)</code>	Tạo node phân nhánh có điều kiện
<code>set_entry_point(name)</code>	Xác định điểm bắt đầu của flow
<code>set_finish_point(name)</code>	Xác định điểm kết thúc
<code>add_branch(name, sub_graph)</code>	Thêm một graph phụ (dạng phân nhánh) vào một node
<code>add_recursion(name, recursive_graph)</code>	Cho phép gọi đệ quy một graph con
<code>compile()</code>	Biên dịch toàn bộ graph thành một ứng dụng chạy được
<code>add_generator_node(name, generator_node)</code>	Thêm một node sử dụng generator (yield từng bước)
<code>add_parallel_nodes(nodes: dict)</code>	Chạy nhiều node song song trong một bước
<code>add_conditional_edges_with_default(name, conditions, default)</code>	Giống <code>add_conditional_edges</code> nhưng có nhánh mặc định
<code>add_stateful_node(name, node_fn)</code>	Dùng khi node cần giữ trạng thái riêng biệt
<code>add_async_node(name, async_node)</code>	Thêm node xử lý bất đồng bộ ( <code>async def</code> )

Tác giả: Đỗ Ngọc Tú  
Công Ty Phần Mềm **VHTSoft**

---

Phiên bản #2

Được tạo 18 tháng 4 2025 08:08:23 bởi Đỗ Ngọc Tú

Được cập nhật 21 tháng 4 2025 03:10:10 bởi Đỗ Ngọc Tú