

멀티 에이전트의 구현 및 활용

Table of Contents

- 00** 자기소개
- 01** Agent란 무엇인가?
- 02** Agent가 필요할때
- 03** Agent 프레임워크 패턴
- 04** 멀티 에이전트

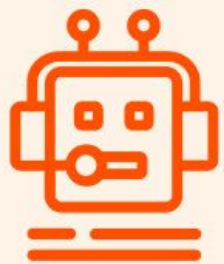
Retail Tech 분야의 AI/ML을 리딩하고 있어요

AI/ML 기술을 통해 유통 분야의 고객 경험 향상, 업무 프로세스 개선, 생산성 향상 등에 기여하고 싶습니다...

특징	AI (인공지능)	자동화 cf) RPA
주요 기능	데이터 분석, 예측, 의사결정 지원	반복적 작업 수행, 프로세스 간소화
적용 분야	개인화된 추천, 수요 예측, 가격 최적화	재고 관리, 결제 프로세스, 물류 운영
고객 경험 개선	맞춤형 서비스, 24/7 고객 지원	셀프 체크아웃, 빠른 주문 처리
재고 관리	수요 예측을 통한 최적화, 동적 재고 조정	자동 재고 추적, 발주 프로세스 자동화
마케팅 활용	개인화된 마케팅 캠페인, 고객 행동 분석	자동화된 이메일 마케팅, 소셜 미디어 게시
운영 효율성	복잡한 문제 해결, 전략적 의사결정 지원	일상적인 작업의 자동화, 인적 오류 감소
데이터 활용	빅데이터 분석, 패턴 인식, 예측 모델링	데이터 수집 및 기본적인 보고 자동화
적응성	새로운 상황에 학습 및 적응 가능	미리 정의된 규칙에 따라 작동
투자 비용	초기 투자 비용이 높음, 지속적인 학습 필요	상대적으로 낮은 초기 비용, 유지 관리 간단
미래 전망	지속적인 혁신과 새로운 적용 분야 확장	기존 프로세스의 효율성 향상에 중점

What is AI?

What is AI? ANI vs. AGI vs. ASI



Artificial narrow intelligence (ANI)

Designed to perform specific tasks



Artificial general intelligence (AGI)

Can behave in a human-like way across all tasks



Artificial super intelligence (ASI)

Smarter than humans—the stuff of sci-fi



AI(Artificial Intelligence) 인공지능

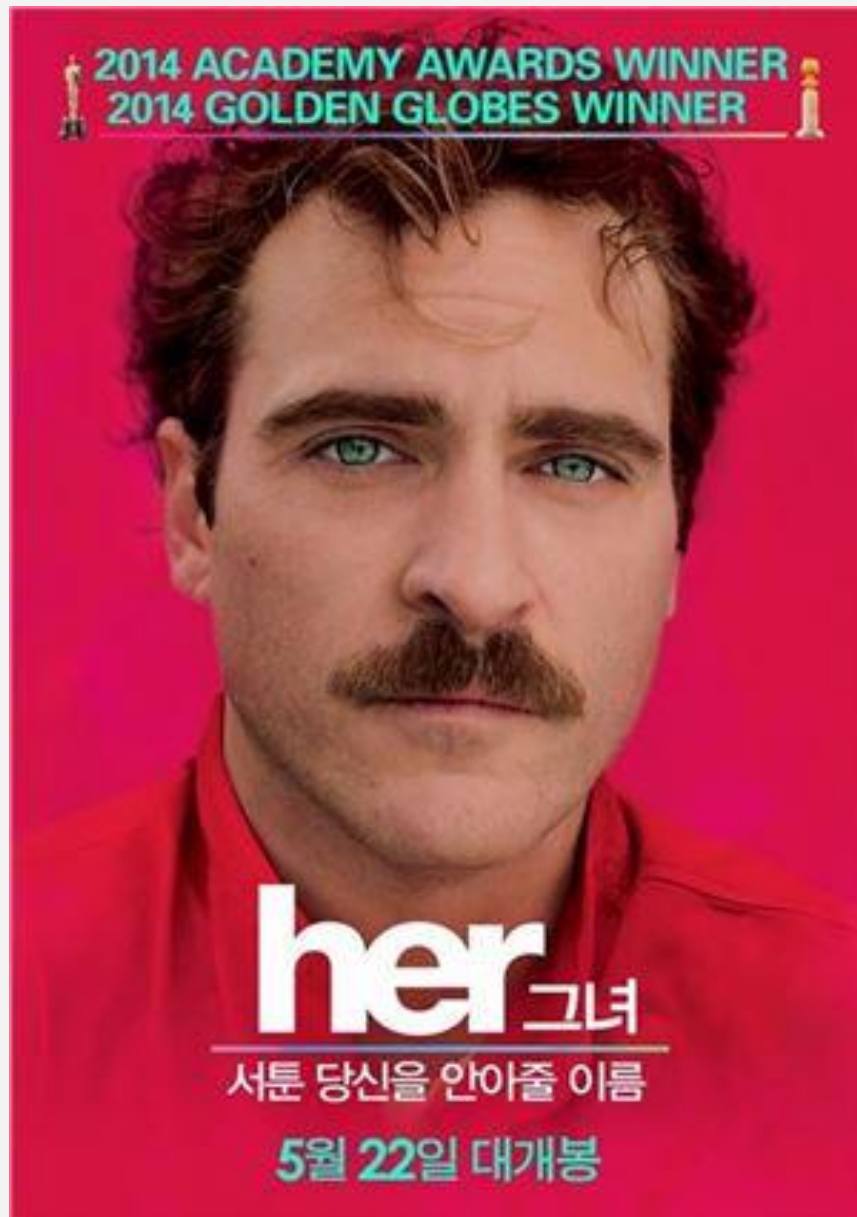
학습 알고리즘에서 ‘**학습(Learning)**’은 주어진 데이터를 분석해 특정 작업을 수행할 수 있는 **규칙이나 패턴을 발견하는 과정**을 뜻한다. 학습 알고리즘의 역할은 데이터로부터 주어진 과업을 수행할 알고리즘을 찾아내는 것이다. 예측 알고리즘에서 ‘**예측(Prediction)**’은 학습된 모델을 사용하여 새로운 데이터에 대한 **결과를 추론하는 과정**을 말한다. 즉, 예측 알고리즘은 학습 알고리즘을 통해 학습되며, 특정 과업을 수행하는 역할을 한다.

- <https://news.skhynix.co.kr/post/all-around-ai-2>

AGI(Artificial General Intelligence) 범용 인공지능

인간 수준의 지능을 가진 인공지능을 말하며, 인간과 같이 추론, 이해, 학습, 감정, 특정 상황에 적응하는 능력 등이 있는 인공지능 현존하는 인공지능은 특정화 된(specific task)에 특화된 Narrow AI이며, OpenAI의 GPT-4 수준이 AGI의 초기 형태로 언급, GPT-4는 일부 표준화된 테스트에서 인간보다 더 뛰어난 수행 능력을 보여줌

영화 속 AGI vs ASI



에이전트란?

Generative AI agent can be defined *as an application* that *attempts to achieve a goal* by observing the world and acting upon it using the tools that it has at its disposal. - *Google White Paper Agents*

- <https://www.kaggle.com/whitepaper-agents>



General agent architecture

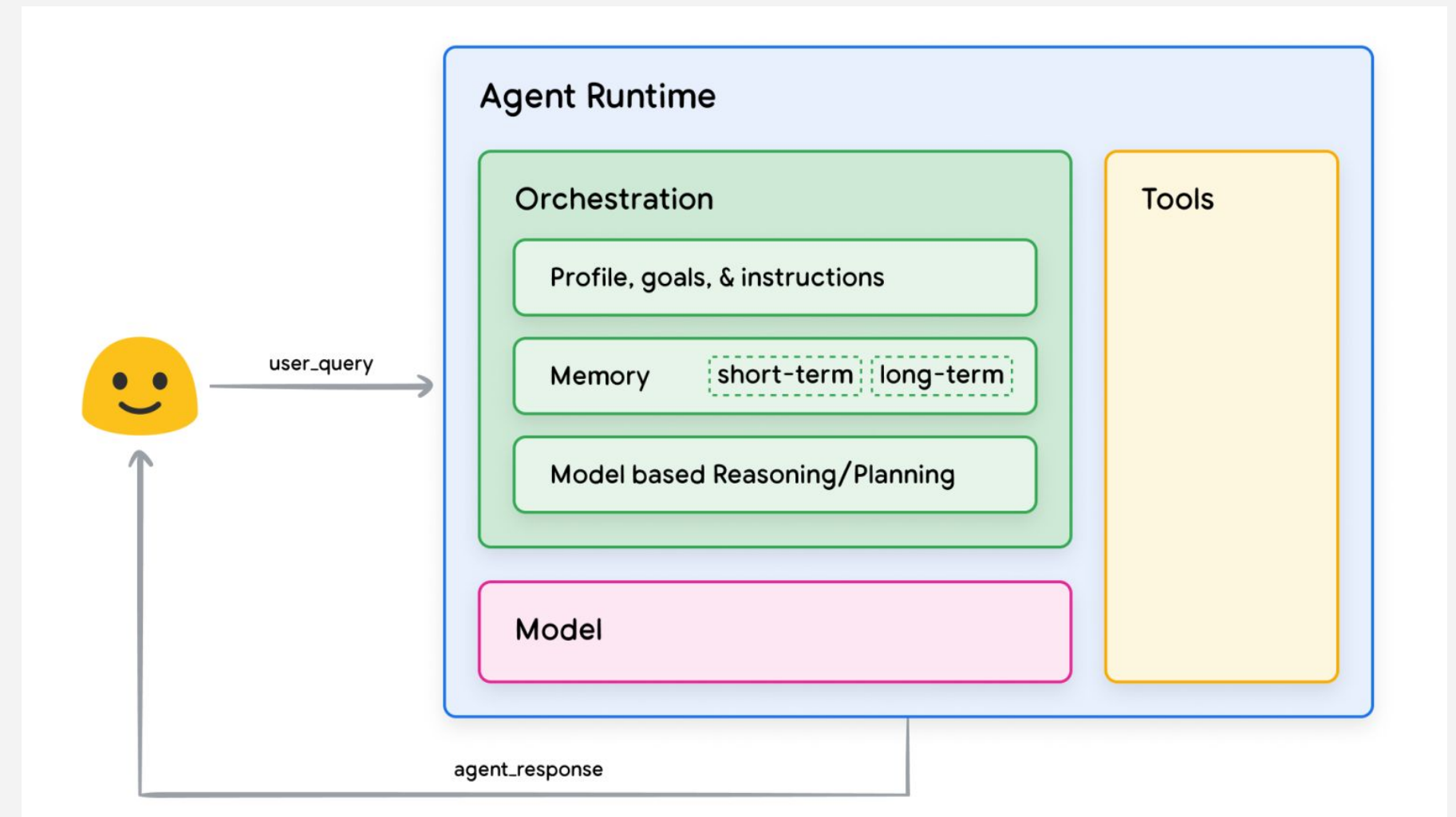
Focusing on the core functionalities, there are three essential components in an agent's cognitive architecture.

Model : Centralized decision maker

Tools : Empowering agents to interact with external data and services

Orchestration : cyclical process

- *Google White Paper Agents*



GenAI 의 뛰어난 역량

	2022-23 ¹	Jan 2025 ²
Anthropic	Claude <ul style="list-style-type: none"> • Not multimodal (text only) • Limited contextual understanding (difficulty with complex conversations) • No tool usage 	Claude 3.5 <ul style="list-style-type: none"> • Multimodal (text, audio, and images) • Enhanced contextual understanding and coherence during long interactions • Experimental computer usage capability for some users
Google Gemini	Google Bard <ul style="list-style-type: none"> • Not multimodal (text only) • Fair reasoning • Limited contextual understanding (difficulty with complex conversations) • Limited real-time data integration • Low personalization (limited adaptability) 	Gemini 2.0 Flash <ul style="list-style-type: none"> • Multimodal (text, audio, and images) • Advanced reasoning (capable of multistep problem-solving and nuanced analysis) • Enhanced contextual understanding (maintains coherence in long dialogues) • Real-time data integration (from Google Search) • Advanced personalization (user context)
Meta	Llama 1 <ul style="list-style-type: none"> • Not multimodal (text only) • Fair reasoning • Limited contextual understanding (difficulty with complex conversations) • No API access 	Llama 3.3 <ul style="list-style-type: none"> • Text-based (earlier versions were multimodal, LLaMa 3.2) • Advanced reasoning (capable of multistep problem-solving and nuanced analysis) • Enhanced contextual understanding (maintains coherence in long dialogues) • API access (tools for model and agent development)

Reasoning - 논리적 사고 과정

논리적 사고 과정을 모방하는 AI의 능력이다. 기존 지식을 바탕으로 새로운 결론을 도출하거나 문제를 해결하는 과정으로 연역적, 귀납적, 귀추적 추론 등 다양한 유형이 있으며 명시적인 규칙과 지식 표현을 사용하는 경향이 있다.

Inference - 결과 도출 행동 과정

학습된 AI 모델이 새로운 데이터에 대해 예측이나 결론을 내리는 과정이다. 훈련 과정에서 학습한 패턴을 새로운 데이터에 적용하는 것을 의미한다. 대규모 데이터셋에서 학습한 통계적 패턴을 기반으로 작동한다. 실시간으로 입력 데이터를 처리하고 결과를 출력하는 데 중점을 둔다.

- <https://www.mckinsey.com/capabilities/mckinsey-digital>

[/our-insights/superagency-in-the-workplace-empowering-people-to-unlock-ais-full-potential-at-work](https://www.mckinsey.com/our-insights/superagency-in-the-workplace-empowering-people-to-unlock-ais-full-potential-at-work)

GenAI 의 뛰어난 역량

	2022-23 ¹	Jan 2025 ²
Microsoft	Phi-1 <ul style="list-style-type: none">• Not multimodal (text only)• Fair reasoning (ie, limited to coding tasks)• Focused training (smaller, coding-focused data set)	Phi-4 <ul style="list-style-type: none">• Multimodal (text, audio, and images)• Advanced reasoning (capable of multistep problem-solving and nuanced analysis)• Comprehensive training (diverse data)
OpenAI	GPT-3.5 <ul style="list-style-type: none">• Not multimodal (text only)• Fair reasoning ability (eg, scored high on SAT, but bottom 10% on bar examination)• Limited contextual understanding (difficulty with coherence in complex conversations)• Standard API access (for text generation)	OpenAI o1 <ul style="list-style-type: none">• Multimodal (text and images)• Advanced reasoning (eg, top 10% on bar examination)• Enhanced contextual understanding (maintains coherence in long dialogues)• Advanced API access (supports multimodal inputs)

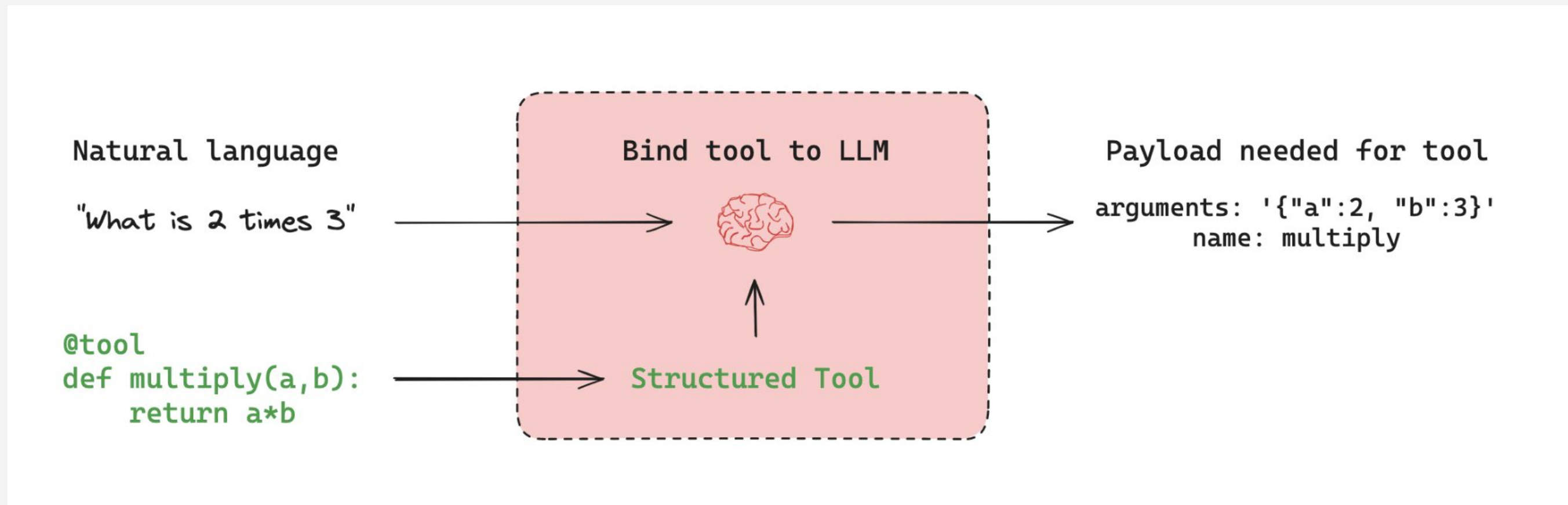
- <https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/superagency-in-the-workplace-empowering-people-to-unlock-ais-full-potential-at-work>

Model vs Agents

Models	Agents
Knowledge is limited to what is available in their training data.	Knowledge is extended through the connection with external systems via tools
Single inference / prediction based on the user query. Unless explicitly implemented for the model, there is no management of session history or continuous context. (i.e. chat history)	Managed session history (i.e. chat history) to allow for multi turn inference / prediction based on user queries and decisions made in the orchestration layer. In this context, a 'turn' is defined as an interaction between the interacting system and the agent. (i.e. 1 incoming event/ query and 1 agent response)
No native tool implementation.	Tools are natively implemented in agent architecture.
No native logic layer implemented. Users can form prompts as simple questions or use reasoning frameworks (CoT, ReAct, etc.) to form complex prompts to guide the model in prediction.	Native cognitive architecture that uses reasoning frameworks like CoT, ReAct, or other pre-built agent frameworks like LangChain.

여기서 잠깐 Langchain 으로 간단한 작업을 하나 만들어볼까?

Tools are useful whenever you want *an agent to interact with external systems*. External systems (e.g., APIs) often require a particular input schema or payload, rather than natural language. https://langchain-ai.github.io/langgraph/concepts/agenic_concepts/#tool-calling-agent



여기서 잠깐 Langchain 으로 간단한 작업을 하나 만들어볼까?

```
agent = initialize_agent(tools, llm, agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION, verbose=True)
```

```
agent.run("오늘 XX마트 갈건데 장볼꺼야 아이템 알려줘")
```

```
from langchain.agents import initialize_agent, Tool

tools = [
    Tool(
        name = "고객ID조회",
        func=lambda x: "default",
        description="고객ID를 조회합니다."
    ),
    Tool(
        name = "상품추천",
        func=rec.run,
        description="고객ID조회한 후 그 값을 기반으로 장보기를 하거나 방문할 경우 개인화 추천을 제공합니다."
    ),
    Tool(
        name = "상품검색",
        func=src.run,
        description="고객에게 해당 키워드에 대한 검색 결과를 알려줍니다. 가격정보를 통해 선택적으로 상품을 제안할 수 있습니다"
    ),
    Tool(
        name = "점포안내",
        func=src.run,
        description="고객에게 점포 위치나 테넌트 시설물 안내등을 제공할 수 있습니다. "
    )
]
```

```
> Entering new AgentExecutor chain...
```

```
고객 ID를 알아내야 상품추천을 할 수 있다.
```

```
Action: 고객ID조회
```

```
Action Input: 없음
```

```
Observation: default
```

```
Thought: 이제 고객의 ID를 알았으므로, 그들의 기본 정보를 기반으로 상품을 추천해줄 수 있다.
```

```
Action: 상품추천
```

```
Action Input: defaultdefault
```

```
Observation: ['상품: 8801114168899\n', '상품: 8801111951609\n', '상품: \n', '상품: 8807246093498\n', '상품: 8801051141566\n', '상품: 88010390\n', '상품: 8801051141566\n']
```

```
Thought: 이제 추천 상품 목록을 가지고 있으므로 고객에게 알려줄 수 있다.
```

```
Final Answer: 장보기에 추천드리는 상품들입니다: '상품: 8801114168899\n', '상품: 8801111951609\n', '상품: 8801051026047\n', '상품: 8807246093498\n', '상품: 8801051141566\n']
```

```
> Finished chain.
```

```
"장보기에 추천드리는 상품들입니다: '상품: 8801114168899\n', '상품: 8801111951609\n', '상품: 8801051026047\n', '상품: 8807246093498\n', '상품: 8801051141566\n']"
```


여기서 잠깐 Langchain 으로 간단한 작업을 하나 만들어볼까?

```
agent_chain.run(input="신라면 중 가장 싼거 얼마야?")
```

```
> Entering new AgentExecutor chain...
```

```
{  
  "action": "상품검색",  
  "action_input": "신라면"  
}
```

```
Observation: ['상품:농심 신라면블랙사발 101g\n가격 :1480\n', '상품:농심  
가격 :5520\n', '상품:농심 신라면블랙 두부김치 94g\n가격 :1480\n', '상품:농  
101g*6개\n가격 :8880\n', '상품:농심 신라면큰사발114g*6개\n가격 :6960\n']
```

```
Thought:{  
  "action": "Final Answer",  
  "action_input": "농심 신라면큰사발114g\n가격 :1000"  
}
```

```
> Finished chain.
```

```
'농심 신라면큰사발114g\n가격 :1000'
```

```
agent_chain.run(input="신라면 블랙 모두 알려줘")
```

```
> Entering new AgentExecutor chain...
```

```
{  
  "action": "상품검색",  
  "action_input": "신라면 블랙"  
}
```

```
Observation: ['상품:농심 신라면블랙사발 101g\n가격 :1480\n', '상품:농심 신라면  
블랙 134g*4개\n가격 :5480\n']
```

```
Thought:{  
  "action": "Final Answer",  
  "action_input": "농심 신라면블랙사발 101g\n가격 :1480\n농심 신라면블랙 두  
0"  
}
```

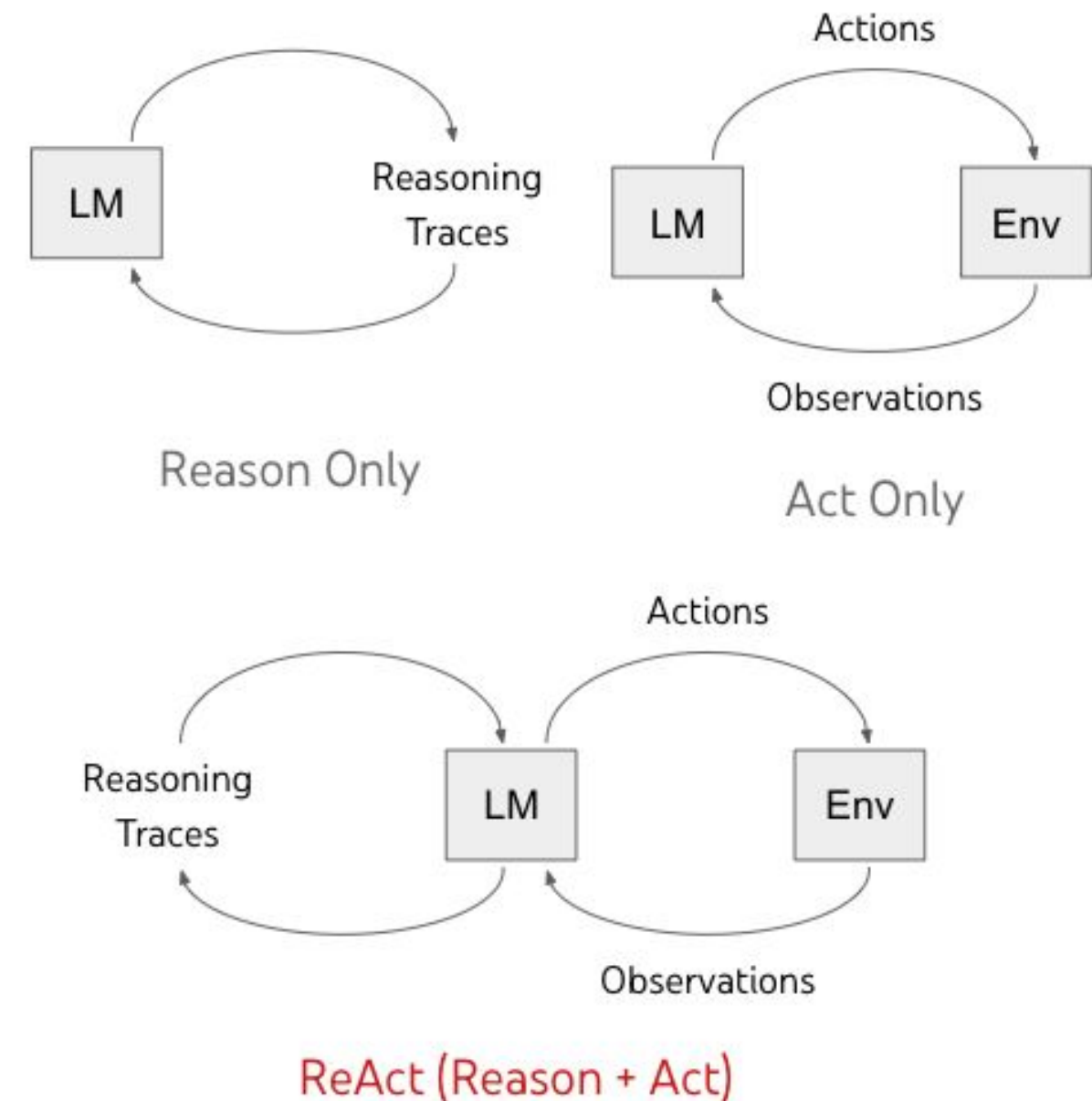
```
> Finished chain.
```

```
'농심 신라면블랙사발 101g\n가격 :1480\n농심 신라면블랙 두부김치 94g\n가격 :1480\
```

Planning

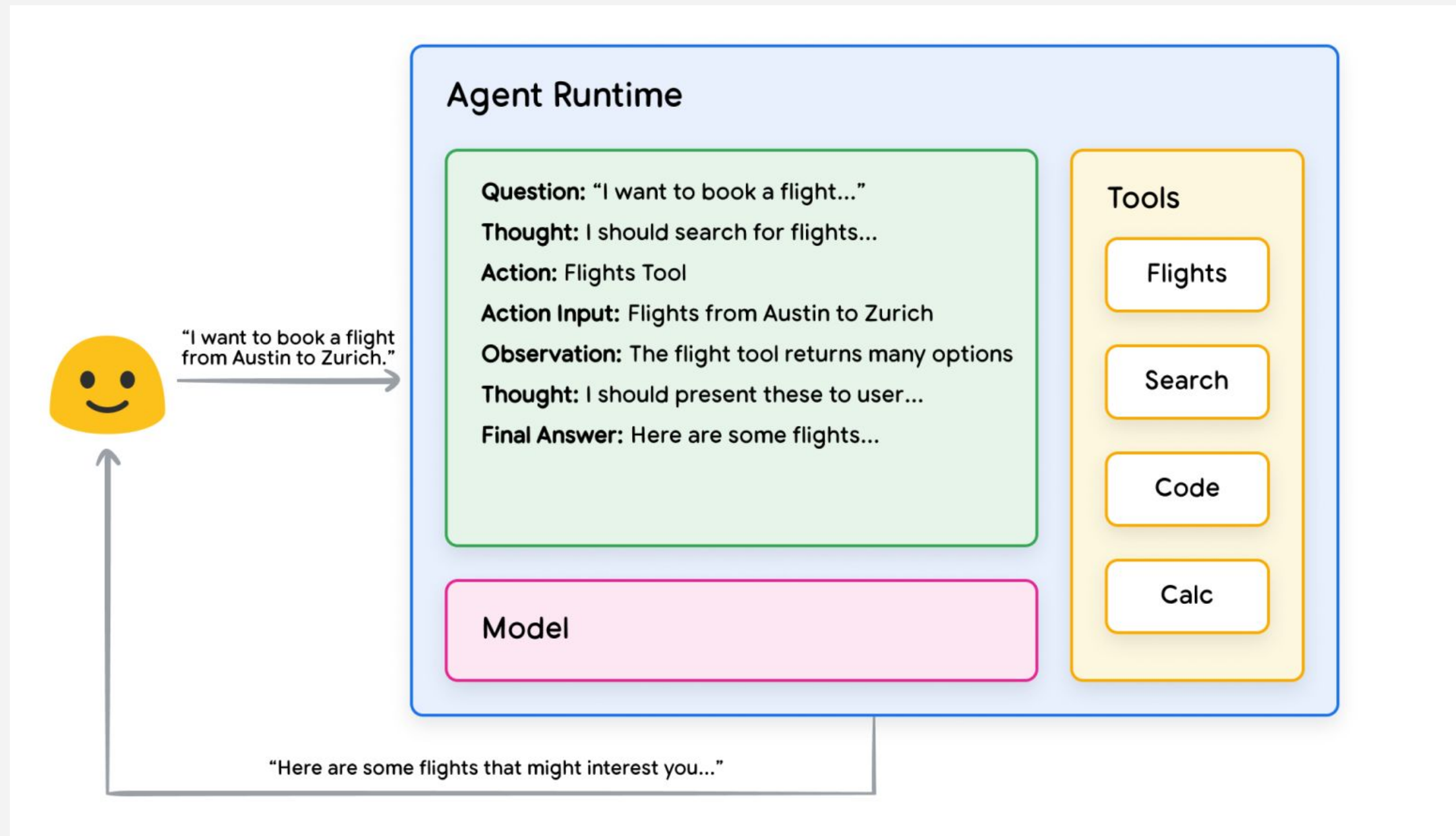
In the ReAct architecture, an *LLM is called repeatedly in a while-loop*. At each step the agent decides which tools to call, and what the inputs to those tools should be. Those tools are then executed, and the outputs are fed back into the LLM as observations. *The while-loop terminates when the agent decides it has enough information* to solve the user request and it is not worth calling any more tools.

<https://research.google/blog/react-synergizing-reasoning-and-acting-in-language-models/>



ReAct loop

- Google White Paper Agents



Agent vs RAG(Retrieval-Augmented Generation)

특징	Agent	RAG
주요 기능	자율적 의사결정 및 작업 수행	정보 검색 및 생성 결합
아키텍처	복잡한 의사결정 트리, 강화학습 모델	검색 모듈과 생성 모듈의 결합
데이터 처리	실시간 환경 데이터 처리	외부 지식베이스에서 관련 정보 검색
학습 방식	지속적 학습, 경험 기반 개선	검색 결과에 기반한 컨텍스트 학습
확장성	다중 작업 및 도메인 확장 가능	지식베이스 확장을 통한 성능 향상
자율성	높음 (독립적 의사결정 가능)	중간 (검색 결과에 의존)
정보 갱신	실시간 환경 상호작용을 통한 갱신	외부 데이터베이스 업데이트를 통한 갱신
복잡성	높음 (다단계 작업 처리 가능)	중간 (정보 검색 및 생성에 초점)
응용 분야	로봇 공학, 자동화 시스템, 게임 AI	질의응답 시스템, 정보 검색, 콘텐츠 생성
처리 속도	상황에 따라 다양 (복잡한 의사결정 필요 시 느릴 수 있음)	비교적 빠름 (효율적인 검색 및 생성)
메모리 사용	장기 메모리 및 경험 저장 필요	주로 외부 데이터베이스에 의존
오류 처리	자체 오류 감지 및 수정 능력	검색 결과의 정확도에 크게 의존

워크플로우와 에이전트 뭐가 다르죠?

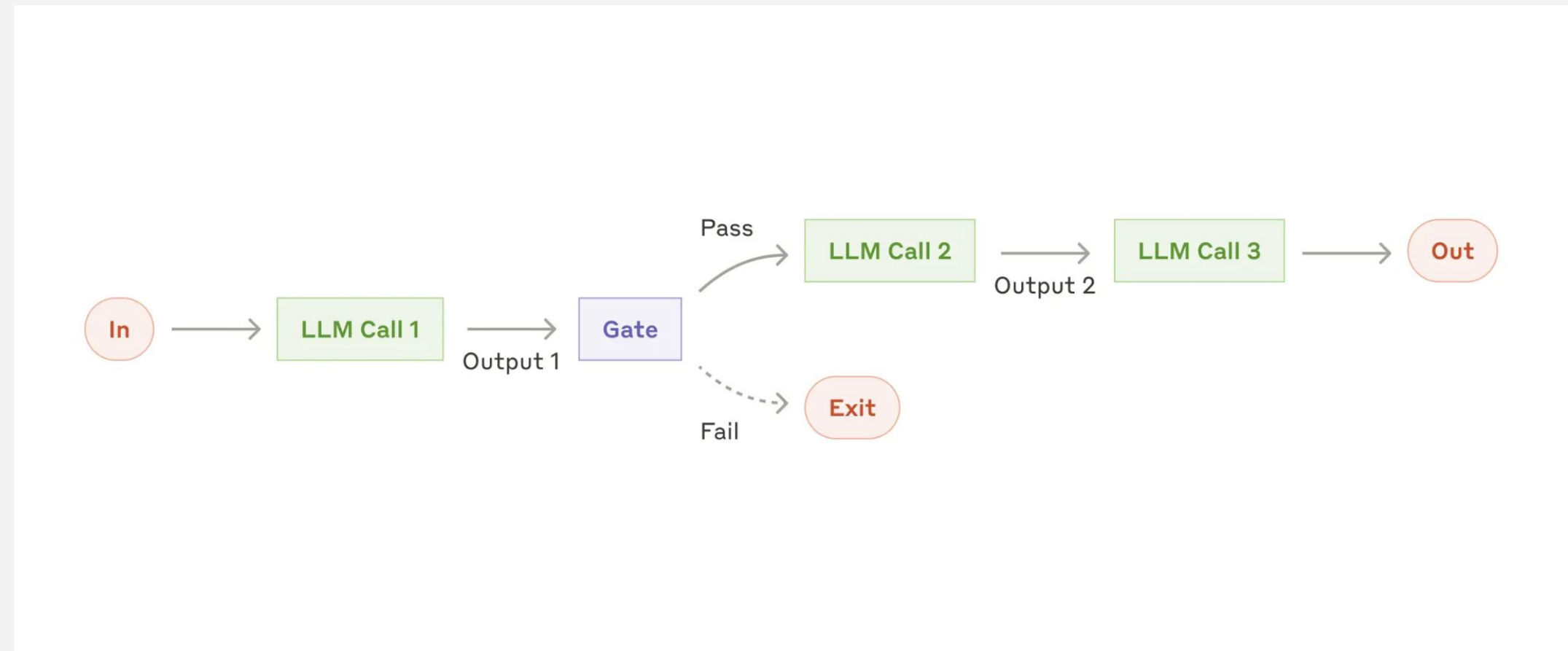
Workflows are systems where LLMs and tools are orchestrated through predefined code paths.

Agents, on the other hand, are systems where LLMs *dynamically* direct their own processes and tool usage, maintaining control over how they accomplish tasks. - Anthropic <https://www.anthropic.com/research/building-effective-agents>

워크플로우 - 프롬프트 체이닝

작업을 일련의 단계로 분해

- 마케팅 카피 생성 다른 언어 번역
- 문서 개요 작성



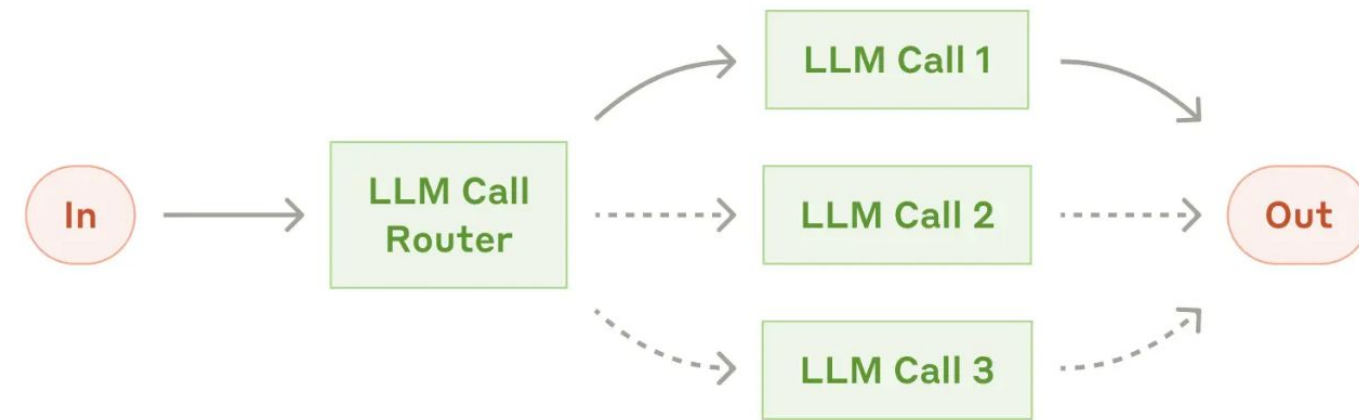
- Anthropic

<https://www.anthropic.com/research/building-effective-agents>

워크플로우 - 라우팅

입력을 분류하여 후속 작업 전달

- 다양한 고객 서비스 질의
(Q&A, 환불 요청, 기술지원 등)



- Anthropic

<https://www.anthropic.com/research/building-effective-agents>

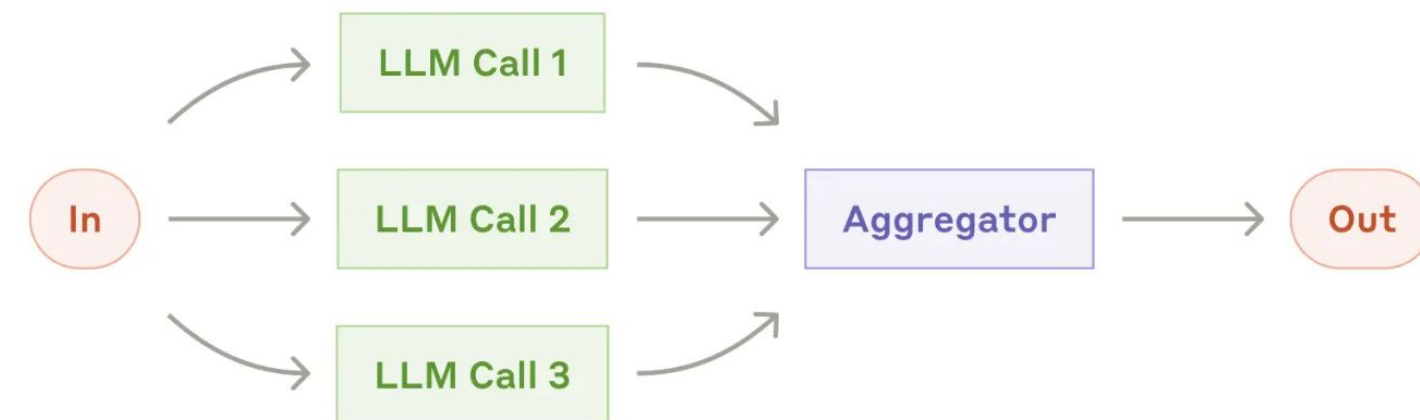
워크플로우 - 병렬화

동시에 수행 집계

- 섹션화: 가드레일, 자동화 평가
- 투표 : 취약점 코드 검토,
부적절 여부 평가

- Anthropic

<https://www.anthropic.com/research/building-effective-agents>



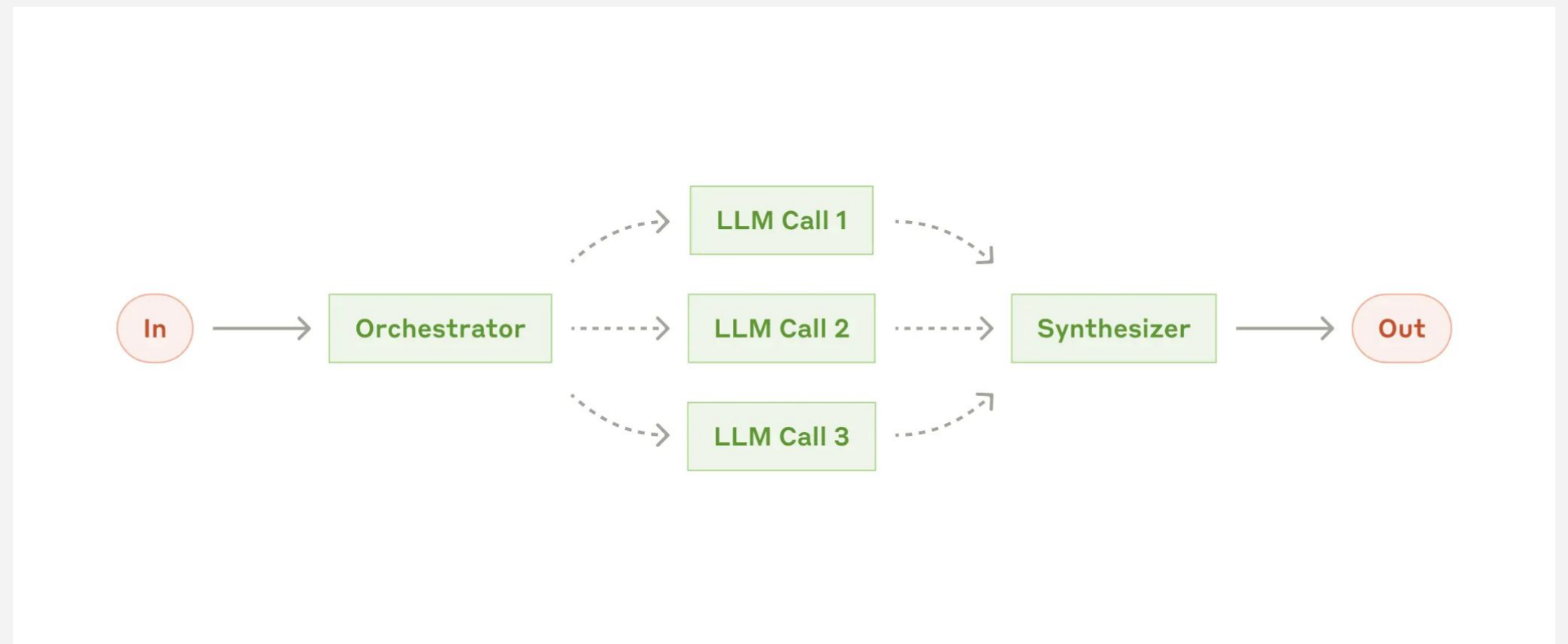
워크플로우 - 오케스트레이터 워커

중앙에서 동적 분할 결과 종합

- 여러 정보 수집, 분석, 관련성 검토

- Anthropic

<https://www.anthropic.com/research/building-effective-agents>



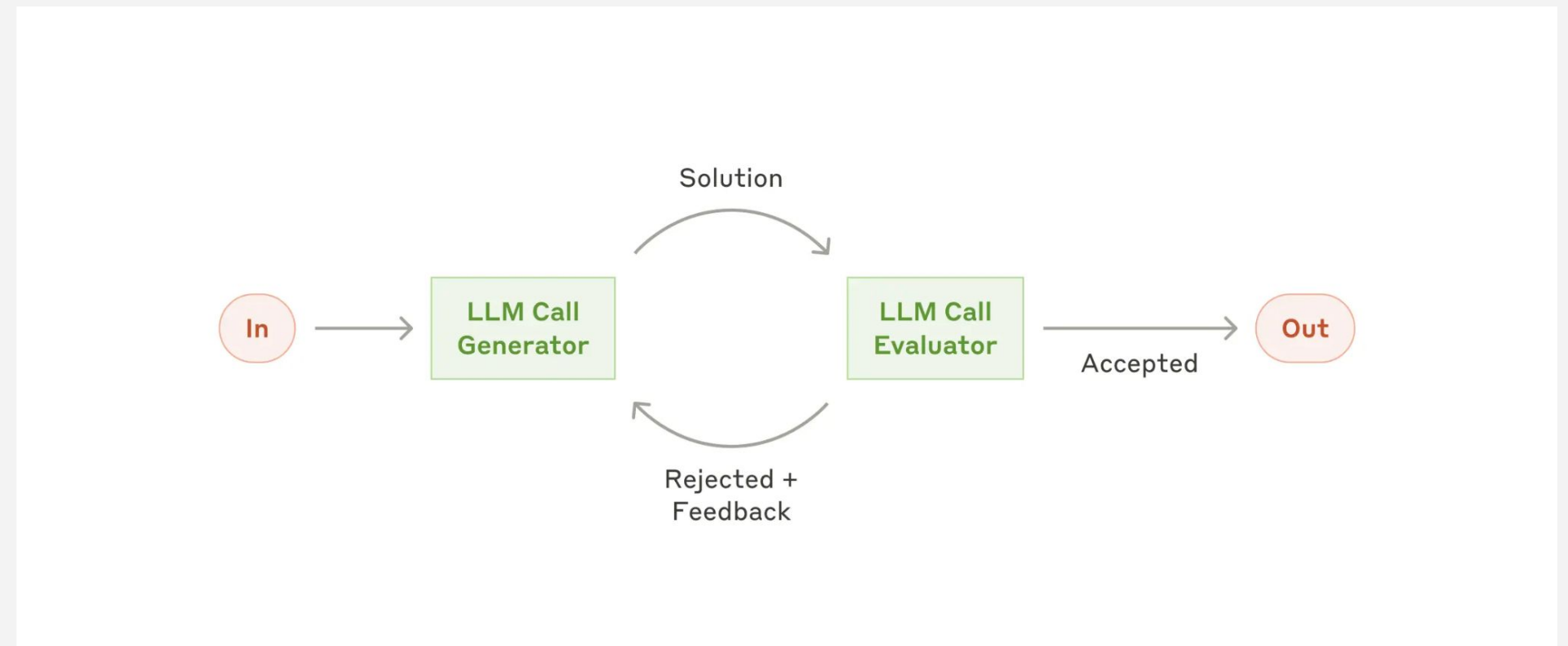
워크플로우 - 이벨류에이터-옵티마이저

응답 생성하고 다른 호출은 평가 및 피드백

- 비평을 포함한 문학 번역
- 복잡한 검색 작업

- Anthropic

<https://www.anthropic.com/research/building-effective-agents>



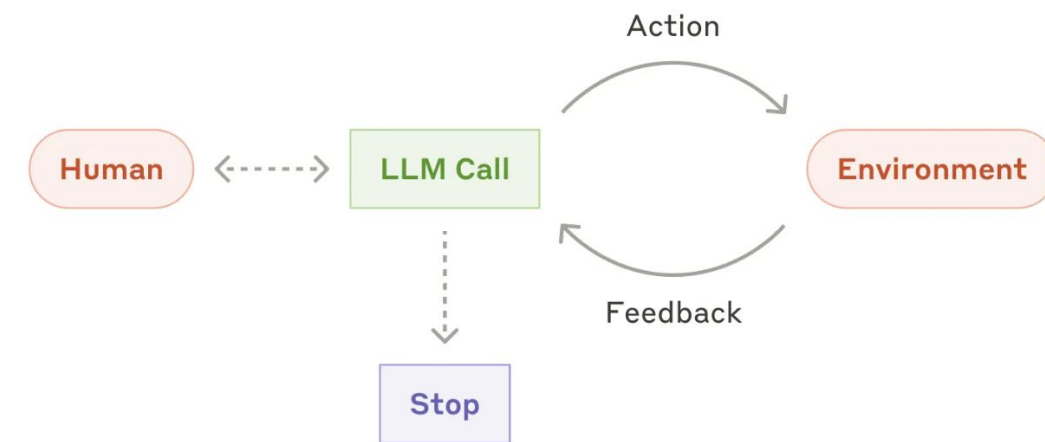
에이전트

대화형 토론을 통해 작업을 시작,
작업이 명확해지면 독립적으로 계획하고 작동

- 더 높은 비용
- 복합 오류의 가능성

- Anthropic

<https://www.anthropic.com/research/building-effective-agents>

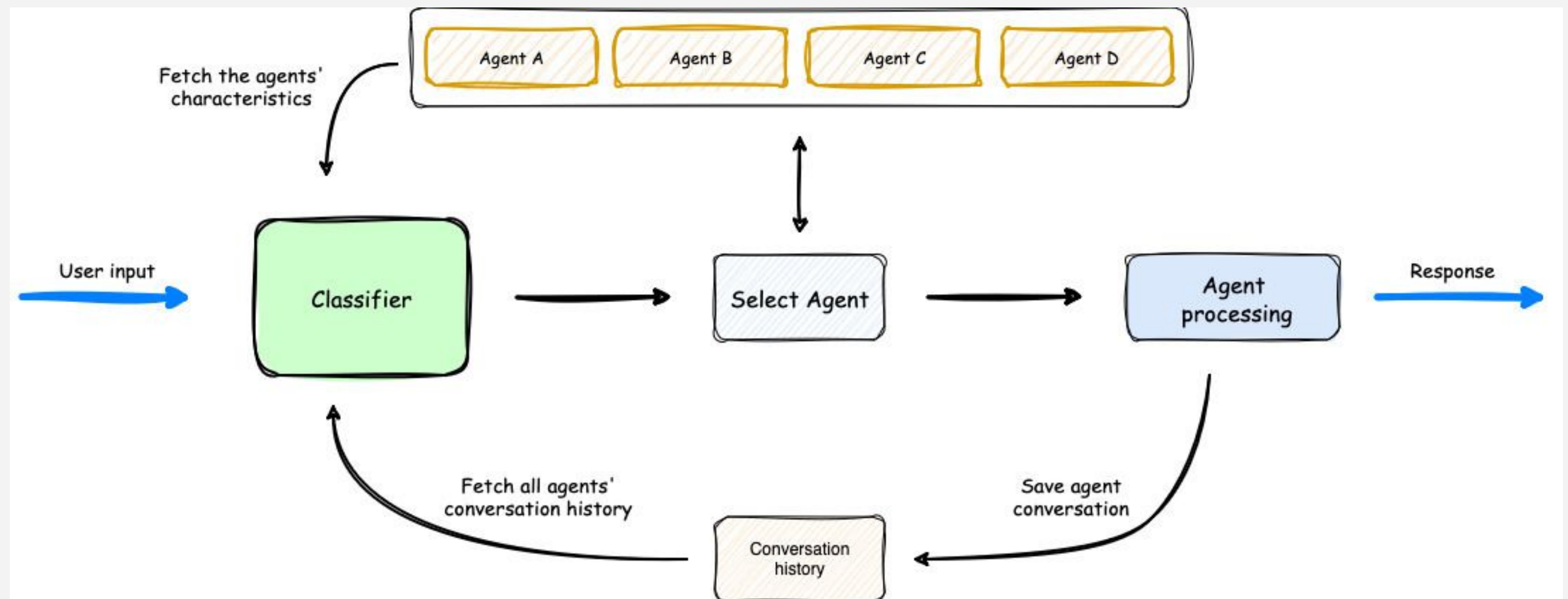


멀티 에이전트 오케스트레이터

n개의 agent중 적합한 agent를 선택하여 처리, 멀티턴 대화에 최적화

- <https://github.com/aws-labs/multi-agent-orchestrator/tree/main>

- <https://nomad-jerry.tistory.com/1>



멀티 에이전트 오케스트레이터

n개의 agent중 적합한 agent를 선택하여
처리, 멀티턴 대화에 최적화
(travel, tech, health agent LLM)

- <https://github.com/aws-labs/multi-agent-orchestrator/tree/main>

- <https://nomad-jerry.tistory.com/1>

대한민국 경주에서 3박 4일 여행일정 만들어줘

네, 경주는 한국의 역사와 문화가 풍부한 도시로 3박 4일 동안 충분히 즐길 수 있는 여행지입니다. 다음은 경주에서의 3박 4일 여행 일정 제안입니다:

1일차:

- 오전: 경주 도착 및 호텔 체크인
- 오후: 불국사 방문 (유네스코 세계문화유산)
- 저녁: 황리단길에서 저녁 식사 및 산책

2일차:

- 오전: 석굴암 방문 (불국사와 함께 유네스코 세계문화유산)
- 오후: 대릉원과 천마총 관람
- 저녁: 동궁과 월지(안압지) 야경 감상

3일차:

- 오전: 경주국립박물관 방문
- 오후: 양동마을 탐방 (유네스코 세계문화유산)
- 저녁: 보문관광단지에서 저녁 식사 및 산책

4일차:

- 오전: 첨성대와 계림 방문
- 오후: 마지막 쇼핑 및 귀가

이 일정은 경주의 주요 명소들을 포함하고 있습니다. 계절이나 개인적인 선호도에 따라 일정을 조정할 수 있습니다. 예를 들어, 봄에는 벚꽃 명소를 추가하거나, 여름에는 보문호수에서의 수상 활동을 고려할 수 있습니다. 또한 경주 한옥마을에서의 숙박이나 전통 한식 체험 등을 추가하여 더욱 풍성한 여행을 만들 수 있습니다.

LangChain vs LangGraph

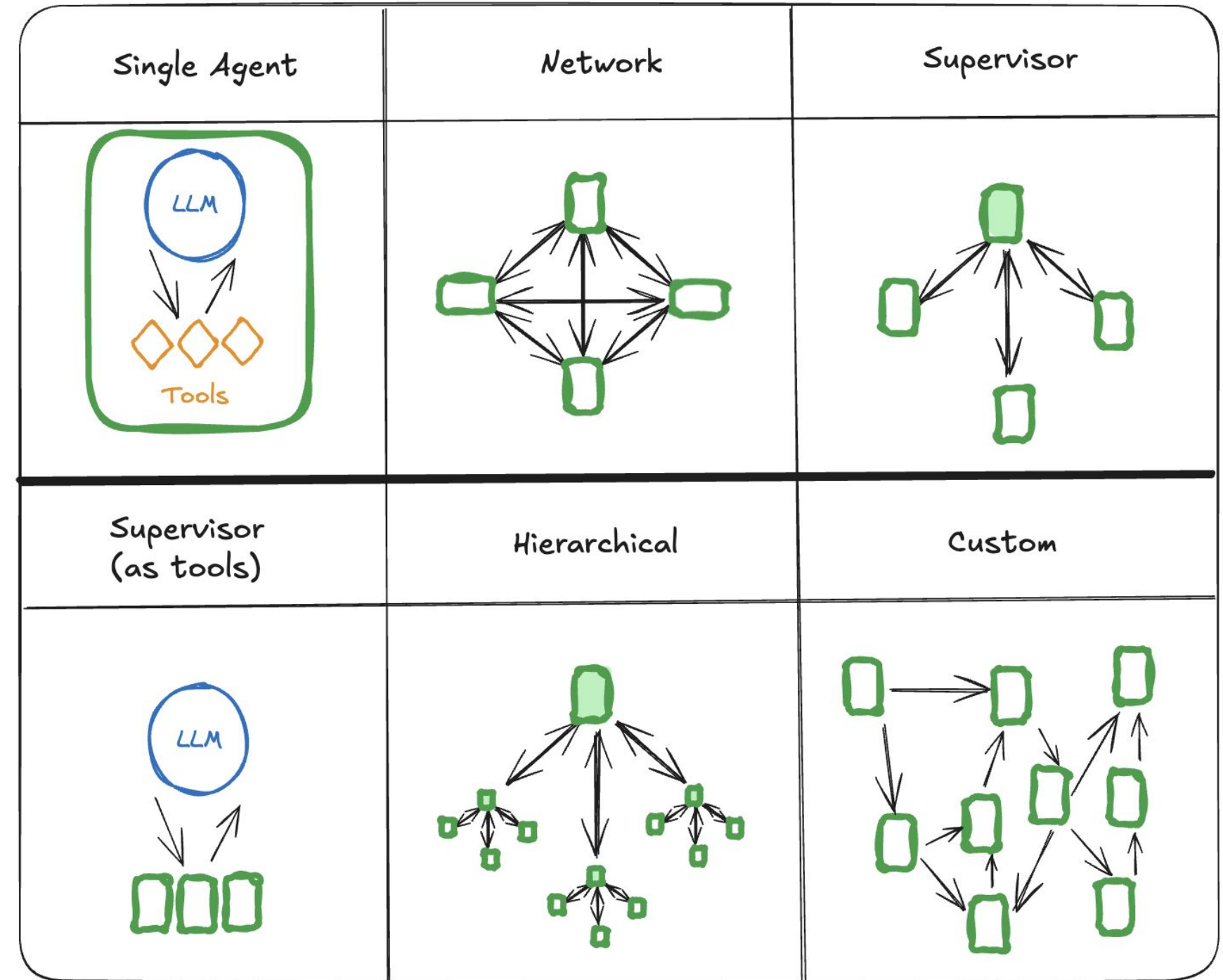
- <https://wikidocs.net/261585>

특징	LangGraph	LangChain
주요 목적	복잡한 워크플로우 및 의사결정 프로세스 구현	LLM 통합 및 체인 구성
구조	그래프 기반	체인 및 에이전트 기반
상태 관리	명시적이고 세밀한 제어	암시적이고 자동화된 관리
유연성	높음 (커스텀 로직 쉽게 구현)	중간 (미리 정의된 컴포넌트 중심)
학습 곡선	상대적으로 가파름	상대적으로 완만한
용도	복잡한 AI 시스템, 다중 에이전트	간단한 LLM 애플리케이션, RAG

멀티에이전트 프레임워크 - LangGraph

library for building stateful, multi-actor applications with LLMs, used to create agent and multi-agent workflows

- <https://github.com/langchain-ai/langgraph>



멀티에이전트 프레임워크 - LangGraph

- **Network**: each agent can communicate with every other agent. Any agent can decide which other agent to call next.
- **Supervisor**: each agent communicates with a single supervisor agent. Supervisor agent makes decisions on which agent should be called next.
- **Supervisor (tool-calling)**: this is a special case of supervisor architecture. Individual agents can be represented as tools. In this case, a supervisor agent uses a tool-calling LLM to decide which of the agent tools to call, as well as the arguments to pass to those agents.
- **Hierarchical**: you can define a multi-agent system with a supervisor of supervisors. This is a generalization of the supervisor architecture and allows for more complex control flows.
- **Custom multi-agent workflow**: each agent communicates with only a subset of agents. Parts of the flow are deterministic, and only some agents can decide which other agents to call next.

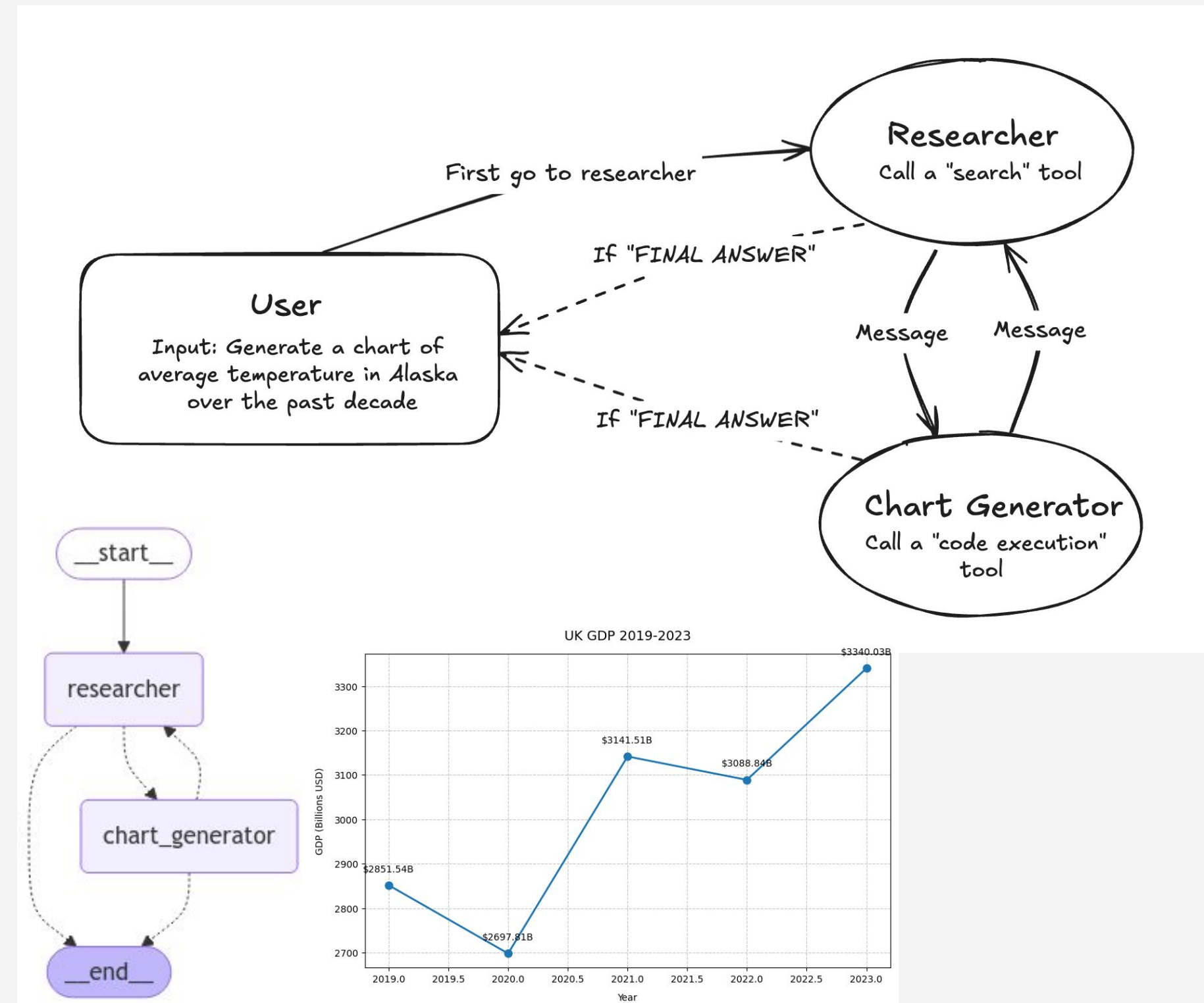
- <https://langchain-ai.github.io/langgraph>

네트워크 - LangGraph

분할 정보 접근 방식 전문가에게 라우팅
agent는 그래프 노드로 정해짐, 계층 구조나
특정 순서가 없는 문제에 적합

- *AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation, by Wu, et. al.*

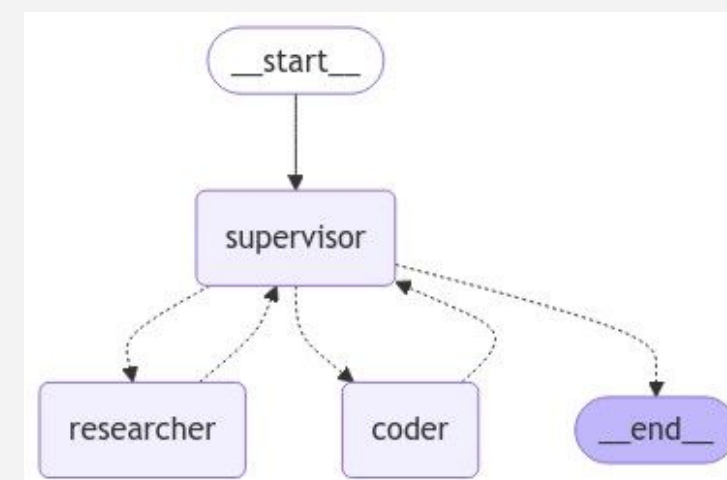
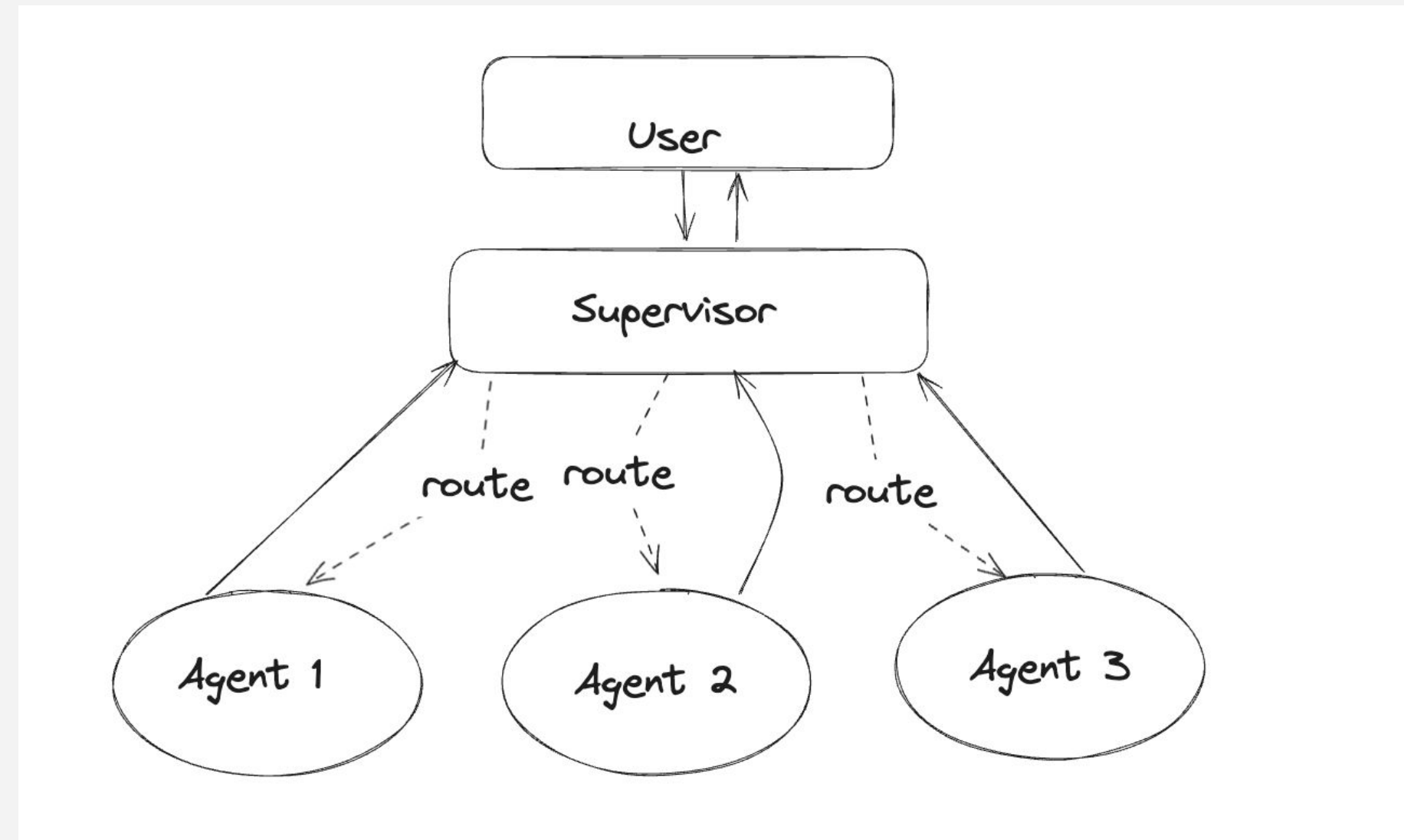
- <https://github.com/langchain-ai/langgraph/blob/main/docs/docs/tutorials/>



슈퍼바이저 - LangGraph

supervisor가 agent group에 task route 에이전트를 노드로 정의하고 슈퍼바이저의 결정에 따라 적절한 에이전트로 라우팅, 병렬 실행

- <https://github.com/langchain-ai/langgraph/blob/main/docs/docs/tutorials/>

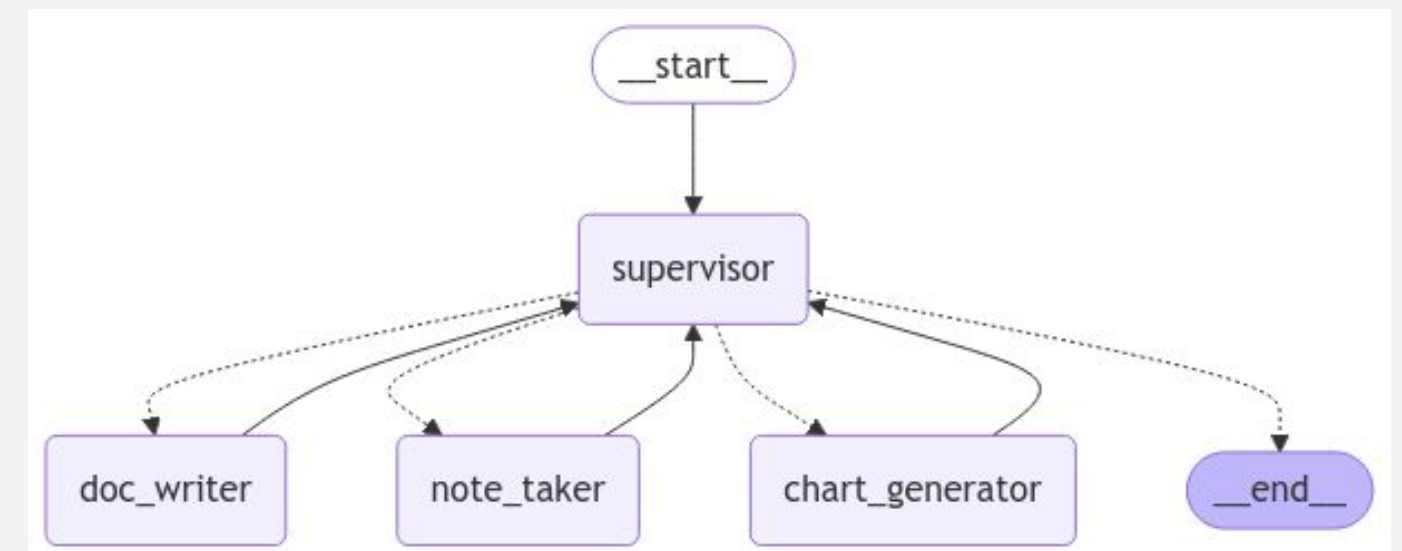
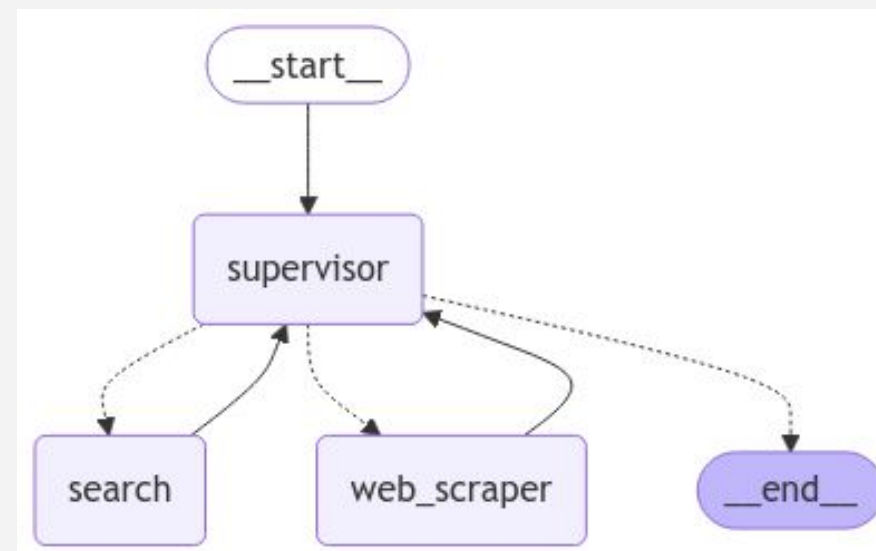
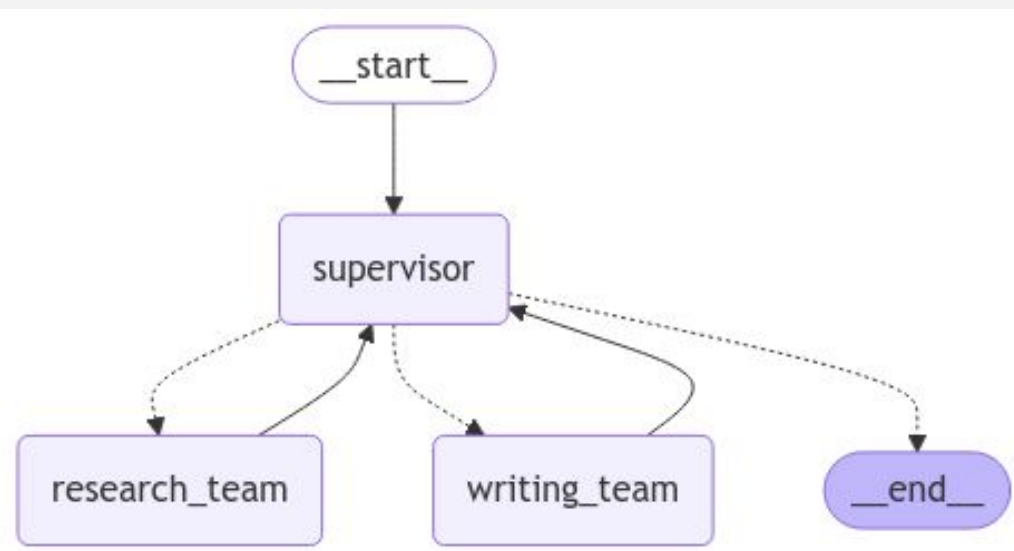
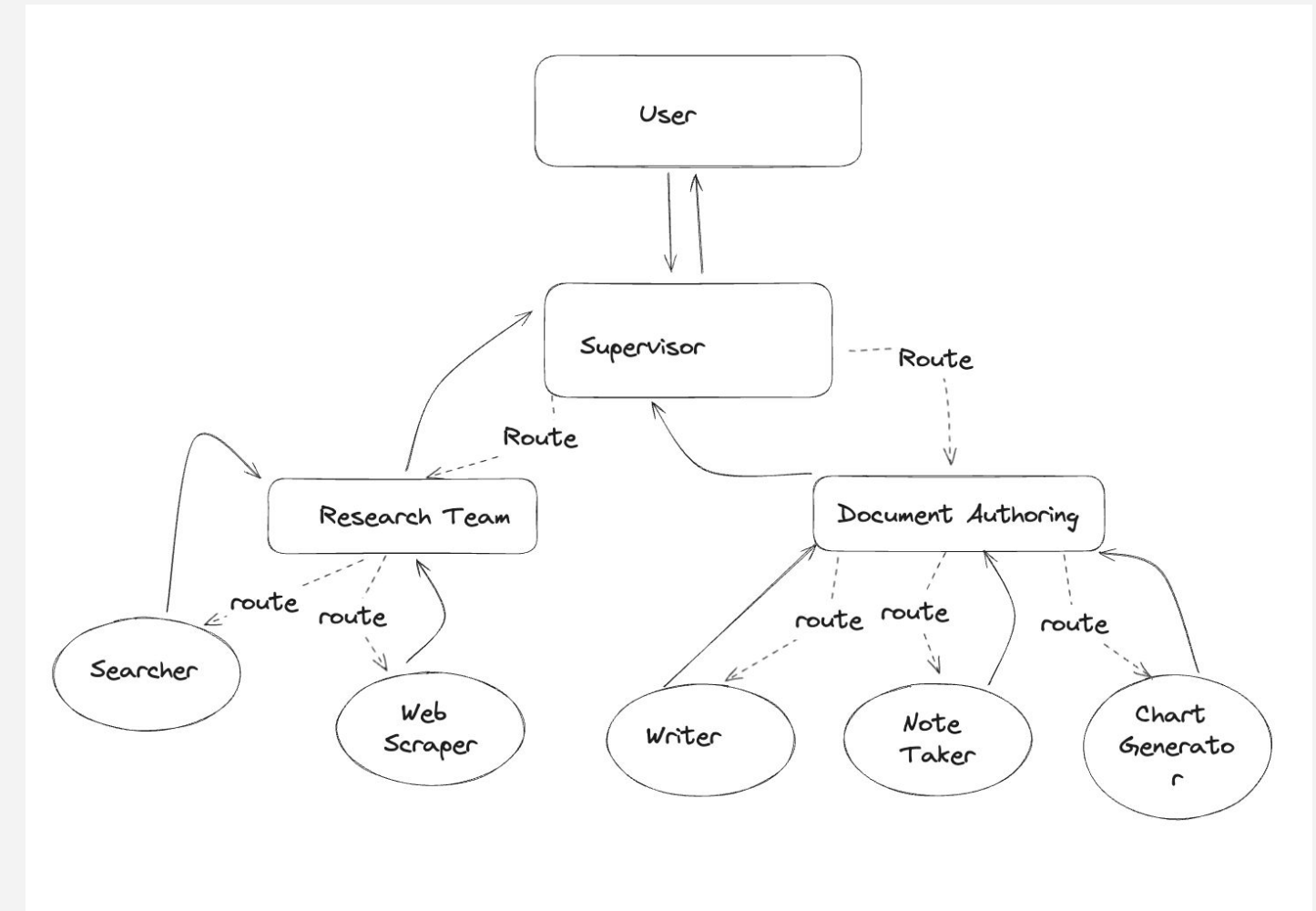


Hierarchical - LangGraph

계층적 시스템 설계, 개별 감독자가 자체 관리하는
별도 에이전트 팀과 슈퍼바이저 분리

- Supervisor가 다수의 워커 노드에 route
작업의 계층적 분산

- <https://github.com/langchain-ai/langgraph/blob/main/docs/docs/tutorials/>



Thank You

[K-DEVCON] Go To Learn 2기 멘티 모집

<https://bluepicture08.notion.site/20-AI-AI-18eecf714ca38071826be2b98074fbc1>



정유선

신세계 AX 데이터 솔루션 담당

jerryjung@apache.org