Langgraph: The Secret to Building Intelligent Agents

CISCO Live!

Jesus Illescas
Developer Advocate @netcode.rocks bsky

CISCOU-3005

Cisco Webex App

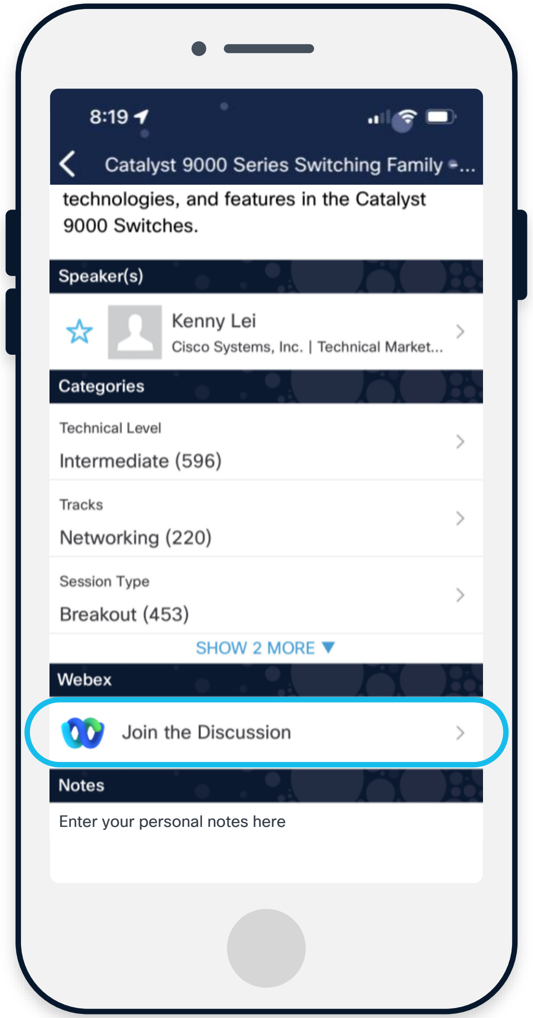
Questions?

Use Cisco Webex App to chat with the speaker after the session

How

1. Find this session in the Cisco Live Mobile App
2. Click "Join the Discussion"
3. Install the Webex App or go directly to the Webex space
4. Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until June 13, 2025.



technologies, and features in the Catalyst 9000 Switches.

Speaker(s)

Kenny Lei
Cisco Systems, Inc. | Technical Market...

Categories

Technical Level
Intermediate (596)

Tracks
Networking (220)

Session Type
Breakout (453)

Webex

Join the Discussion

Notes

Enter your personal notes here

https://ciscolive.ciscoevents.com/ciscolivebot/**#CISCOU-3005**

Agenda

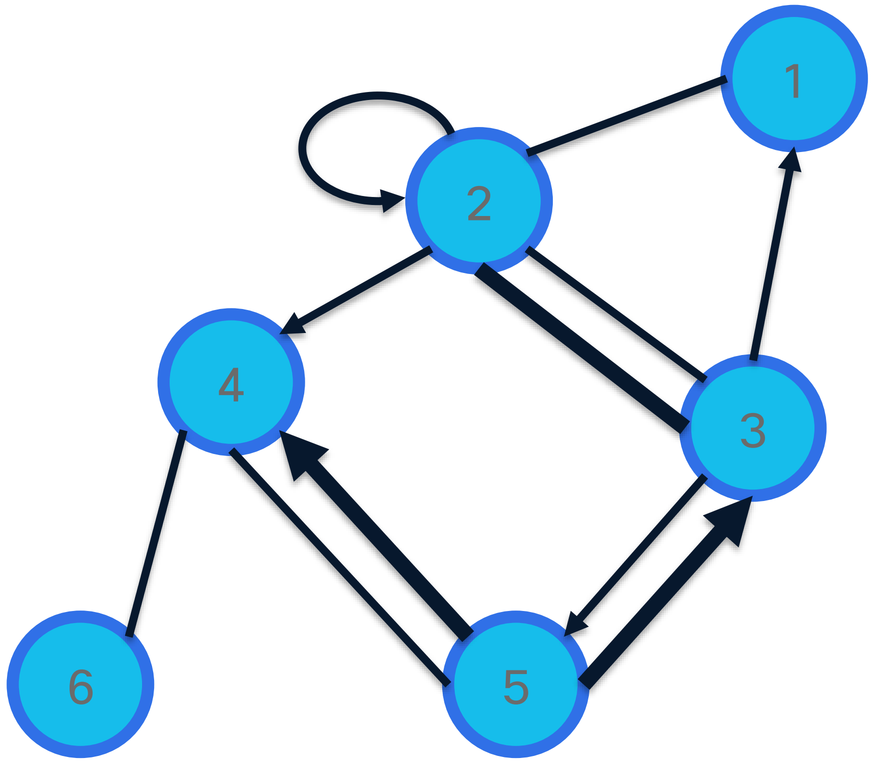
Intro

Why is a graph useful?

For Building!

- Algorithms.
- State machines.
- Workflows.

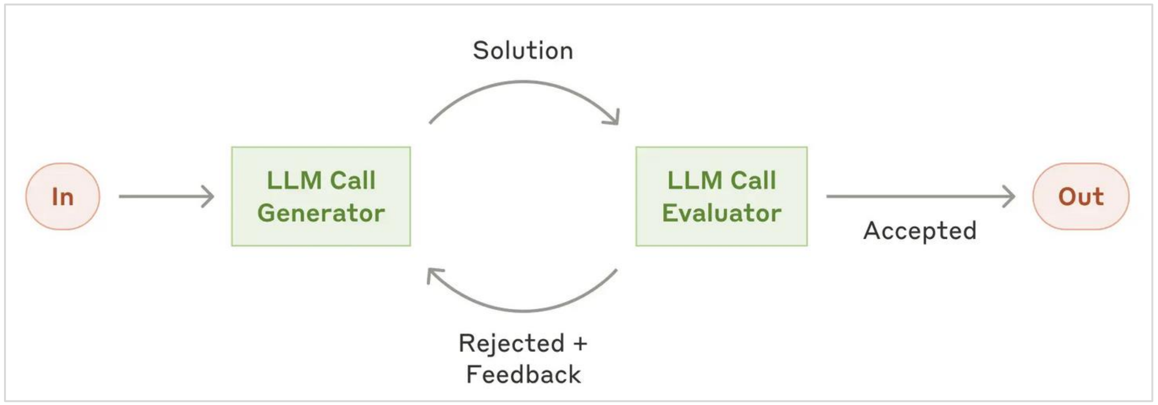
You can control the flow of the graph.

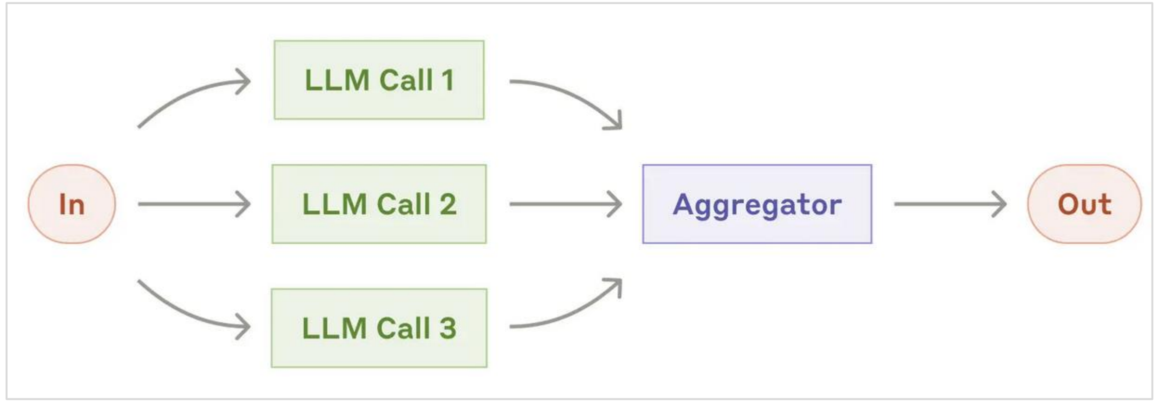


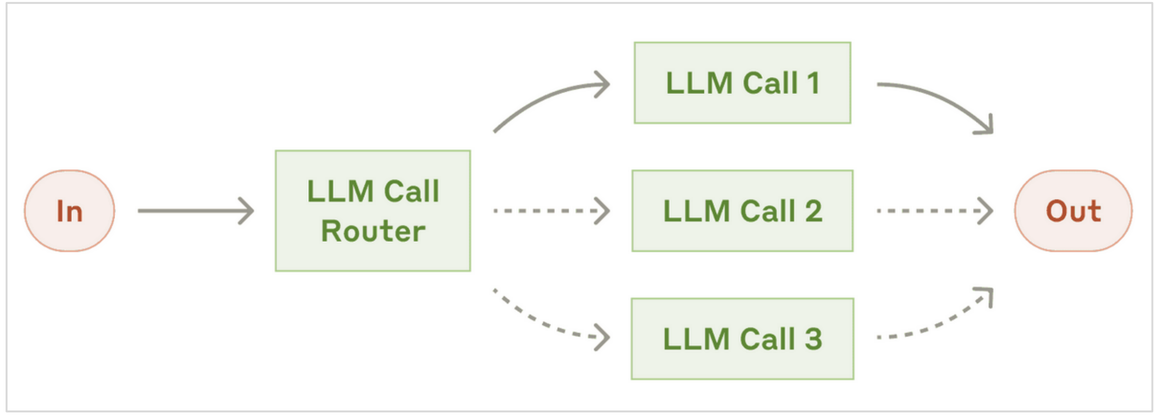
https://en.wikipedia.org/wiki/Graph_theory

Building like what?

Building Effective Agents







https://www.anthropic.com/engineering/building-effective-agents
https://langchain-ai.github.io/langgraph/tutorials/workflows/

What's out there

Code

LangGraph

https://www.langchain.com/langgraph

PydanticAI

https://ai.pydantic.dev/

crewai

https://www.crewai.com/

OpenAI Agents SDK

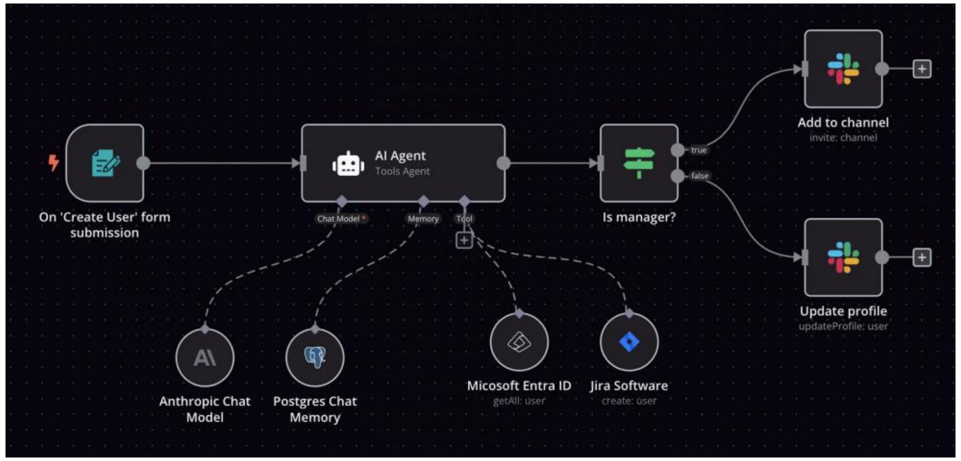
https://openai.github.io/openai-agents-python/

LlamaIndex

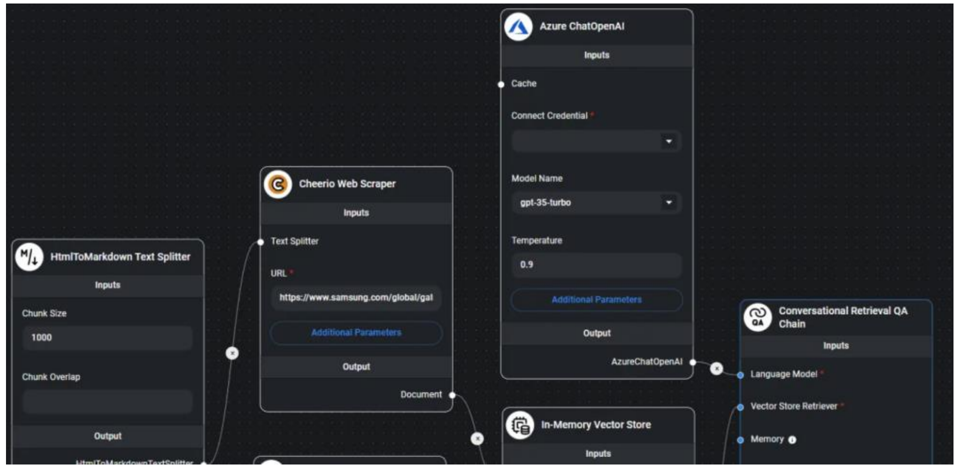
https://www.llamaindex.ai

Not a completed list…

No-Code / Low-Code



https://n8n.io/



https://flowiseai.com/

Multi-agent orchestration

Connect Agents from different frameworks

AGNTCY

https://agntcy.org/

https://github.com/google/A2A

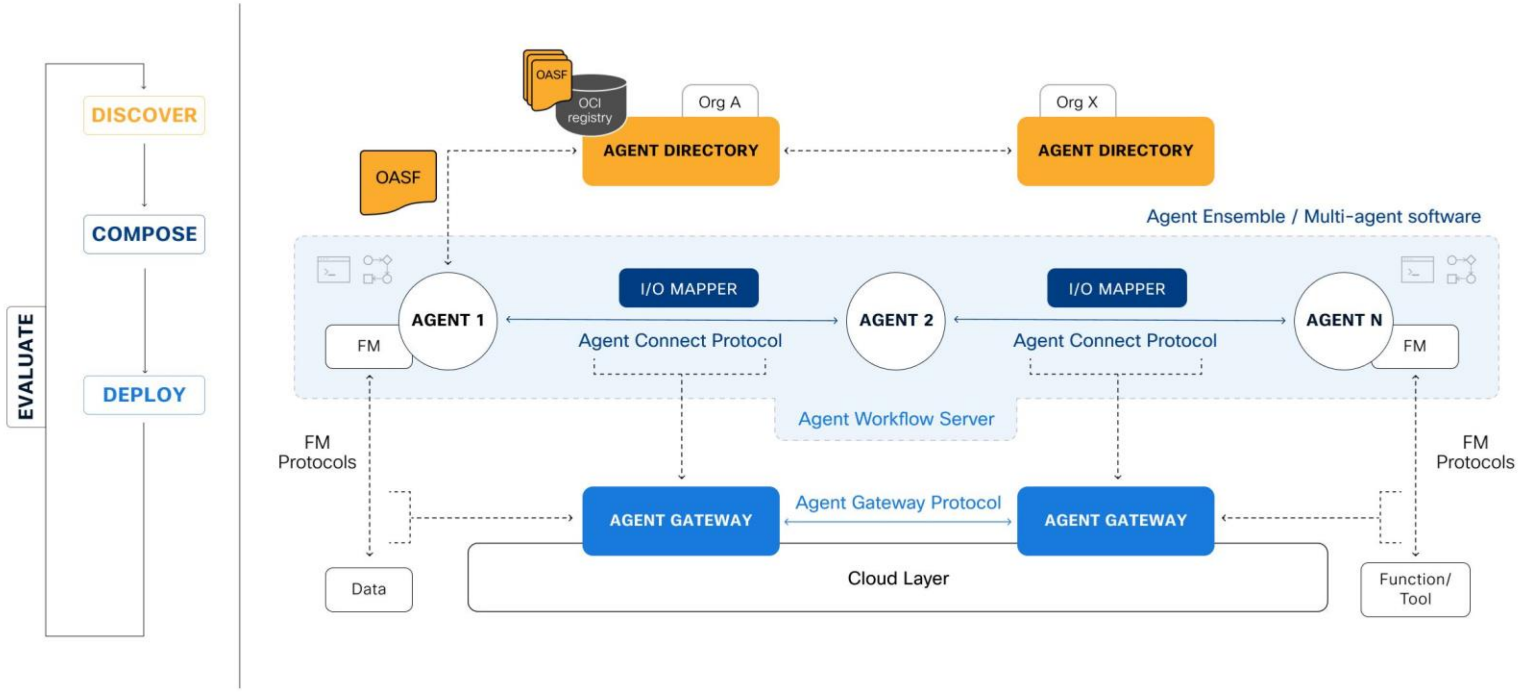
BeeAI Framework

https://i-am-bee.github.io/beeai-framework/

NVIDIA AgentIQ

https://github.com/NVIDIA/AgentIQ/

Still early environment...not a completed list

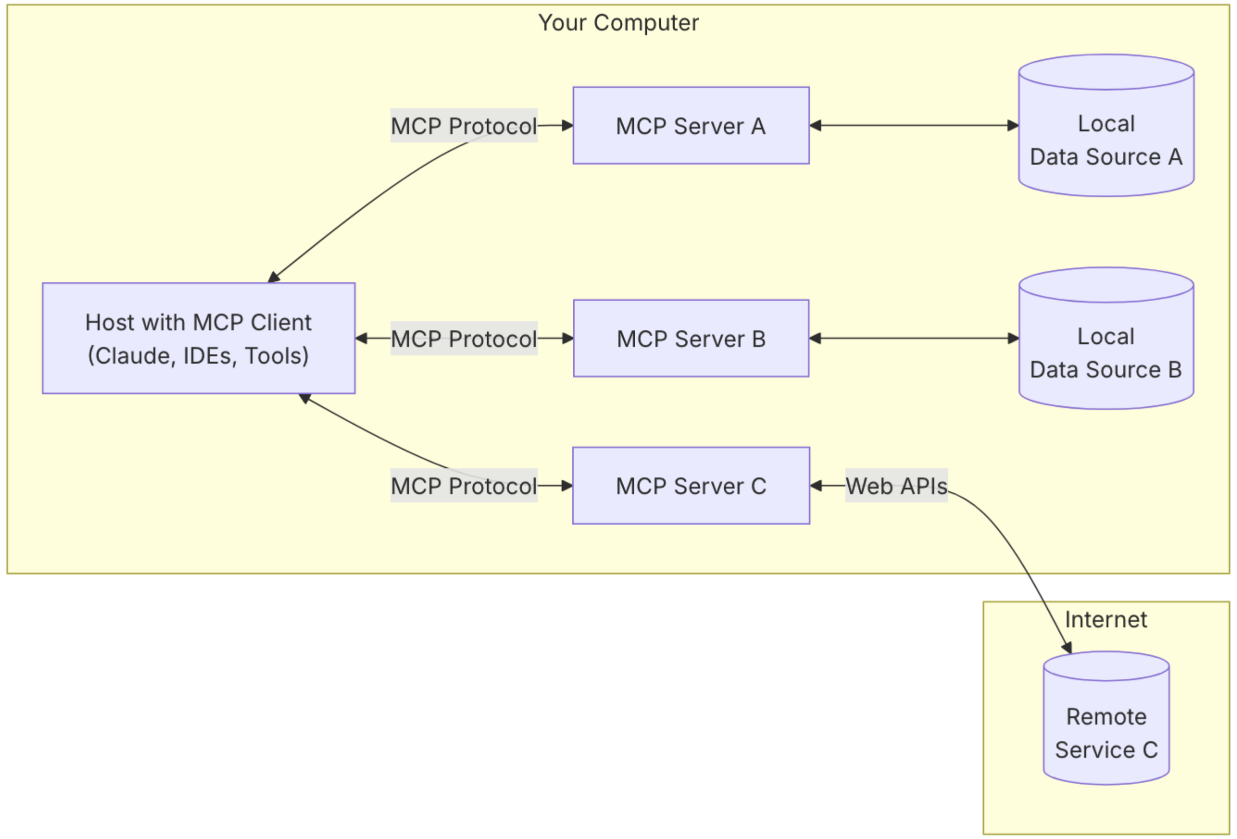


Model Context Protocol

Provide Context to your agent

- Tools.
- Resources.
- Prompts.

Tip, use official SDKs.



https://modelcontextprotocol.io/introduction

Core Concepts

Components

Nodes. Logic of your agents.

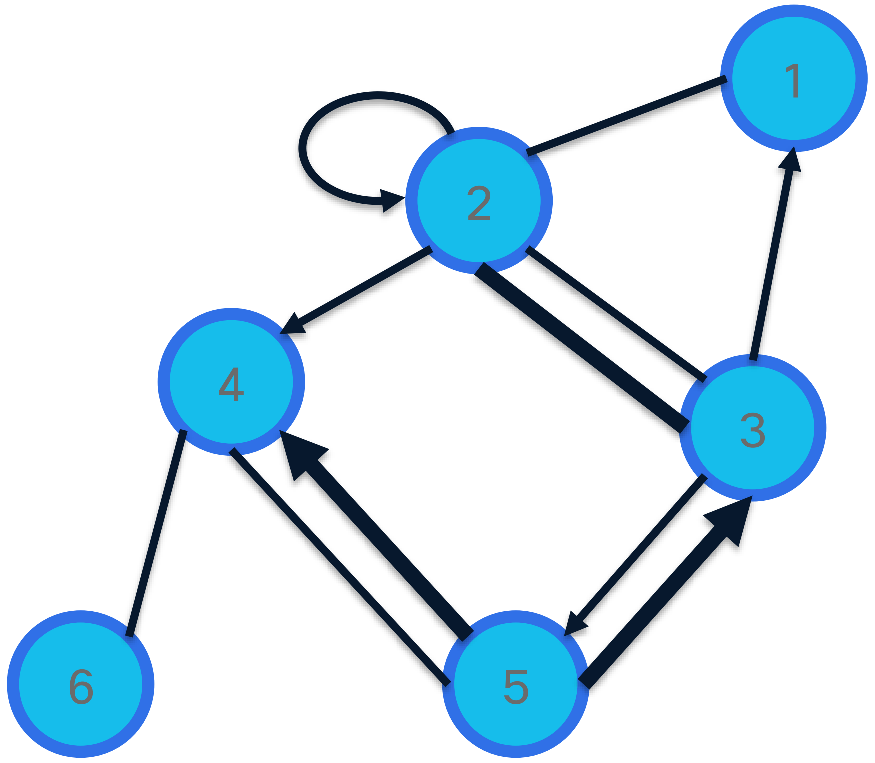
- This is where you call your agent(s).

Edges. Determines which Node to execute next.

- Control Flow.

State. Python object shared among nodes.

- This is Key!



https://langchain-ai.github.io/langgraph/concepts/low_level/

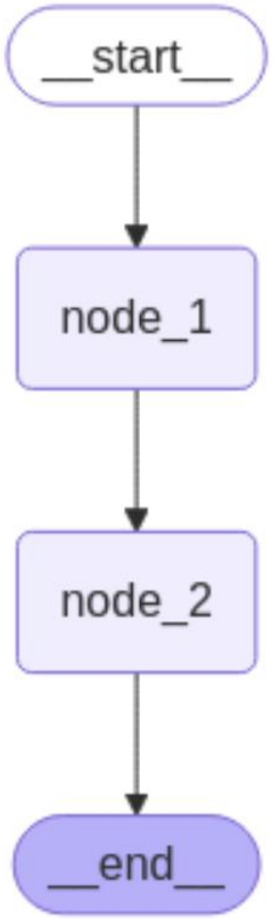
Nodes

```
from IPython.display import Image, display
from langgraph.graph import StateGraph
```

```
class State(TypedDict):
  graph_state: str
```

```
def node_1(state):
  print(f"-- Node 1 {state[‘graph_state’]} --")

def node_2(state):
  print(f"-- Node 2 {state[‘graph_state’]} --")
```

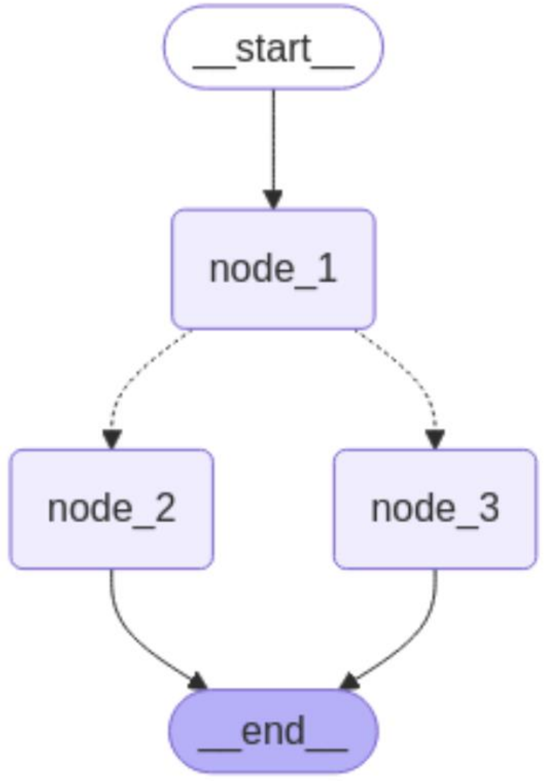
```
builder = StateGraph(State)
builder.add_node("node_1", node_1)
builder.add_node("node_2", node_2)

builder.set_entry_point("node_1")
builder.add_edge("node_1", "node_2")
builder.set_finish_point("node_2")
graph = builder.compile()
```



https://github.com/langchain-ai/langchain-academy/blob/main/module-1/simple-graph.ipynb

Control Flow

```
import random
from typing import Literal
from langgraph.graph import StateGraph, START, END

def decide_mood(state) -> Literal["node_2", "node_3"]:
  user_input = state['graph_state’]
  if random.random() < 0.5:
    return "node_2"
return "node_3"

builder = StateGraph(State)
builder.add_node("node_1", node_1)
builder.add_node("node_2", node_2)
builder.add_node("node_3", node_3)

builder.add_edge(START, "node_1")
builder.add_conditional_edges("node_1", decide_mood)
builder.add_edge("node_2", END)
builder.add_edge("node_3", END)

graph = builder.compile()
```



https://github.com/langchain-ai/langchain-academy/blob/main/module-1/simple-graph.ipynb

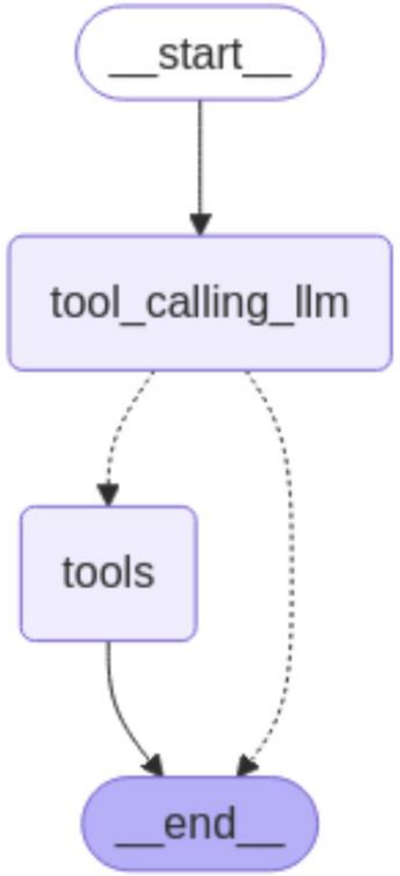
Tools

```
from langgraph.graph import StateGraph, START, END,
MessagesState
from langgraph.prebuilt import ToolNode,
tools_condition
from langchain_openai import ChatOpenAI
```

```
def multiply(a: int, b: int) -> int:
  """Multiply a and b.

     Args:
       a: first int
       b: second int
  """
return a * b
```

```
llm = ChatOpenAI(model="gpt-4o")
llm_with_tools = llm.bind_tools([multiply])
```

```
# Node
def tool_calling_llm(state: MessagesState):
  return {"messages":
     [llm_with_tools.invoke(state["messages"])]}
```



https://github.com/langchain-ai/langchain-academy/blob/main/module-1/router.ipynb

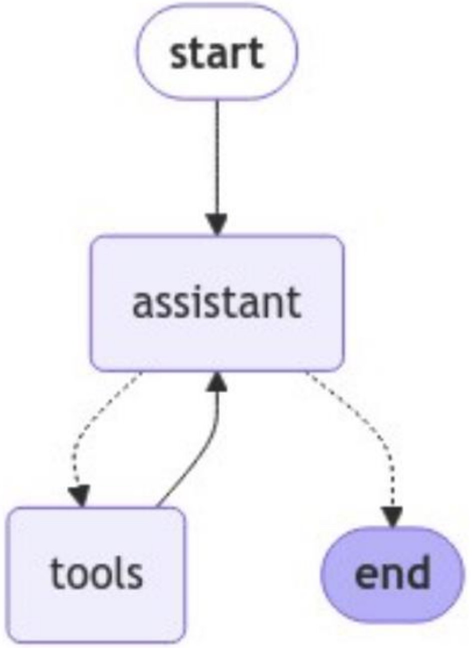
MCP Integration

```python
from langchain_mcp_adapters.client import MultiServerMCPClient
from langgraph.prebuilt import create_react_agent

async with MultiServerMCPClient(
    {
      "math": {
        "command": "python",
        "args": ["/path/to/math_server.py"],
        "transport": "stdio",
      },
    }
) as client:
    agent = create_react_agent(
      "anthropic:claude-3-7-sonnet-latest", client.get_tools()
    )
    math_response = await agent.ainvoke(
      {"messages": [{"role": "user", "content": "what's (3 + 5) x 12?"}]}
)
```

https://github.com/langchain-ai/langchain-mcp-adapters

Run the graph

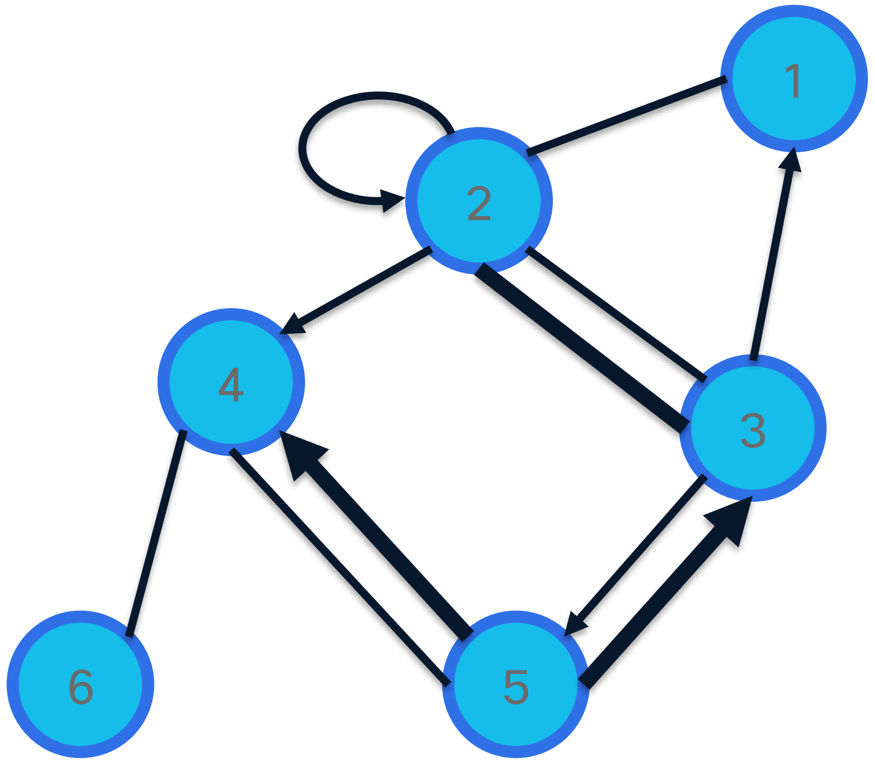
```
builder = StateGraph(MessagesState)

builder.add_node("assistant", assistant)
builder.add_node("tools", ToolNode(tools))

builder.add_edge(START, "assistant")
builder.add_conditional_edges("assistant",
                               tools_condition,
                               )
builder.add_edge("tools", "assistant")
graph = builder.compile()

messages = [HumanMessage(content="Add 3 and 4.
 Multiply the output by 2. Divide the output by 5")]

messages = graph.invoke({"messages": messages})

for m in messages['messages’]:
  m.pretty_print()
```



https://github.com/langchain-ai/langchain-academy/blob/main/module-1/agent.ipynb

Other Components

- Structure Output
- Memory
- Human in the loop
- Evals
- Subgraphs
- Multi-agent
- And many more..

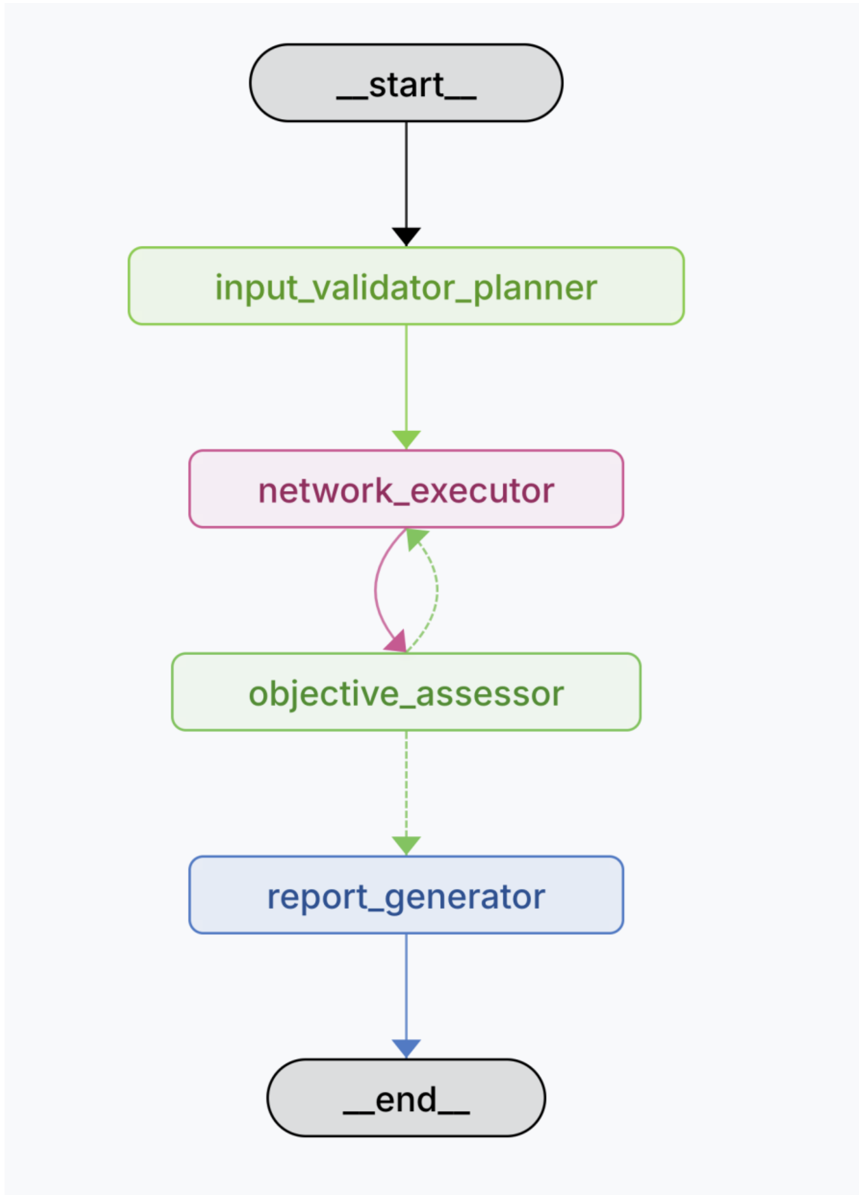


https://langchain-ai.github.io/langgraph/

Demo

Demo

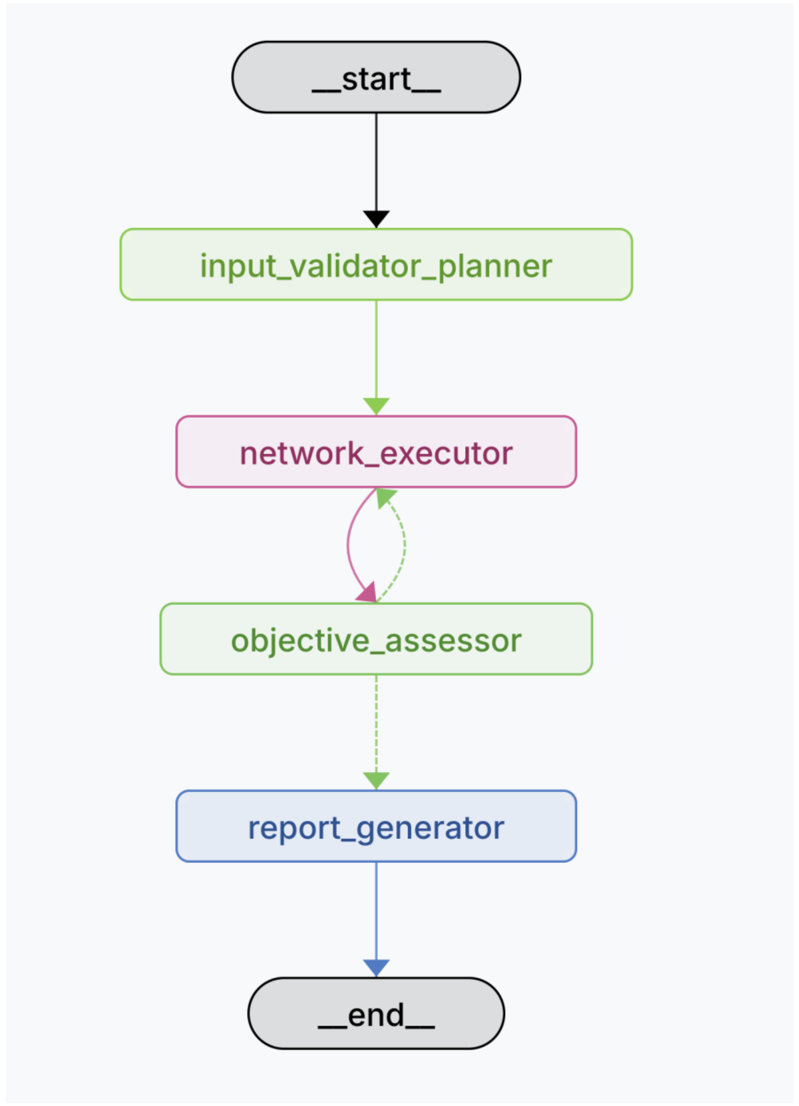
- Objective; extract state & configuration from the network.
- gGNMI tool connected via MCP.
- XRd topology.



Wrap up

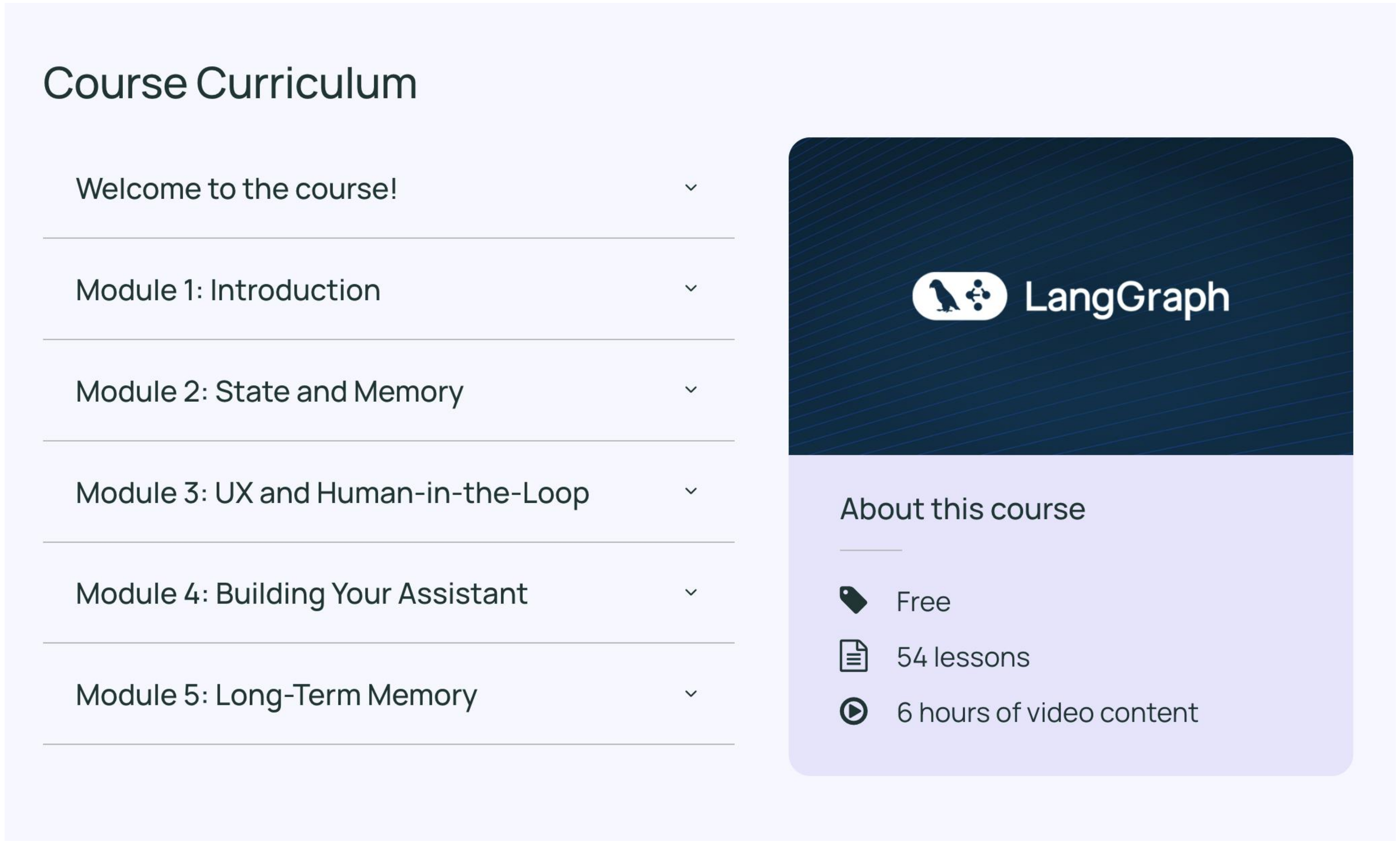
Leassons Learned

- Control the State and Flow with graphs.
- Agent workflow is easier with a Graph.
- An observability tool is crucial.
- The learning curve can be challenging.



Call to Action

- Check out the Langgraph course from Langgchain.
- Write your own graph!


Course Curriculum

Welcome to the course!

Module 1: Introduction

Module 2: State and Memory

Module 3: UX and Human-in-the-Loop

Module 4: Building Your Assistant

Module 5: Long-Term Memory

LangGraph

About this course

Free

54 lessons

6 hours of video content

https://academy.langchain.com/courses/intro-to-langgraph

Complete your session evaluations

Complete a minimum of 4 session surveys and the Overall Event Survey to be entered in a drawing to win 1 of 5 full conference passes to Cisco Live 2026.

Earn 100 points per survey completed and compete on the Cisco Live Challenge leaderboard.

Level up and earn exclusive prizes!

Complete your surveys in the Cisco Live mobile app.

Continue your education

Visit the Cisco Showcase for related demos

Book your one-on-one Meet the Engineer meeting

Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs

Visit the On-Demand Library for more sessions at www.CiscoLive.com/on-demand

Contact me at: @netcode.rocks bsky

Thank you

CISCO Live!