WHITEPAPER

MSAP.ai

Getting Started AI Agent 개발 — Flowise Basic 1주차 교육자료

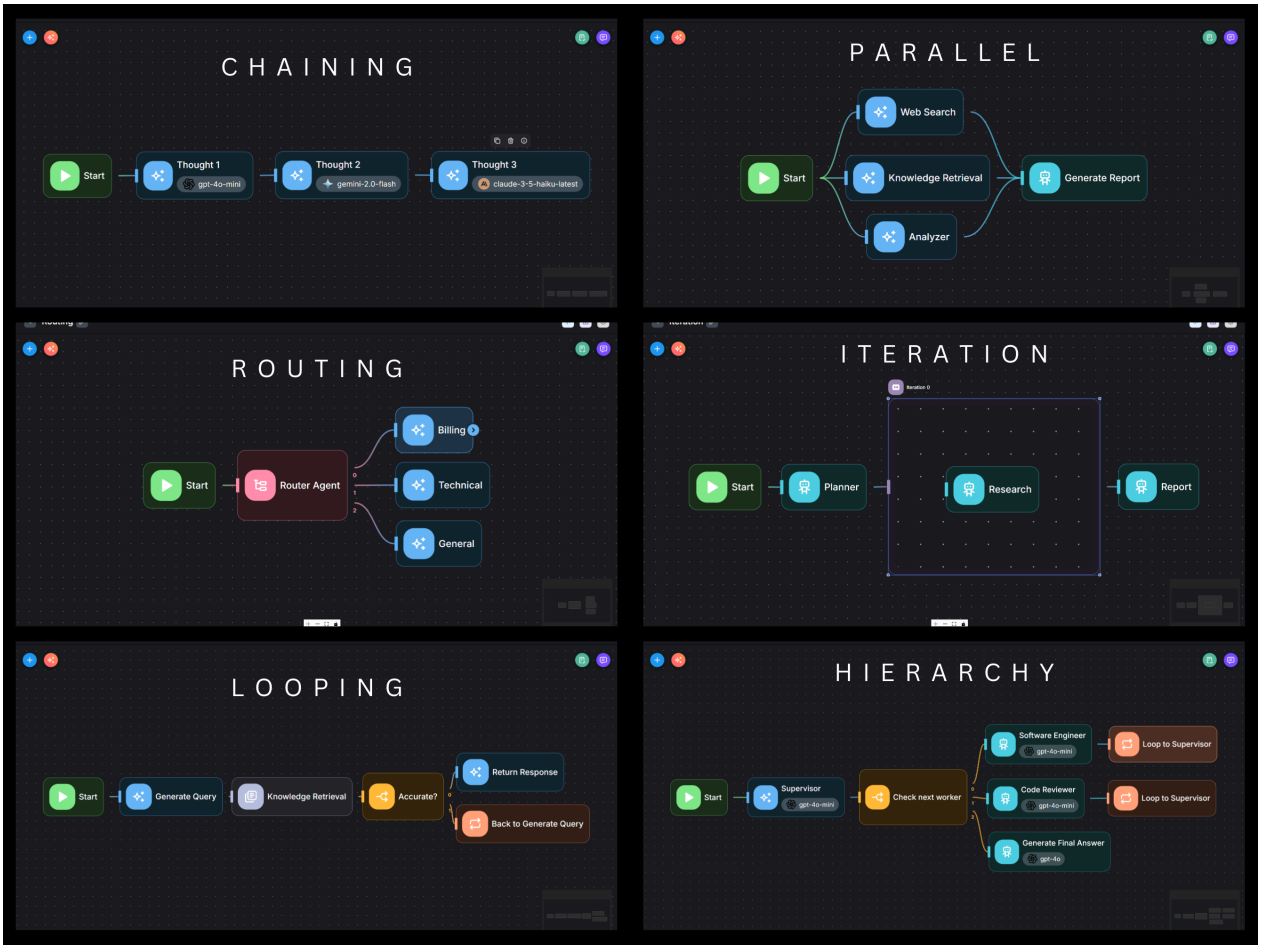
14개 노드 · Flow State · Hello World · Marketplace 5...

처음 새 도구를 만나실 때 가장 답답한 것은 "내가 이걸 다 읽고 나면 무엇을 할 수 있게 되는가" 가 모호하다는 점입니다. 이번 교재는 그 답답함을 1주차 끝에 자가 진단하실 수 있도록, 학습 종료 시점에 답하실 수 있어야 하는 5가지 질문을 미리 약속드리는 것으로 시작합니다.

2026-05-27 · 작성자: SMART CF Team · msap.ai

목차

Getting Started AI Agent 개발 — Flowise Basic 1주차 교육자료



> ***한 권의 약속*** *— 이번 교재는 AI Agent 를 처음 개발하시는 실무 엔지니어 분들을 위한 1주차 입문 교재입니다. 이 교재를 끝까지 읽으시면, Flowise 의 AgentFlow V2 가 등장한 배경, 자동화 플랫폼·코드 프레임워크와의 차이, 14개 표준 노드의 5필드 레퍼런스, Hello World 3노드 워크플로, Marketplace 입문 예제 5종을 직접 손으로 만들어 보실 수 있게 됩니다. 모든 실습은 Docker 1줄로 띄운 Flowise 인스턴스 위에서 동작하며, OpenAI API Key 또는 사내 OpenAI 호환 LLM 1개만 있어도 모든 예제가 작동합니다.*

0장: 학습 목표와 사전 준비

0.1 이 교재가 답하는 5가지 질문

처음 새 도구를 만나실 때 가장 답답한 것은 "내가 이걸 다 읽고 나면 무엇을 할 수 있게 되는가" 가 모호하다는 점입니다. 이번 교재는 그 답답함을 1주차 끝에 자가 진단하실 수 있도록, 학습 종료 시점에 답하실 수 있어야 하는 5가지 질문을 미리 약속드리는 것으로 시작합니다.

0.1.1 5가지 질문 매트릭스

#	질문	본 자료 어느 장에서 답해지는가
Q1	Flowise 는 무엇이고 누가 왜 만들었는가	1장
Q2	Flowise 같은 Workflow 형 도구는 왜 필요한가	2장
Q3	`Start → LLM → Direct Reply` 3노드로 첫 응답을 만들 수 있는가	6장
Q4	14개 표준 노드를 5필드 포맷으로 구별할 수 있는가	7~11장
Q5	Flow State 를 활용해 비인접 노드 간 데이터를 공유하고, Marketplace 예제 5종을 import·변형할 수 있는가	12~14장

본 5개 질문은 마지막 15장의 자가 진단 체크리스트와 1:1 으로 연결됩니다. 학습이 끝나신 시점에 본 표를 다시 펴 보시고, 5개 모두에 자신 있게 답하실 수 있다면 본 자료 1주차 목표는 달성하신 것입니다.

> 💡 ***팁*** *— 본 표를 휴대폰 캡처로 저장해 두시고, 학습 중간중간 자기 위치를 점검하시면 동기 부여에 큰 도움이 됩니다.*

0.1.2 학습 경로 — 빠른 트랙과 깊은 트랙

여러분의 시간 제약에 따라 두 가지 학습 경로를 제안드립니다. 어느 쪽을 선택하셔도 이 교재의 핵심 메시지는 흔들리지 않습니다.

트랙	소요 시간	정독 범위	도달 가능 산출	적합 학습자
빠른 트랙	약 4시간	Ch 0·1·6·14·15	Hello World 3노드 + Marketplace 1예제 변형	시간 부족한 백엔드 엔지니어, 사내 도입 평가 담당
깊은 트랙	약 1주 (하루 1~2시간)	전 장 + 모든 실습	14 노드 5필드 완비 + 5 패턴 + Marketplace 5예제 변형	PoC 책임 엔지니어, 사내 1차 Agent 시스템 설계

직무 한정 시간이 짧으신 경우 빠른 트랙을 권합니다. 빠른 트랙만으로도 Q1·Q2·Q3 세 질문에 답하실 수 있게 됩니다. PoC 책임을 맡으신 분이라면 깊은 트랙이 사내 도입 단계에서 만나실 의문 대부분을 사전 차단해 드립니다.

0.2 사전 준비물 — 브라우저, Flowise 인스턴스, 모델 API 키

이 교재의 모든 실습을 따라가시려면 다음 세 가지가 준비되시면 충분합니다. 본 절에서는 각 항목의 최소 요건만 정리하고, Flowise 인스턴스 설치 절차의 상세는 4장에서 다룹니다.

0.2.1 브라우저와 화면 해상도

Flowise UI 는 React 기반 SPA 입니다. Chrome, Edge, Firefox, Safari 어느 브라우저든 정상 동작합니다. 화면 해상도는 1280×800 이상을 권장드립니다. 그 이하에서는 좌측 노드 팔레트가 중앙 캔버스를 가려서 노드를 드래그·드롭하실 때 학습 곡선이 가팔라집니다.

> 💡 ***팁*** *— 사내 표준 노트북 13인치 1920×1080 으로 충분합니다. 듀얼 모니터 환경이라면 한쪽에 캔버스, 다른 쪽에 본 교재를 띄워 두시면 학습 효율이 두 배입니다.*

0.2.2 Flowise 인스턴스 + 모델 API 키

이 교재의 모든 실습은 Flowise 좌측 사이드바에 `Agentflows` 메뉴가 노출되는 버전 (v2.x 이후, 본 자료 작성 시점은 v3.x 라인) 을 전제합니다. Docker 1줄 (`docker run flowiseai/flowise`) 또는 npm 1줄 (`npx flowise start`) 두 경로 모두 동등하게 지원되며, 자세한 절차는 4장 설치하기 에서 다룹니다.

모델 API 키는 OpenAI API Key 1개만 있어도 이 교재의 모든 실습이 작동합니다. 사내 LLM 환경 (vLLM·Ollama·LM Studio·TGI 같은 OpenAI 호환 서버) 을 사용하시는 경우 `ChatOpenAI Custom` 자격증명 1건을 등록하고 BaseURL 의 끝에 `/v1` 을 정확히 붙이시면 동일하게 동작합니다.

> ⚠ ***자주 막히는 곳*** *— Flowise 첫 화면 좌측 사이드바에 `Chatflows` 만 보이고 `Agentflows` 가 보이지 않는다면 v1 시기 옛 이미지를 띄우신 것입니다. `docker pull flowiseai/flowise:latest` 한 줄로 즉시 해결됩니다.*

여기까지 점검이 끝나셨다면 이 교재의 모든 실습을 진행하실 준비가 끝난 것입니다. 다음 장에서는 Flowise 라는 도구가 어떤 정체를 가졌고 누가 왜 만들었는지부터 풀어보겠습니다.

1장: Flowise 정체 — 왜 만들어졌고 무엇이 다른가

1.1 한 줄 정의와 4 가지 구성 축

새 도구를 빠르게 학습하시는 가장 효율적인 방법은 그 도구의 한 줄 정의를 자기 언어로 발화 가능하게 만드는 것입니다. Flowise 의 한 줄 정의는 다음과 같습니다.

> ***Flowise = FlowiseAI Inc. 가 만든, LangChain.js 위에 GUI 캔버스 한 겹을 얹어 LLM 워크플로(체인·RAG·에이전트·도구·메모리) 를 시각적 노드 그래프로 조립하게 해 주는 오픈소스 도구.***

1.1.1 한 줄 정의의 4 분해

위 한 줄 정의는 네 개의 키워드로 분해됩니다. 각 키워드의 의미를 표 한 장으로 정리하시면, 사내 도입 의사결정 회의에서 그대로 인용 가능한 1줄 정의의 골격이 됩니다.

키워드	의미	학습자가 기억해야 할 한 줄
만든 주체	FlowiseAI Inc. — 오픈소스 프로젝트이자 회사가 책임지고 유지보수	"버려진 프로젝트가 아니라 회사가 책임지는 활성 OSS."
기술 기반	LangChain.js 위에 React 비주얼 캔버스 한 겹	"LangChain 의 표현력 그대로, 진입 장벽만 ↓."
사용 모델	코드 직접 작성이 아닌 노드 드래그·드롭 + 엣지 연결	"시민 개발자도 PoC 가능. 엔지니어도 빠른 프로토타이핑."
라이선스	Apache 2.0	"법무·보안 검토 가장 무난한 라이선스 중 하나."

Apache 2.0 라이선스의 의미를 한 줄 더 풀어드리면, 상업적 사용·수정·배포 모두 자유롭고 사내 도입 시 라이선스 리스크가 거의 없습니다. 사내 PoC 가 운영 단계로 이관될 때 추가 라이선스 협상이 필요 없다는 것이 4분해 중 학습자께서 사내 회의에서 가장 자주 인용하실 문장입니다.

1.1.2 이 교재의 v2 전제와 영문 자료의 v1 분리

이번 교재는 처음부터 끝까지 **v2 AgentFlow 스키마** 를 전제합니다. 그런데 인터넷의 영문 블로그·유튜브 자료의 상당수는 2023~2024 시기의 v1 자료입니다. v1 자료를 따라하시다 보면 노드 이름이 이 교재와 어긋나서 혼란이 발생하기 쉽습니다.

다행히 v1 자료와 v2 자료를 1초 안에 구별하실 수 있는 한 줄 규칙이 있습니다.

> 💡 ***팁 — 노드 이름 끝이 `Agentflow` 로 끝나면 v2, 그렇지 않으면 v1 입니다.***

예를 들어 v2 에서는 `startAgentflow`, `llmAgentflow`, `agentAgentflow`, `directReplyAgentflow`, `conditionAgentflow`, `loopAgentflow`, `humanInputAgentflow` 같은 이름을 사용합니다. v1 에서는 `OpenAI Function Agent`, `LangChain ReAct Agent`, `Buffer Memory` 처럼 자유로운 이름이 쓰였습니다. 검색하실 때는 키워드에 `agentflow v2` 를 명시하시면 v1 자료 혼재를 피하실 수 있습니다.

1.2 누가 · 왜 · 언제 만들었나

1.2.1 FlowiseAI Inc. 와 2023 상반기 GitHub 공개

Flowise 는 2023년 상반기에 FlowiseAI Inc. 가 GitHub 에 공개한 오픈소스 프로젝트로 시작되었습니다. 이름의 유래는 "Flow" + "Wise" 의 합성으로 "흐름을 잘 다루는 도구" 라는 직관적 정체성을 가집니다. 풀어 말하면, LLM 위에 쌓이는 체인·에이전트·RAG·도구 호출 같은 흐름을 코드로 한 줄씩 짜는 방식이 아니라 캔버스 위에 노드와 엣지로 명시적으로 표현하겠다는 선언입니다.

2024~2025 년을 거치며 Flowise 는 GitHub 스타 수가 가파르게 증가해 동일 카테고리의 비주얼 LLM 도구 중 최상위권에 자리 잡았고, 2026년 현재 v3.x 시리즈로 활발히 개발되고 있습니다.

> 💡 ***팁*** — *최신 릴리스를 1번 확인하시려면 Flowise GitHub Releases 페이지를 직접 열어보십시오. 최근 한 달 안에 minor 또는 patch 릴리스가 1건 이상 있으면 활성 프로젝트라는 가장 단순한 신호입니다.*

1.2.2 LangChain.js 표현력 그대로 + GUI 한 겹

Flowise 의 탄생 동기는 LangChain.js 의 강력함과 진입 장벽의 비대칭에서 출발합니다. LangChain.js 는 LLM 패턴을 풍부하게 추상화한 라이브러리이지만, 학습 곡선이 가팔라 비개발자 시민 개발자가 직접 다루기 어렵고, 빠른 PoC 가 필요한 엔지니어조차 의존성 설정·메모리 옵션·도구 등록 코드를 매번 새로 짜는 부담을 안았습니다.

Flowise 는 그 위에 GUI 캔버스 한 겹을 얹어, 동일한 패턴 (Chain·Agent·RAG·Tool·Memory) 을 노드와 엣지로 표현 가능하게 만들었습니다. 즉 "LangChain.js 의 표현력은 그대로, 진입 장벽만 낮춘다" 가 Flowise 가 시장에 던진 가설입니다. 본 가설은 GitHub 스타 추이로 빠르게 검증되었습니다.

LangChain.js 추상화	Flowise V2 대응 노드	의미
Chain	`LLM Node` (단순) / 여러 노드 조립	순차적 LLM 호출
Agent (ReAct)	`Agent Node`	도구 자율 선택
RAG	`Retriever Node` + `LLM Node`	문서 기반 응답
Tool	`Tool Node` 또는 `Agent.Tools`	외부 함수 호출
Memory (Buffer/Window/Summary/Vector)	LLM/Agent 의 `Memory` 옵션	대화 이력 관리
Prompt Template	LLM Node 의 `Messages`	동적 변수 치환

여러분이 LangChain.js 코드로 1주 걸리던 RAG 챗봇을 Flowise 캔버스 위에서는 30분 안에 만드실 수 있는 비교 사례가 본 매핑표의 실용적 결과입니다.

1.3 V1 ↔ V2 분기 규칙 — 7 항목 비교 표

1.3.1 7 항목 비교 표

V1 Chatflow 와 V2 AgentFlow 의 차이를 7개 항목으로 정리한 표가 이 교재에서 가장 자주 인용되는 한 장입니다. 본 표 1개가 학습자께서 v1 자료의 노드를 v2 노드로 잘못 매핑하시는 사고를 사전 차단합니다.

비교 항목	V1 Chatflow	V2 AgentFlow
흐름 제어 주체	LangChain.js Chain/Agent 내부	Flowise 자체 execution queue
노드 이름 규칙	자유로움 (OpenAI Function, LangChain Agent 등)	모두 `*Agentflow` 접미
루프 표현	에이전트 내부에 숨겨짐	`loopAgentflow` / `iterationAgentflow` 노드
조건 분기	별도 노드 없음 (에이전트 추론에 의존)	`conditionAgentflow` / `conditionAgentAgentflow`
Human-in-the-Loop	미지원 또는 외부 패턴 우회	`humanInputAgentflow` 1급 시민

비교 항목	V1 Chatflow	V2 AgentFlow
사이드바 메뉴	`Chatflows`	`Agentflows` (별도 메뉴)
상태 공유	변수/메모리에 한정	`$flow.state` 키-값 저장소 표준

본 표를 휴대폰 캡처로 저장해 두시면 영문 자료에서 만나는 v1 노드를 v2 패턴으로 매핑하실 때 즉시 참조 가능합니다.

1.3.2 정상 환경 점검 — `Agentflows` 메뉴 가시화 한 줄

이 교재의 모든 실습은 좌측 사이드바의 `Agentflows` 메뉴에서 출발합니다. 정상 환경 여부는 다음 한 줄로 점검하실 수 있습니다.

> ***브라우저에서 Flowise 첫 화면을 띄우셨을 때, 좌측 사이드바에 `Agentflows` 와 `Chatflows` 가 별도 항목으로 모두 보이면 v2 정상 지원 환경입니다.***

> ***⚠ 자주 막히는 곳 —*** *`Chatflows` 만 보이고 `Agentflows` 가 보이지 않으면 v1 시기 옛 이미지입니다. `docker pull flowiseai/flowise:latest` 1줄로 해결됩니다.*

본 점검을 통과하셨다면 1장의 학습 목표 (Flowise 한 줄 정의 + v1 ↔ v2 분기 규칙) 를 모두 달성하신 것입니다. 다음 장에서는 한 단계 추상화를 올려, "Flowise 같은 Workflow 형 도구가 왜 필요한가" 를 5가지 이유로 풀어보겠습니다.

2장: 왜 Workflow 형 도구가 필요한가 — 5가지 이유

2.1 5가지 이유 한눈에 — Code ↔ Automation 사이의 골

여러분이 사내에서 AI Agent 도입을 제안하실 때, 가장 자주 받으시는 질문은 "왜 LangChain 코드로 직접 짜지 않고, 왜 n8n 같은 자동화 플랫폼으로 충분하지 않은가" 두 가지입니다. 이번 장에서는 그 두 질문에 한 번에 답하기 위한 5가지 이유를 정리합니다. 본 5가지 이유는 3장의 "AgentFlow V2 와 자동화 플랫폼의 차이 5가지" 와 1:1 짝을 이루며, 두 장을 함께 읽으시면 "왜" 와 "그래서 무엇이 다른가" 의 흐름이 자연스럽게 연결됩니다.

2.1.1 5가지 이유의 한 줄 요약

#	이유	한 줄 요약
1	PoC 와 운영 사이의 골(Gap)	빠른 조립성과 운영의 가시성·재현성을 동시에 잡는 유일한 카테고리
2	Code Framework 한계	가시성·거버넌스·재현성 3축의 한계
3	자동화 플랫폼 한계	LLM 의존 태스크의 5축 복잡도 감당 어려움

#	이유	한 줄 요약
4	Prompt → Workflow 진화 5단계	Flowise = 4단계 (Workflow), 5단계 (Orchestration) 는 후속
5	MCP 1급 통합	벤더 공식 도구를 별도 커넥터 개발 없이 그대로 사용

본 표는 사내 도입 의사결정 회의 슬라이드 1장으로 그대로 인용 가능합니다. 5가지 이유 중 사내 현실과 가장 가까운 1~2가지가 학습자의 도입 정당성 진입점이 됩니다.

2.1.2 5가지 이유와 3장의 5가지 차별점의 1:1 짝

본 장의 5가지 이유는 다음 3장의 "AgentFlow V2 와 자동화 플랫폼의 차이 5가지" 와 정확히 1:1 으로 짝을 이룹니다. 본 장에서 "왜 Workflow 형 도구가 필요한가" 의 일반 논거를 학습하시고, 3장에서 "그래서 V2 가 자동화 플랫폼과 무엇이 다른가" 의 구체적 노드 단위 차이를 학습하시면 됩니다.

> 💡 ***팁** — 본 2장을 정독하신 직후 곧바로 3장으로 점프하시기를 권합니다. 두 장이 연결될 때 학습 효과가 극대화됩니다.*

2.2 이유 1·2 — PoC↔운영 골 + Code Framework 한계

2.2.1 이유 1 — PoC↔운영의 5가지 난제

2025~2026 년을 거치며 기업의 AI Agent 도입은 "PoC 는 쉬운데, 운영은 어렵다" 는 공통 난제에 부딪혔습니다. 처음 몇 줄의 프롬프트와 함수 호출로 PoC 를 만드는 일은 누구나 할 수 있게 되었지만, 그 PoC 를 사내 운영 시스템으로 옮기는 순간 다음 다섯 가지 난제가 한꺼번에 드러납니다.

#	운영 난제	Flowise 의 대응
(a)	흐름이 어떻게 실행되는지 가시화되지 않음	노드 그래프로 흐름 명시화
(b)	어떤 도구가 어떤 입력으로 호출되었는지 추적되지 않음	Agent Node 의 trace 출력
(c)	동일 입력에 대해 결과가 재현되지 않음	노드 단위 설정 JSON 직렬화 + 버전 관리
(d)	사람이 검토·승인할 끼어들기 지점이 없음	`Human Input Node` 1급 시민
(e)	운영 중 모델·도구·프롬프트를 교체하기 어려움	노드 하나 교체로 즉시 대응

여러분의 사내 현실에서 위 5가지 난제 중 몇 가지에 해당하는지 자가 진단해 보십시오. 1~2개라도 해당된다면 Workflow 형 도구의 정당성은 충분히 확보된 것입니다.

2.2.2 이유 2 — Code Framework 의 가시성·거버넌스·재현성 한계

LangChain·LangGraph·CrewAI 같은 코드 기반 프레임워크는 가장 유연하고 표현력이 큰 선택지입니다. 그러나 운영 관점에서 다음 세 가지 한계가 반복적으로 나타납니다.

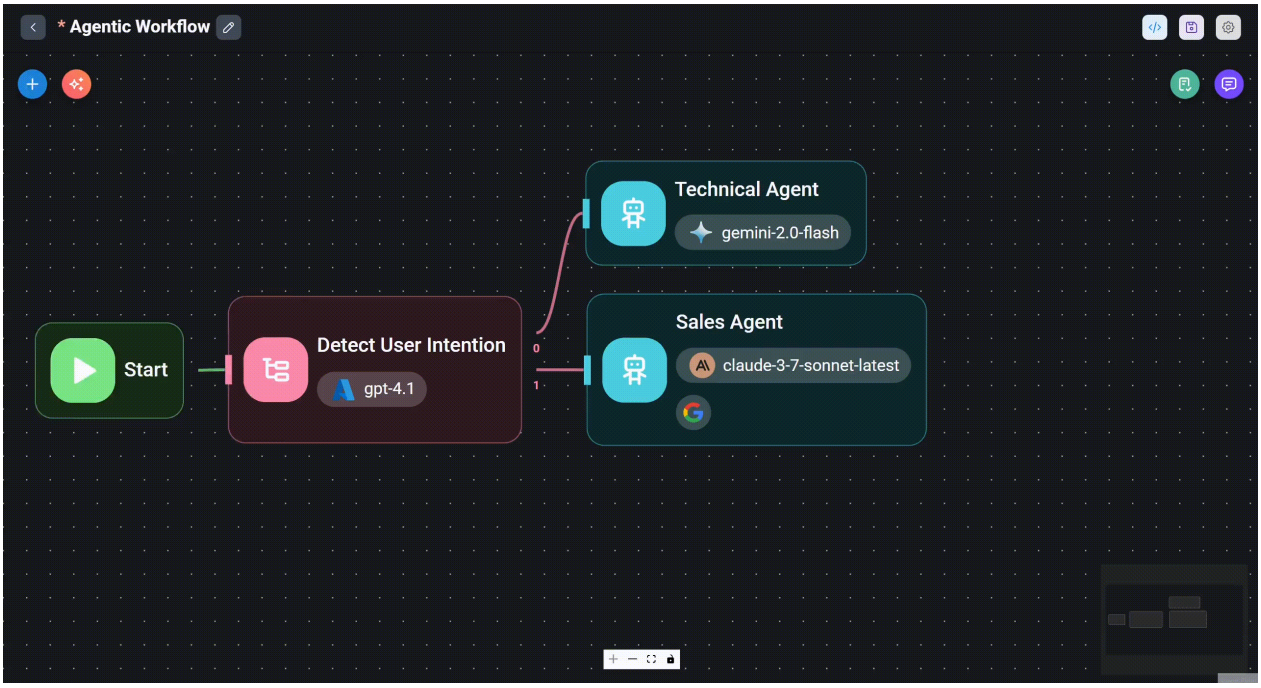
첫째, **가시성** 입니다. 코드 프레임워크의 흐름은 라이브러리 내부 함수 호출과 추론 루프 안에 숨겨져 있어, "지금 이 에이전트가 어떤 도구를 어떤 순서로 호출하고 있는가" 를 운영자가 한눈에 보기 어렵습니다. 디버깅 시점이 되어서야 로그·trace 를 수동으로 따라가야 합니다.

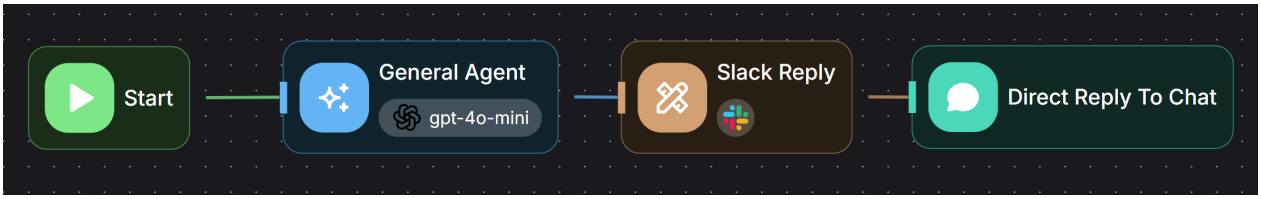
둘째, **거버넌스** 입니다. 누가 어떤 워크플로를 만들었고, 어떤 도구를 등록했으며, 어떤 모델을 사용 중인지, 그리고 변경 이력은 어떻게 되는지 같은 메타데이터가 코드 저장소·CI/CD·로그 시스템에 흩어집니다. 사내 규정 준수(예: 외부 API 호출 화이트리스트, 사용자 데이터 외부 전송 금지) 를 강제하려면 별도 정책 엔진을 코드 위에 얹어야 합니다.

셋째, **재현성** 입니다. 코드는 항상 환경 (라이브러리 버전·환경 변수·모델 응답) 에 영향을 받습니다. 같은 입력이라도 라이브러리 업그레이드 한 번에 결과가 달라질 수 있고, 모델 응답의 비결정성까지 더해지면 "어제 잘 됐는데 오늘 왜 다르지" 같은 운영 의문이 반복됩니다.

> 💡 ***팁*** *— 사내에 LangChain 사용 팀이 있으시다면 본 단락을 그 팀 리드와 1대1 으로 비교해 보십시오. 사내 자가 진단 질문 3개 — "흐름을 한 화면에 볼 수 있는가" / "변경 이력이 한 곳에 모이는가" / "어제 결과를 오늘 똑같이 재현 가능한가" — 가 본 단락의 실용 적용입니다.*

2.3 이유 3·4·5 — 자동화 플랫폼 한계 · 진화 5단계 · MCP





2.3.1 이유 3 — 자동화 플랫폼의 5축 한계

반대편 극단에 있는 노코드 자동화 플랫폼 (n8n·Make·Zapier) 은 가시성과 거버넌스 측면에서 강점이 있지만, LLM 의존 태스크의 복잡도를 감당하기 어렵습니다. 자동화 플랫폼의 노드는 본질적으로 "함수 호출 (입력 → 출력)" 모델을 따르며, 다음 다섯 축에서 한계가 분명합니다.

#	자동화 플랫폼 한계	AgentFlow V2 의 대응 노드
1	에이전트 간 협업이 없음	`Agent Node` 의 자율 위임
2	사람의 끼어들기가 1급 시민이 아님	`Human Input Node` 일시정지 + 영속 체크포인트
3	비인접 노드 간 데이터 공유가 어려움	`$flow.state` 키-값 저장소
4	스트리밍 응답이 1급 시민이 아님	SSE 기반 토큰 스트리밍
5	표준 도구 프로토콜 (MCP) 통합이 약함	MCP 호환 도구의 워크플로 컴포넌트화

사내에 이미 사용 중인 자동화 플랫폼이 있으시고 위 5축 중 일부를 우회 패턴으로 운영 중이시라면, 그 우회 패턴의 운영 비용을 1줄로 정리해 보십시오. 그 비용이 이 교재의 5축 한계 해소가 가져다 줄 실제 가치입니다.

2.3.2 이유 4·5 — 진화 5단계 + MCP 1급 통합

지난 3년간 LLM 응용의 엔지니어링 담론은 다음 다섯 단계의 진화를 거쳤습니다.

시대	핵심 단위	대표 도구·개념
Prompt	프롬프트 한 줄	ChatGPT, Prompt Engineering
Context	프롬프트 + 외부 문맥	RAG, VectorDB, Context Engineering
Agent Framework	에이전트 클래스·체인	LangChain·LangGraph·CrewAI
Agent Workflow	노드 그래프 (흐름)	**Flowise** · Langflow · Dify
Agent Orchestration	워크플로 위의 컨트롤 플레인	워크플로 매니저 · MCP 서버군 · 정책 엔진

Flowise 는 정확히 4단계 (Agent Workflow) 에 위치합니다. 1~3단계의 한계 (가시성 부족·운영 어려움) 와 5단계의 복잡성 (전사 거버넌스 필요) 사이에서 "PoC 와 운영을 동일 캔버스에서 잇는다" 는 위치입니다. 5단계는 이 교재의 범위를 벗어나며, 후속 자매 자료에서 다룹니다.

마지막 다섯 번째 이유인 **MCP (Model Context Protocol) 1급 통합** 은 Anthropic 이 제안한 "LLM 이 외부 도구·데이터 소스에 표준화된 방식으로 접근하는" 프로토콜입니다. 자동화 플랫폼은 외부 도구마다 별도 커넥터를 개발해야 하지만, MCP 호환 도구는 한 번 등록하면 어떤 MCP 호환 LLM 클라이언트 (Claude Desktop · Cursor · Flowise 등) 에서도 동일한 인터페이스로 호출됩니다. Flowise V2 는 MCP 도구를 단순 "Agent 보조 도구" 가 아니라 **워크플로 컴포넌트 자체** 로 취급하여, 캔버스 위의 노드처럼 끼워 넣으실 수 있습니다.

> 💡 **팁** — *GitHub MCP 는 GitHub 팀이 직접 만들고 유지보수합니다. Atlassian Jira · Brave Search 등 다른 대형 서비스도 공식 MCP 를 제공합니다. 즉 별도 커넥터 개발 없이 "벤더 공식 도구" 를 그대로 끌어 쓰실 수 있다는 것이 본 이유의 실용적 결과입니다.*

여기까지로 2장의 5가지 이유를 모두 정리했습니다. 다음 3장에서는 같은 5축을 "AgentFlow V2 노드가 자동화 플랫폼과 어떻게 다른가" 의 시각으로 1번 더 봅니다. 두 장을 연결해 읽으시면 이 교재의 첫 의문 (왜 Flowise 가 필요한가) 은 완전히 해소됩니다.

3장: AgentFlow V2 핵심 개념 — execution queue + Flow State 첫 등장

3.1 한 줄 정의와 3개 키워드 분해

AgentFlow V2 의 한 줄 정의는 1장에서 이미 미리 보셨습니다. 본 절에서는 그 한 줄을 3개 키워드로 분해하여, 14 노드 학습의 일관 기준을 마련합니다.

> ***AgentFlow V2 = "외부 프레임워크에 의존하지 않고, Flowise 가 자체적으로 만든 노드로 LLM 워크플로를 명시적(explicit) 으로 조립하는" 새로운 워크플로 모델.***

3.1.1 3개 키워드 분해

키워드	의미	실용적 결과
외부 의존 최소화	LangChain·LangGraph 같은 외부 라이브러리에 위임하던 흐름 제어를 Flowise 자체 execution queue 로 가져옴	워크플로 동작이 LangChain 내부 상태가 아닌 Flowise 노드 그래프로 결정
명시적 조립	V1 의 "에이전트 내부 마법" (ReAct 추론·도구 선택·메모리 압축) 을 노드 단위로 쪼개 캔버스 위에 드러냄	루프는 `Loop Node` , 조건은 `Condition Node` , HITL 은 `Human Input Node` 처럼 흐름 제어가 모두 가시화
노드 = 독립 실행 단위	모든 노드는 자기 입력만 받으면 자기 일만 처리하고 정해진 출력 슬롯에 결과를 떨굼	노드 하나만 교체하면 동일 워크플로를 다른 LLM·도구·API 로 갈아 끼움

본 3개 키워드를 자기 언어로 발화 가능하시게 되시면, 4~11장의 14 노드 학습 시 노드마다 "왜 이 노드가 존재하는가" 의 정당성을 자가 점검하실 수 있게 됩니다.

3.1.2 V1 vs V2 비교 표 재확인 + 사이드바 메뉴 캡처

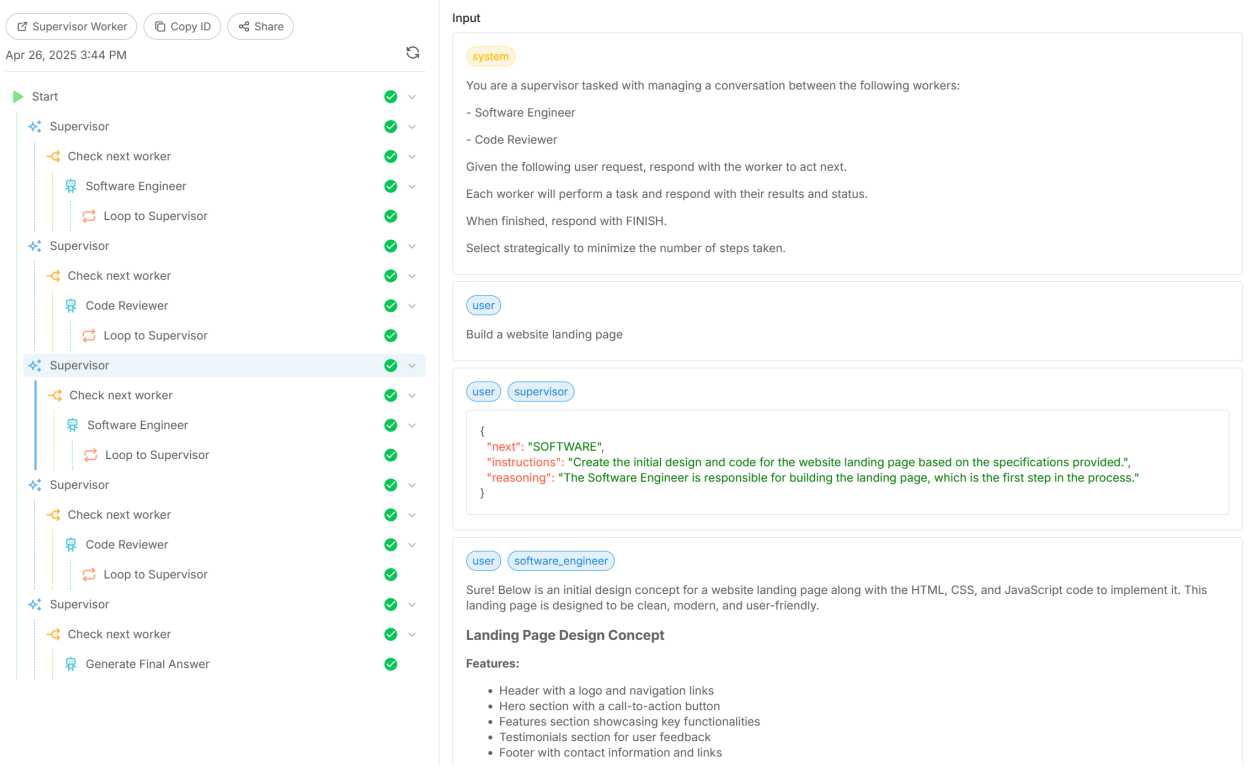
1장 1.3.1 의 7행 비교 표를 본 위치에서 1번 더 노출합니다. 두 번째로 보시는 시점에 학습자께서 7행 모두를 자기 언어로 설명 가능한지 자가 검증해 보십시오.

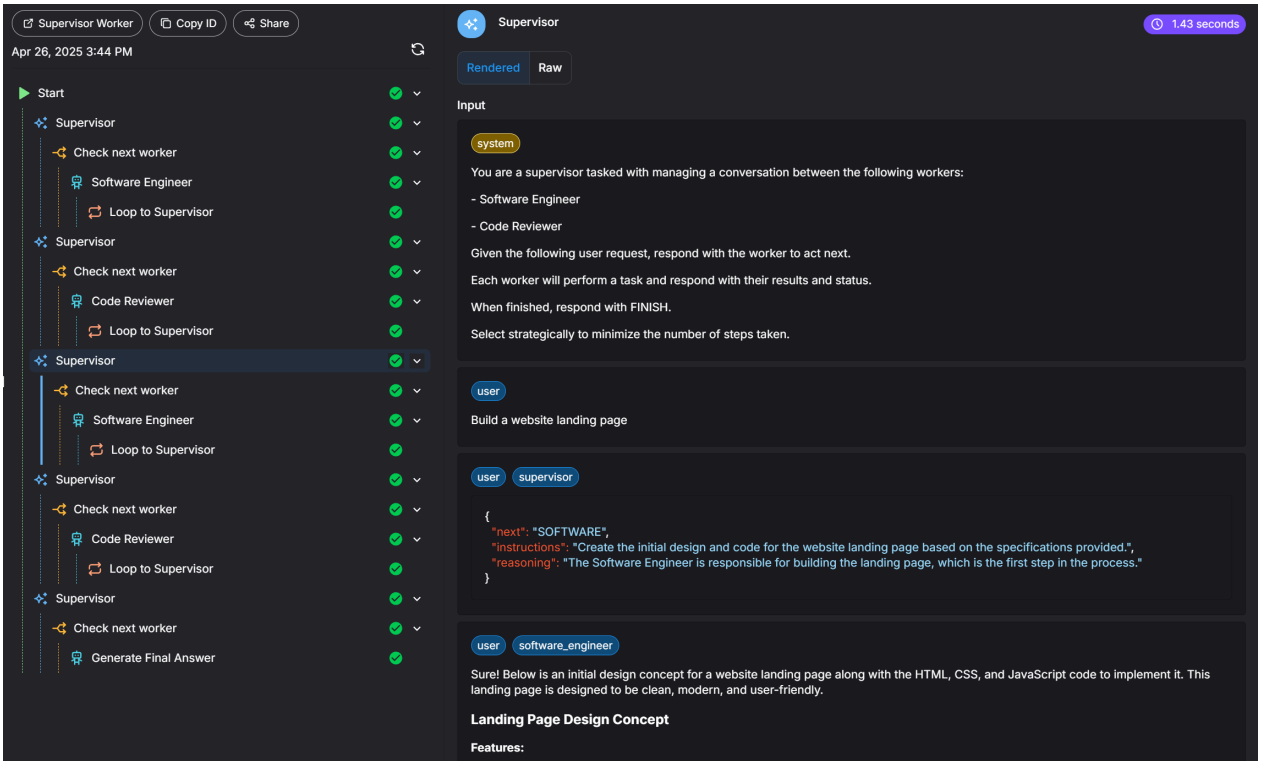
비교 항목	V1 Chatflow	V2 AgentFlow
흐름 제어 주체	LangChain.js Chain/Agent 내부	Flowise 자체 execution queue
노드 이름 규칙	자유로움	모두 `*Agentflow` 접미
루프 표현	에이전트 내부에 숨겨짐	`loopAgentflow` / `iterationAgentflow`
조건 분기	별도 노드 없음	`conditionAgentflow` / `conditionAgentAgentflow`
Human-in-the-Loop	미지원 또는 외부 패턴 우회	`humanInputAgentflow` 1급 시민
사이드바 메뉴	`Chatflows`	`Agentflows` (별도 메뉴)
상태 공유	변수/메모리 한정	`$flow.state` 키-값 저장소

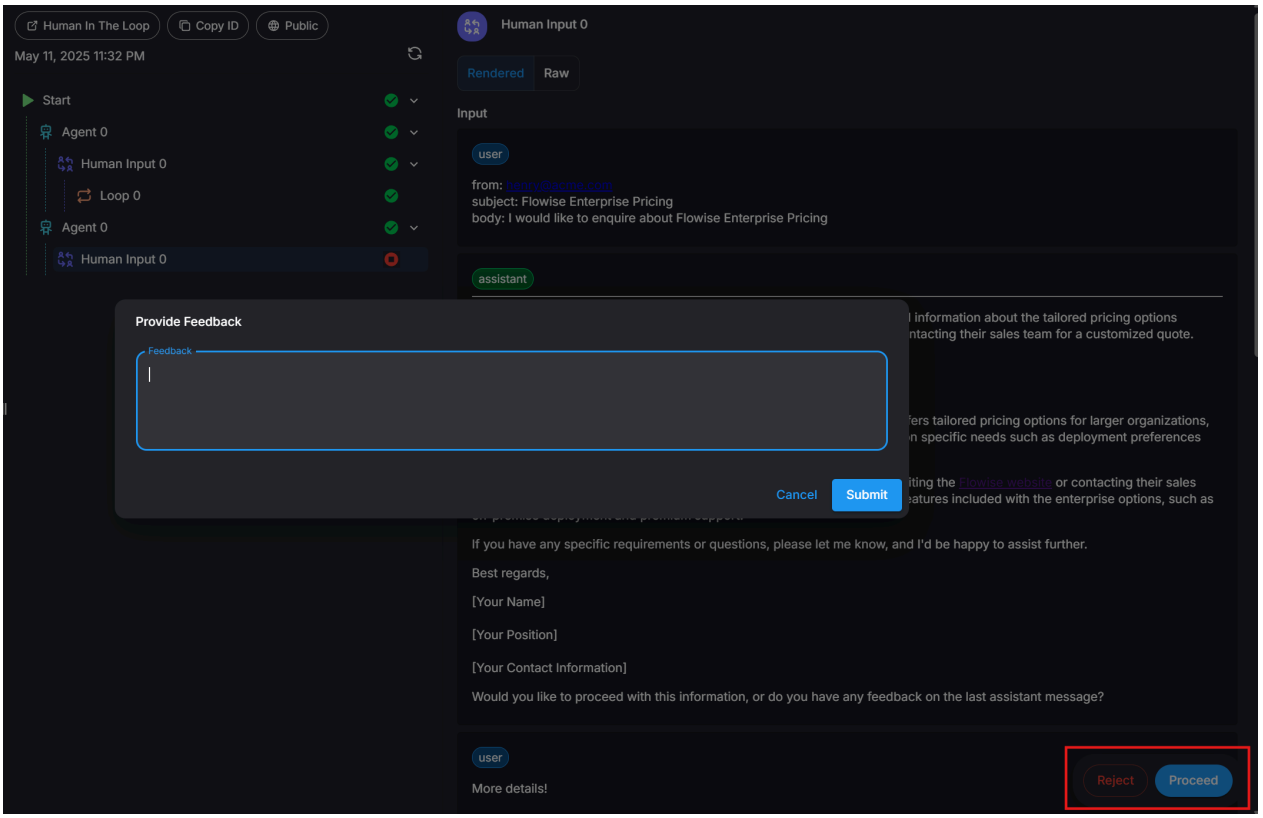
본 표를 두 번째 보신 시점에 7행 모두를 설명 가능하시다면 1·2장 학습 흡수율이 매우 높으신 것입니다.

3.2 자동화 플랫폼과의 5가지 차이 — 캡처와 함께

본 절은 2장의 5가지 이유를 3장에서 다시 "구체적 노드 단위 차이" 의 시각으로 봅니다. 1차 자료의 캡처를 함께 보시면 시각적으로 흡수됩니다.







3.2.1 차이 1·2·3 — Agent-to-Agent · HITL · Shared State

자동화 플랫폼의 노드는 "함수 호출" 입니다. 입력을 받아 출력을 낸다, 그게 전부입니다. 그러나 Flowise 의 `Agent Node` 는 자율적인 LLM 에이전트이고, 한 에이전트가 다른 에이전트에게 작업을 위임할 수 있습니다. 위임받은 워커 에이전트는 슈퍼바이저 에이전트의 전체 대화 이력을 컨텍스트로 들고 작업하기 때문에, 단순한 함수 호출과 달리 "맥락을 공유한 협업" 이 가능합니다.

현실 예시로, 고객 문의 분류 에이전트 (슈퍼바이저) 가 들어오는 문의를 읽고 "환불 문의는 환불팀 에이전트, 기술 지원은 기술팀 에이전트에게 위임" 하는 패턴이 가장 흔합니다. 환불팀 에이전트는 분류 에이전트가 본 원문 메시지·고객 ID·과거 대화를 모두 함께 받기 때문에, "다시 처음부터 설명해 주세요" 같은 사용자 짜증을 유발하지 않고 바로 작업을 이어갈 수 있습니다.

두 번째 차이인 **HITL (Human-in-the-Loop)** 은 워크플로 자체를 일시정지할 수 있다는 점에서 자동화 플랫폼과 결정적으로 다릅니다. 사람이 승인 (approve) 하면 `proceed` 경로로 진행되고, 반려 (reject) 하면 `reject` 경로로 분기합니다. 더 중요한 것은 Flowise 애플리케이션이 재시작되어도 일시정지 상태가 유지된다는 점입니다 (체크포인트 메커니즘). 마케팅 매니저가 휴가 중이면 워크플로가 며칠이고 일시정지 상태로 대기하고, 매니저가 돌아와 승인 버튼을 누르면 자동으로 재개됩니다.

세 번째 차이인 **Shared State (`$flow.state`)** 는 워크플로 실행 단위의 키-값 저장소를 제공합니다. 자동화 플랫폼의 데이터 흐름은 본질적으로 선형이라 5단계 전 값을 8단계에서 다시 쓰려면 매 단계마다 끌고 가야 하지만, Flow State 가 있으면 `{{ $flow.state.keyName }}` 한 줄로 어느 노드에서든 읽으실 수 있습니다.

3.2.2 차이 4·5 — Streaming · MCP 1급 통합

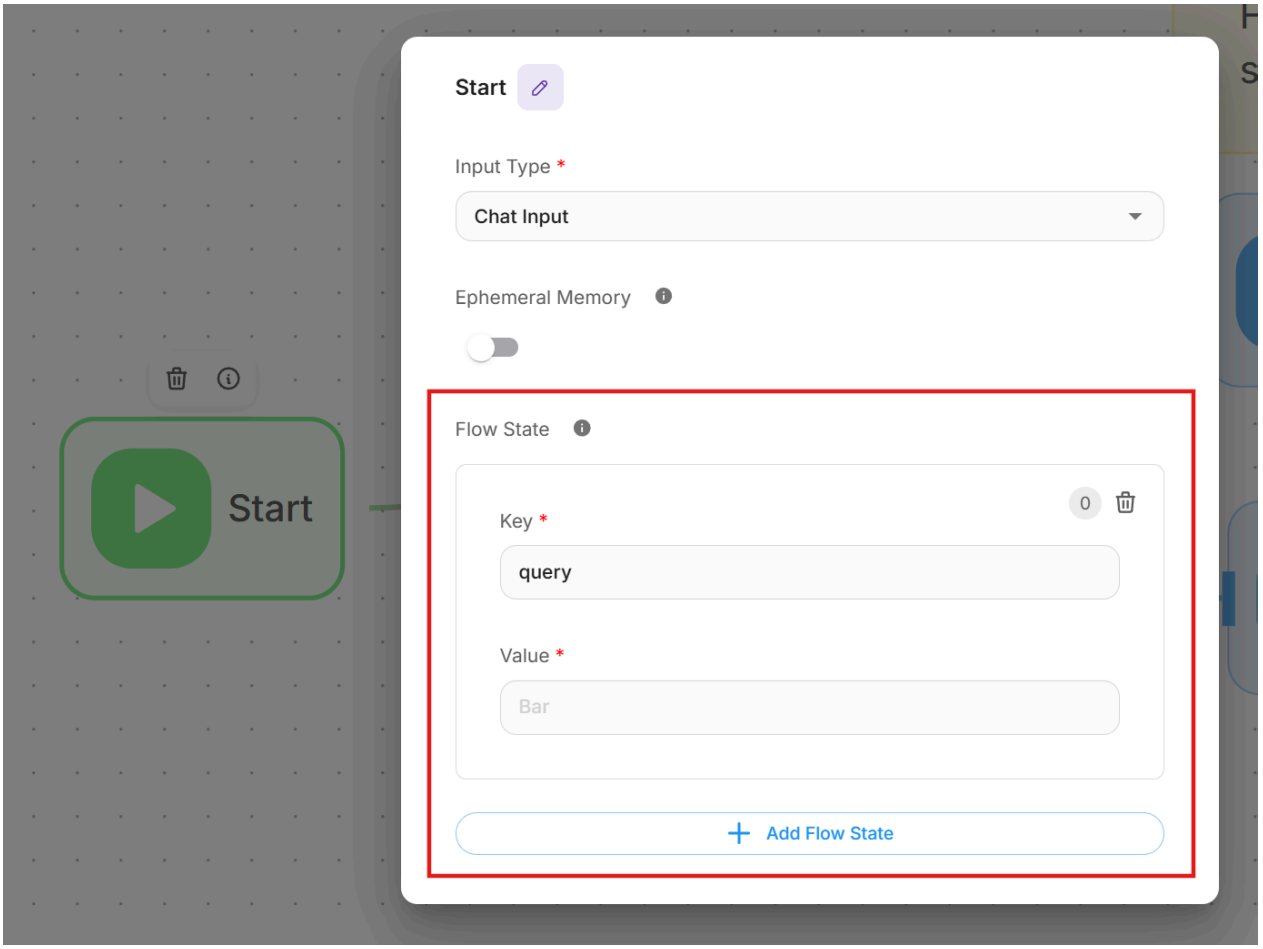
네 번째 차이는 **Streaming** 입니다. 자동화 플랫폼은 워크플로 실행 결과를 "완료 후 한 번에" 반환하지만, Flowise V2 는 Server-Sent Events (SSE) 기반 스트리밍을 지원해 LLM 이 토큰을 만들어내는 즉시 클라이언트로 흘려보냅니다. 사용자 입장에서는 ChatGPT 처럼 "타이핑되는" 응답을 받으실 수 있고, 체감 응답 속도가 비-스트리밍 대비 1/3~1/5 수준으로 줄어듭니다.

> 💡 ***팁*** *— SSE 는 HTTP 응답 채널 위에서 서버가 클라이언트로 단방향 이벤트를 push 하는 표준입니다. WebSocket 보다 단순하고 방화벽 친화적입니다.*

다섯 번째 차이인 **MCP 1급 통합** 은 2.3.2 에서 이미 풀었습니다. Flowise V2 의 MCP 도구 등록 실제 단계는 9장 (Tool · Retriever · HTTP) 에서 손으로 따라 만드실 수 있습니다.

3.3 Flow State 첫 등장 — 미리보기

3.3.1 Flow State 한 줄 정의 + 비유



`$flow.state` 는 V2 워크플로 내부에서 노드들이 공유하는 키-값 저장소입니다. 자동화 플랫폼에는 없는 V2 만의 핵심 기능이라 별도 장으로 다뤄야 하지만, 본 절에서는 한 줄 정의와 비유 1개만 미리 보여드립니다.

> ***Flow State = "워크플로 한 번의 실행 동안만 유효한, 모든 노드가 읽고 쓸 수 있는 키-값 저장소."***

비유하자면 한 회의에서 화이트보드에 적은 임시 메모입니다. 회의가 끝나면 지워지지만, 회의 도중에는 누구나 보고 고칠 수 있습니다. 영구 사용자 프로필처럼 세션을 넘어 살아남아야 하는 데이터는 Redis · PostgreSQL 같은 외부 저장소에 따로 적재하셔야 합니다.

> 💡 ***팁*** *— 상세는 12장에서 다룹니다. 본 절에서는 "한 실행의 화이트보드" 비유 한 가지만 기억하시면 충분합니다.*

3.3.2 Flow State 갱신 가능 노드 7개 미리보기

Flow State 의 값을 갱신할 수 있는 노드는 14개 중 다음 7개입니다.

갱신 가능 노드 7개	갱신 불가 노드 7개
LLM, Agent, Tool, HTTP, Retriever, Custom Function, Execute Flow	Start, Direct Reply, Condition, Condition Agent, Iteration, Loop, Human Input

Start Node 는 갱신이 아닌 **선언** 의 위치이고, Direct Reply Node 는 응답 전송 후 분기 종료라 갱신이 의미 없습니다. Condition · Condition Agent · Iteration · Loop · Human Input 5개는 흐름 제어 노드라 상태 변경 책임을 별도 운영 노드에 넘기는 설계입니다. 본 표를 기억하시면 12장의 함정 1번 (선언 안 한 키 사용) 의 사전 면역이 됩니다.

4장: Flowise 설치 — Docker 1줄 / npm 1줄

4.1 사전 요구사항 점검

이번 장에서는 이 교재의 첫 실습 단원입니다. Docker 1줄 또는 npm 1줄로 Flowise 인스턴스를 띄우고 브라우저에서 첫 화면을 확인하는 것이 본 장의 목표입니다.

4.1.1 Docker 또는 Node.js 버전 점검

> ***objective*** *— 로컬 환경에 Docker 또는 Node.js 둘 중 하나가 갖춰져 있는지 1줄 명령으로 확인합니다.*

```
docker --version
```

```
Docker version 24.0.7, build afdd53b
```

또는

```
node --version
```

```
v20.11.0
```

Docker 가 ≥ 20.10 또는 Node.js 가 ≥ 18.x 통과하시면 본 장의 실습 진입 가능합니다. 두 버전 모두 미달이시면 Docker Desktop 또는 Node.js LTS 를 먼저 설치해 주십시오.

> 💡 ***팁*** *— Windows 학습자께서는 WSL2 환경 안의 Docker 를 권장드립니다. WSL2 안에서* `docker --version` *이 응답하면 정상입니다.*

4.1.2 포트 3000 충돌 사전 점검

> ***prerequisite*** *— Flowise 기본 포트 3000 이 다른 프로세스에 점유되어 있지 않은지 확인합니다.*

```
# Mac / Linux
lsof -i :3000
```

```
# Windows (PowerShell)
netstat -ano | findstr :3000
```

위 명령에 응답이 없으면 (`lsof` 가 아무것도 출력 안 함, `netstat` 도 빈 줄) 포트 3000 이 비어 있는 것입니다. 응답이 1줄 이상 나오면 React 개발 서버 · Grafana · 다른 Docker 컨테이너 등이 점유 중이라는 신호이므로, 4.3.1 의 충돌 회피 (호스트 포트만 4000 으로 변경) 를 미리 적용하시면 됩니다.

> ⚠ ***자주 막히는 곳*** *— 사내 노트북에서 흔히 점유 중인 후보는 React 개발 서버 (* `npm start` *) · Grafana · 다른 Docker 컨테이너입니다. 충돌 시 4000 변경이 가장 안전합니다.*

4.2 Docker 1줄 설치 + npm 1줄 보조

4.2.1 Docker 1줄 설치

> ***steps*** *— Docker 1줄 명령으로 Flowise 컨테이너를 띄우고 브라우저에서 첫 화면을 확인합니다.*

```
docker run -d \
 -p 3000:3000 \
 --name flowise \
 -v ~/.flowise:/root/.flowise \
 flowiseai/flowise
```

```
a1b2c3d4e5f6... (컨테이너 ID — 64자 hex)
```

명령 옵션을 한 줄씩 풀어드리면 다음과 같습니다. `-d` 는 백그라운드 실행, `-p 3000:3000` 은 호스트 포트:컨테이너 포트 매핑, `--name flowise` 는 컨테이너 이름 지정 (이후 `docker stop flowise` / `docker start flowise` 로 제어 가능), `-v ~/.flowise:/root/.flowise` 는 데이터 영속 볼륨 마운트입니다. 마지막 인자가 이미지 이름입니다.

브라우저에서 `http://localhost:3000` 으로 접속하시면 Flowise 첫 화면이 표시됩니다.

> ✅ ***체크포인트*** *— 좌측 사이드바에* `Chatflows` *와* `Agentflows` *두 메뉴가 모두 보이면 v2 정상 환경입니다.*

4.2.2 npm 1줄 보조 + 첫 화면 확인

> ***steps (대안)*** *— 사내 보안 정책상 Docker 가 막힌 환경에서는 npm 으로 직접 실행하시면 됩니다.*

```
npx flowise start --PORT=3000
```

```
Flowise Server is listening at http://0.0.0.0:3000
```

npm 경로는 Node.js 18+ 가 필요합니다. 사내 Node.js 버전 매니저 (nvm) 를 사용 중이시라면 `nvm use 20` 으로 LTS 버전을 활성화하신 뒤 위 명령을 실행하십시오.

> ✅ ***체크포인트*** *— 두 경로 모두 브라우저 첫 화면에서 좌측 사이드바의 `Agentflows` 메뉴 가시화로 정상 확인됩니다.*

4.3 자주 막히는 곳 — 3000 충돌 · WSL2 · M1 호환

4.3.1 3000 포트 충돌 회피 — `-p 4000:3000`

> ***troubleshooting*** *— 포트 3000 이 점유 중이면 호스트 포트만 4000 으로 변경합니다.*

```
docker run -d -p 4000:3000 --name flowise -v ~/.flowise:/root/.flowise flowiseai/flowise
```

> ⚠ ***좌:우 의미*** *— `-p 4000:3000` 의 왼쪽 `4000` 이 호스트 포트, 오른쪽 `3000` 이 컨테이너 내부 포트입니다. 컨테이너 내부는 항상 3000 으로 두시고 호스트 포트만 바꾸십시오.*

변경 후 브라우저 URL 도 `http://localhost:4000` 으로 접속하셔야 합니다.

4.3.2 WSL2 · M1 호환 1줄 메모

Windows WSL2 환경에서는 WSL2 안의 Docker 실행을 권장드립니다. Apple Silicon M1/M2 학습자께서는 첫 실행 시 logs 를 확인하시고, 도구 호환성 문제가 발생하면 `--platform linux/amd64` 옵션을 시도해 보십시오.

```
# M1/M2 호환 옵션
docker run -d --platform linux/amd64 -p 3000:3000 --name flowise -v ~/.flowise:/root/.flowise flowiseai/flowise
```

2026 현재 Flowise 이미지는 arm64 도 지원하지만, marketplace 의 일부 도구가 amd64 만 제공하는 경우가 있습니다. 그런 경우 위 옵션이 우회 경로가 됩니다.

여기까지로 4장 설치 학습이 끝났습니다. 다음 장에서는 모델 자격증명을 등록하여 6장 Hello World 실습의 사전 전제를 마무리합니다.

5장: 모델 자격증명 등록 — OpenAI · Custom LLM

5.1 Credentials 메뉴 진입과 OpenAI API 등록

본 장의 목표는 좌측 사이드바의 `Credentials` 메뉴에서 모델 자격증명 2종 (OpenAI API · 사내 Custom LLM) 을 등록하는 것입니다. 본 장을 끝까지 따라 하시면 6장 Hello World 의 모델 선택 단계에서 막히지 않습니다.

5.1.1 좌측 사이드바 → Credentials → Add Credential

> ***steps*** *— Credentials 메뉴 진입 + OpenAI API 자격증명 검색.*

좌측 사이드바의 `Credentials` 를 클릭하시면 자격증명 목록 화면으로 진입합니다. 우측 상단의 `Add Credentia l` 버튼을 클릭하시면 자격증명 카탈로그 다이얼로그가 열립니다. 검색창에 `OpenAI` 를 입력하시면 여러 카드가 노출되며, 그 중 `OpenAI API` 카드를 선택하십시오.

다이얼로그에서 입력하실 필드는 두 개입니다.

필드	값 예시	설명
Name	`openai-default`	자격증명 식별자. 이후 모델 선택 시 드롭다운에서 참조
OpenAI API Key	`sk-...`	OpenAI 계정의 API 키 (sk- 로 시작)

> ✅ ***체크포인트*** *— `Add Credential` 클릭 후 등록 목록에 `openai-default` 항목이 나타나면 정상.*

5.1.2 OpenAI API Key 입력 + Save

> ***steps*** *— API 키 입력 후 저장 + 등록 확인.*

`OpenAI API Key` 필드에 본인의 키 (`sk-` 로 시작하는 약 50자 문자열) 를 붙여 넣으시고 `Save` 버튼을 누르시면 자격증명 1건이 등록됩니다.

> ⚠ ***자주 막히는 곳*** *— 키를 워크플로 LLM 노드의 헤더에 직접 평문으로 박지 마십시오. 워크플로 JSON 을 내보내실 때 키가 그대로 노출되어 GitHub 등에 실수로 push 되면 보안 사고로 이어집니다. 반드시 본 절의 방식으로 Credentials 메뉴에 등록한 뒤 모델 노드에서 자격증명 이름만 참조하시는 것이 표준입니다.*

본 항이 끝나시면 6장 Hello World 의 LLM 노드 모델 선택 단계에서 `openai-default` 가 드롭다운에 노출됩니다.

5.2 사내 Custom LLM (Gemma 호환 BaseURL) 등록

5.2.1 ChatOpenAI Custom 자격증명 선택

> ***steps*** *— 사내 LLM 엔드포인트가 OpenAI 호환 API 면 ChatOpenAI Custom 자격증명 1건으로 등록 가능합니다.*

`Add Credential` → 검색창에 `Custom` 입력 → `ChatOpenAI Custom` 또는 `OpenAI Compatible` 카드 선택. 다음과 같은 호환 서버들이 본 자격증명 1건으로 모두 처리됩니다.

호환 서버	특징
vLLM	고성능 추론 서버. PagedAttention 기반
Ollama	로컬·온프렘 친화. Llama·Mistral·Gemma 호환
LM Studio	데스크탑 UI 기반 로컬 모델 서버
TGI (Text Generation Inference)	Hugging Face 공식 추론 서버

위 서버들은 모두 OpenAI 호환 모드를 지원하므로 동일 자격증명 양식으로 등록하실 수 있습니다.

5.2.2 BaseURL `/v1` 끝맺음 강제

> ***troubleshooting*** *— 사내 LLM 자격증명 실패의 90% 원인이 BaseURL 끝* `/v1` *누락입니다.*

BaseURL 필드에 사내 LLM 엔드포인트 URL 을 입력하시되, **반드시** `/v1` **으로 끝맺으셔야** 합니다.

```
✓ http://gemma.internal:8000/v1
✓ http://vllm-server.dev.local:8080/v1
× http://gemma.internal:8000      ← /v1 누락. 모든 호출 404.
× http://vllm-server.dev.local:8080/v1/   ← 끝에 / 가 추가로 붙어도 일부 서버에서 404.
```

> ***⚠ 자주 막히는 곳 —*** `/v1` *누락 시 모델 호출이 404 로 실패합니다. 사내 LLM 운영팀에서 endpoint URL 을 받으실 때 끝의* `/v1` *유무를 한 번 더 확인하시는 것이 안전합니다.*

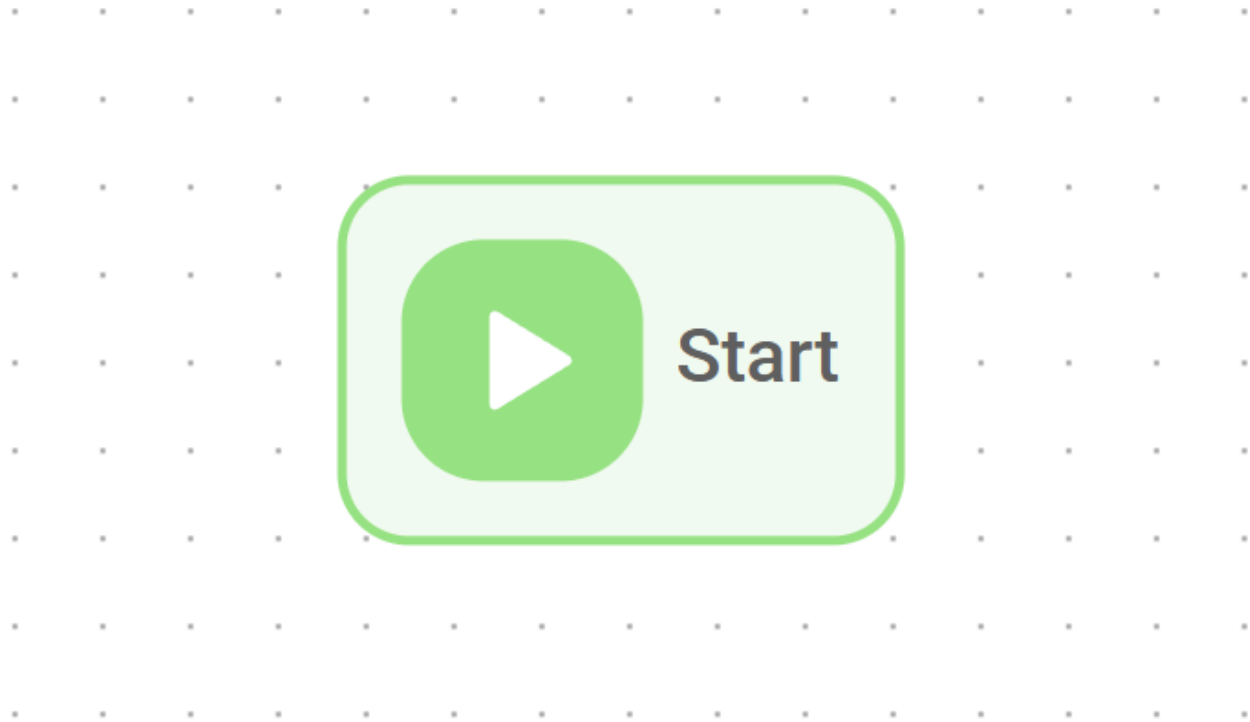
여기까지 5장 자격증명 등록이 끝났습니다. 다음 6장에서 본 자료 첫 실습 산출인 Hello World 3노드 워크플로를 만들고 첫 응답을 받겠습니다.

6장: Hello World — `Start → LLM → Direct Reply` 3노드 워크플로

6.1 빈 캔버스 진입과 Start Node 배치

이번 장에서는 이 교재의 첫 실습 산출 단원입니다. 본 장을 끝까지 마치시면 `Start → LLM → Direct Reply` 3노드 워크플로 1개로 사용자 입력에 대한 첫 응답을 받으시게 됩니다.

6.1.1 빈 캔버스 진입



> ***steps*** *— 좌측* `Agentflows` *메뉴 → 우측 상단* `Add New` *→ 빈 캔버스 진입.*

좌측 사이드바의 `Agentflows` 를 클릭하시면 Agentflows 목록 화면으로 진입합니다. 우측 상단의 `Add New` 버튼을 클릭하시면 URL 이 `/agentflows/new` 또는 `/agentcanvas/new` 로 변경되고, 중앙에 그리드 배경의 빈 캔버스가 표시됩니다. 우측 상단의 `Save` 버튼은 회색 (비활성) 상태입니다.

> ✅ ***체크포인트*** *— 빈 캔버스 + 비활성* `Save` *버튼이 보이시면 정상 진입.*

URL 끝이 `/agentcanvas/new` 인지 한 번 더 확인하시는 것이 v2 캔버스 진입의 가장 단순한 식별 신호입니다. `/chatflows/new` 로 들어오셨다면 잘못된 메뉴를 클릭하신 것이므로 좌측 사이드바에서 다시 `Agentflows` 를 선택하십시오.

6.1.2 Start Node 배치 + Input Type = Chat

> ***steps*** *— 노드 팔레트 검색* `start` *→* `startAgentflow` *드래그 → 캔버스 좌측에 드롭 → Input Type=Chat.*

좌측 노드 팔레트의 검색창에 `start` 를 입력하시면 `startAgentflow` 카드가 노출됩니다. 카드를 클릭한 채 캔버스 좌측 영역으로 드래그·드롭하시면 Start Node 1개가 배치됩니다. 배치된 노드를 클릭하시면 우측 Inputs 패널이 열립니다.

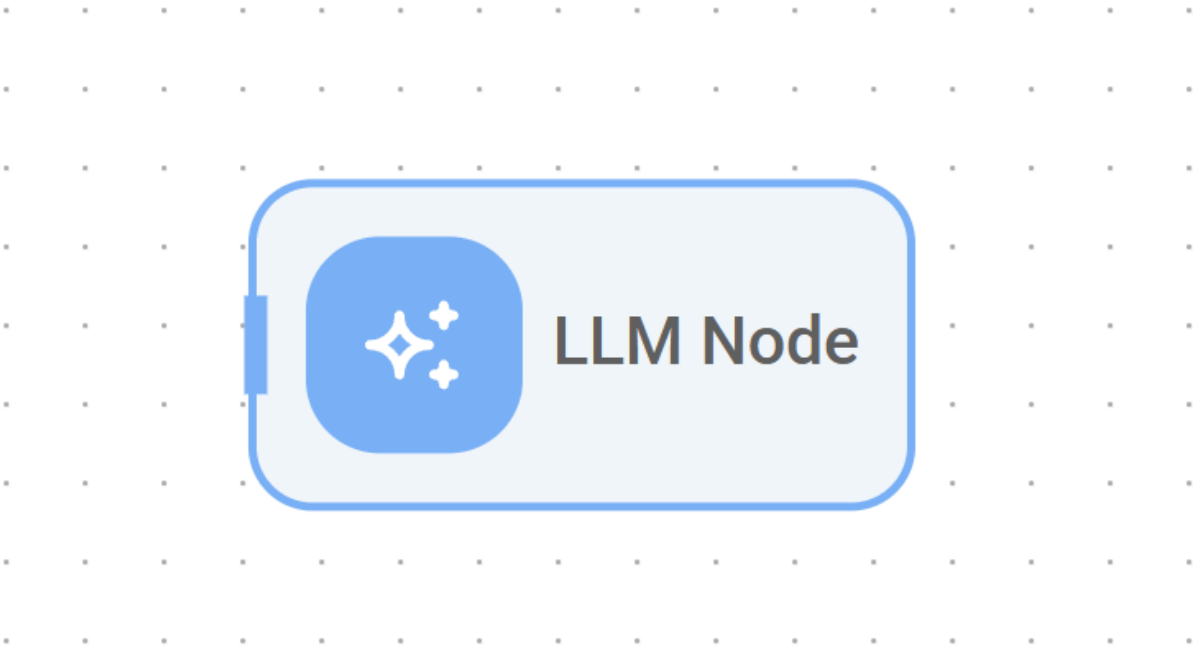
필드	Hello World 단계 설정
Input Type	`Chat` (채팅 UI 사용)
Ephemeral Memory	(선택 안 함)

필드	Hello World 단계 설정
Flow State	(비워둠 — 12장에서 다룸)

> ✅ ***체크포인트*** *— 캔버스에 Start 노드 1개 + Input Type=Chat 설정 확인.*

Flow State 패널은 본 Hello World 단계에서는 비워두십시오. Flow State 가 필요해지면 12장에서 본격 학습하시게 됩니다.

6.2 LLM Node 배치 + Model 선택



6.2.1 LLM Node 배치 + Model 선택

> ***steps*** *— 노드 팔레트 검색* `llm` *→* `llmAgentflow` *드래그 → 캔버스 중앙에 드롭 → Model = ChatOpenAI + Credential = openai-default.*

노드 팔레트 검색창에 `llm` 을 입력하시면 `llmAgentflow` 카드가 노출됩니다. 카드를 캔버스 중앙으로 드래그·드롭하시면 LLM Node 1개가 배치됩니다. 노드를 클릭하시면 우측 Inputs 패널이 열립니다.

필드	Hello World 단계 설정
Model	`ChatOpenAI` (드롭다운에서 선택)
Credential	`openai-default` (5장에서 등록한 자격증명)
Model Name	`gpt-4o-mini` 또는 사내 표준 모델명
Messages	(다음 항 6.2.2 에서 설정)
Memory	(선택 안 함 — Hello World 단계)

> ⚠ ***자주 막히는 곳*** *— Credential 드롭다운이 비어 있으시다면 5장 자격증명 등록이 누락된 것입니다. 좌측 사이드바의* `Credentials` *메뉴로 돌아가셔서* `openai-default` *1건을 먼저 등록하시고 본 단계를 다시 진행하십시오.*

입문 단계 모델은 비용 효율 모델인 `gpt-4o-mini` 로 충분합니다. 이 교재의 모든 실습이 동일 모델로 동작합니다.

6.2.2 Messages 입력 — `{{ $flow.input }}` 변수

> ***steps*** *— LLM Node 의 Messages 패널에 User 메시지 1개 추가 → `{{ $flow.input }}` 한 줄 입력.*

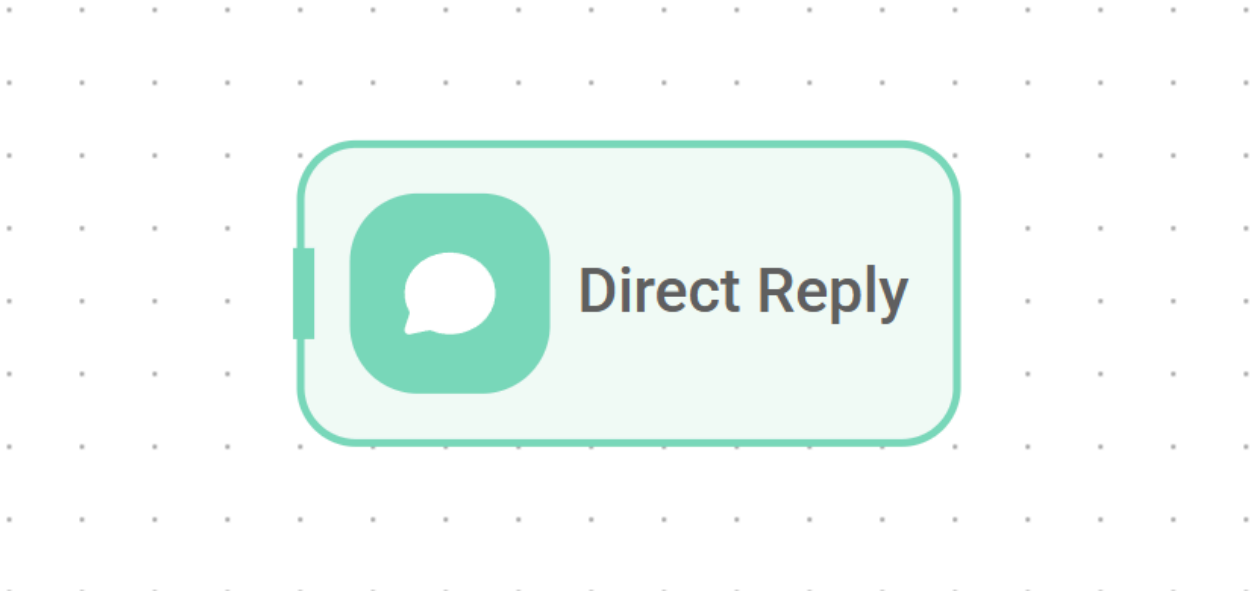
Messages 패널의 `+ Add Message` 버튼을 클릭하시면 메시지 추가 다이얼로그가 열립니다. Role 드롭다운에서 `User` 를 선택하시고, Content 필드에 다음을 그대로 입력하십시오.

```
{{ $flow.input }}
```

이 한 줄이 채팅 패널의 사용자 입력을 LLM 메시지로 흘려보내는 표준 패턴입니다. `{{` 를 타이핑하시면 자동완성이 사용 가능한 모든 변수 (이전 노드 출력 · `$flow.input` · `$vars` · `$flow.state` 키) 를 보여드립니다.

> 💡 ***팁*** *— `{{ $flow.input }}` 외에 시스템 메시지를 1개 추가하시면 (예: "당신은 친절한 한국어 어시스턴트입니다") 챗봇의 페르소나 설정이 가능합니다. 본 Hello World 단계는 User 메시지 1개만으로 충분합니다.*

6.3 Direct Reply 배치 + 엣지 연결 + 첫 응답



6.3.1 Direct Reply 배치 + 엣지 연결

> ***steps*** *— 노드 팔레트 검색* `direct` *→* `directReplyAgentflow` *드래그 → 캔버스 우측에 드롭 → Message=* `{{ $llmAgentflow_0.output }}` *→ 3 노드 엣지 좌→우 연결 → Save 활성화.*

노드 팔레트 검색 `direct` → `directReplyAgentflow` 카드를 캔버스 우측에 드롭하시면 Direct Reply Node 1개가 배치됩니다. 노드를 클릭하시면 Inputs 패널의 `Message` 필드가 보이는데, 다음과 같이 입력하십시오.

```
{{ $llmAgentflow_0.output }}
```

여기서 `$llmAgentflow_0` 는 캔버스의 첫 번째 LLM 노드의 자동 부여 ID 입니다. `{{` 자동완성으로 정확한 ID 를 확인하실 수 있습니다.

이제 3개 노드를 엣지로 연결합니다. Start 노드의 우측 앵커에서 마우스를 끌어 LLM 노드의 좌측 앵커로 놓으십시오. LLM 노드의 우측 앵커에서 다시 Direct Reply 노드의 좌측 앵커로 연결하십시오.

> ✅ ***체크포인트*** *— 우측 상단* `Save` *버튼이 회색에서 파란색으로 활성화되면 3 노드 + 2 엣지 구성이 정상.*

`Save` 버튼을 클릭하시면 워크플로가 저장됩니다. 저장 후 다이얼로그에서 워크플로 이름을 `hello-world` 같이 부여하시면 됩니다.

6.3.2 첫 응답 확인 + Direct Reply 누락 트러블슈팅

> ***result*** *— 우측 채팅 패널에 사용자 입력 1줄 → LLM 응답 출력.*

저장이 완료되시면 우측 상단의 채팅 아이콘을 클릭하셔서 채팅 패널을 여십시오. 입력창에 `안녕` 을 입력하시고 Enter 를 누르시면 LLM 응답이 패널에 출력됩니다.

```
사용자: 안녕
어시스턴트: 안녕하세요! 무엇을 도와드릴까요? 궁금하신 점이나 알고 싶으신 내용을 자유롭게 말씀해 주세요.
```

응답이 정상 출력되면 이 교재의 첫 실습 산출인 Hello World 3노드 워크플로가 완성된 것입니다.

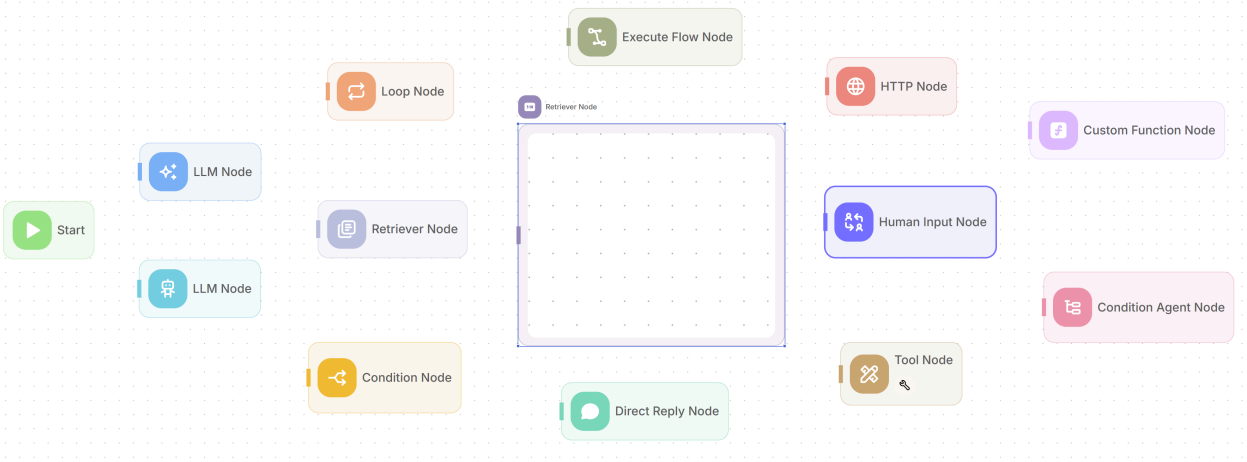
> ⚠ ***자주 막히는 곳 — "응답이 안 나와요" 의 90% 원인 3가지***
>
> 1. ***Direct Reply 노드 누락*** *— Start + LLM 만 있고 Direct Reply 가 없으면 응답이 채팅 패널까지 도달하지 못합니다.*
> 2. ***엣지 연결 누락*** *— 노드는 배치했지만 노드 간 엣지를 연결하지 않으셨다면 워크플로가 실행되지 않습니다.*
> 3. ***Credential 미선택*** *— LLM 노드의* `Credential` *드롭다운에서* `openai-default` *를 선택하지 않으셨다면 모델 호출이 실패합니다.*

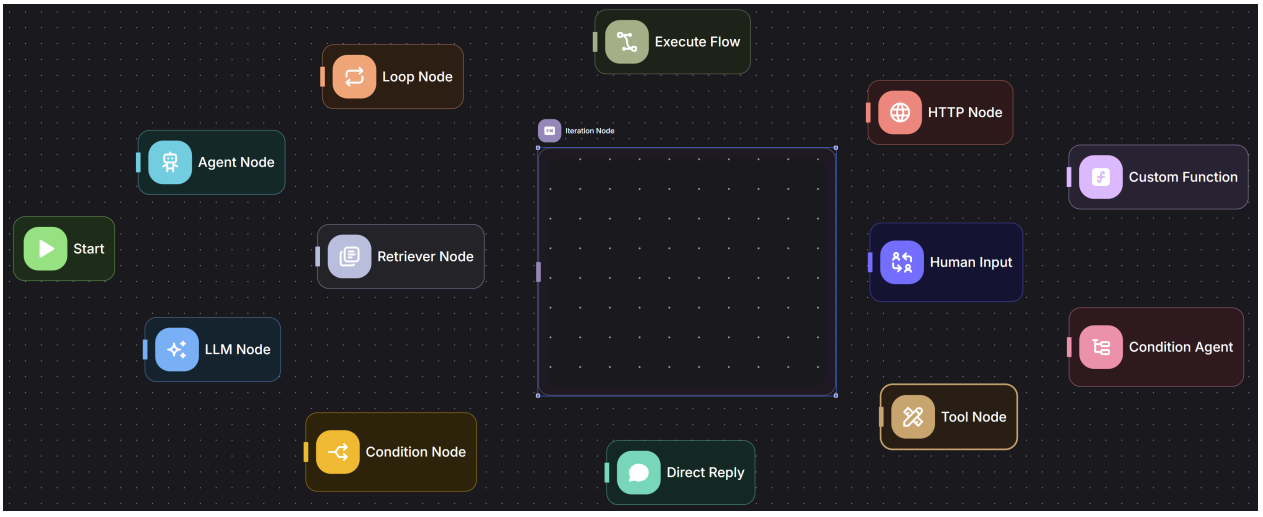
위 3가지를 1번씩 점검하시면 거의 모든 첫 응답 실패가 해소됩니다.

여기까지로 6장 Hello World 학습이 끝났습니다. 이 교재의 5가지 핵심 질문 중 Q3 (Hello World 직접 만들기) 에 답하실 수 있는 단계에 도달하신 것입니다. 다음 7장에서는 14개 표준 노드를 한 표로 한눈에 보여드리고, 8~11장에서 각 노드를 5필드 포맷으로 깊이 학습합니다.

7장: 14 노드 카탈로그 — cheatsheet 한눈에 보기

7.1 14 노드 한 줄 요약 + 카탈로그 캡처





이번 장에서는 14개 표준 노드의 cheatsheet 입니다. 본 표 1개로 자기 필요에 따라 8·9·10·11장 중 어디로 점프할지 결정하실 수 있습니다.

7.1.1 14 노드 한 줄 요약 표

#	노드	한 줄 정의	본 자료 상세 장
1	**Start**	워크플로 진입점, Flow State 키 선언	8장
2	**LLM**	도구 없는 단일 LLM 호출	8장

#	노드	한 줄 정의	본 자료 상세 장
3	**Agent**	도구를 자율 선택하는 ReAct 에이전트	8장
4	**Tool**	지정된 단일 도구를 결정적으로 호출	9장
5	**Retriever**	Document Store 에서 의미 검색	9장
6	**HTTP**	범용 REST API 호출	9장
7	**Condition**	규칙 기반 결정적 분기	10장
8	**Condition Agent**	LLM 기반 동적 분기 (자연어 시나리오)	10장
9	**Iteration**	배열 for-each 반복	10장
10	**Loop**	이전 노드로 회귀 (while 루프)	10장
11	**Human Input**	워크플로 일시정지 · HITL 체크포인트	8장
12	**Direct Reply**	최종 응답 전송, 분기 종료	8장
13	**Custom Function**	임의 JavaScript 실행	11장
14	**Execute Flow**	다른 Flowise 워크플로를 모듈로 호출	11장

> 💡 ***팁*** *— 본 표를 PDF 페이지 1장으로 출력하셔서 책상 옆에 두시거나 휴대폰 캡처로 저장하시면 학습 내내 가장 자주 참조하시게 됩니다.*

7.1.2 카탈로그 캡처 (라이트 + 다크)

위 두 캡처는 Flowise 좌측 노드 팔레트를 검색 키워드 없이 펼친 상태의 전체 카탈로그입니다. 사내 환경 테마와 일치하는 캡처를 선택해서 참조하십시오. 14개 노드 카드의 시각적 배치를 1번 보시면, 8~11장 학습 시 각 노드의 위치를 손쉽게 떠올리실 수 있게 됩니다.

영문 docs 의 노드 이름 (`*Agentflow` 접미) 이 본 캡처에서 한 번 더 강화됩니다. 검색 키워드 (`agentflow v2` 명시) + 본 캡처 두 가지를 기억하시면 v1 자료와의 혼동을 영구 차단하실 수 있습니다.

7.2 5필드 포맷의 의미 — 14 노드 학습의 일관 기준

이 교재의 8~11장은 모든 노드를 동일한 5필드 포맷으로 정리합니다. 본 절은 그 5필드의 의미를 미리 풀어드리는 학습 안내입니다.

7.2.1 5필드 각 필드의 의미와 사용 가이드

필드	의미	학습자가 활용하실 때
(1) 한 줄 정의	이 노드가 무엇인가	사내 동료에게 노드 설명할 때 그대로 인용
(2) 언제 쓰는가	사용 시나리오 1줄	워크플로 설계 시 노드 선택의 1차 기준
(3) 주요 입력 필드	우측 Inputs 패널의 키 목록	캔버스에서 노드 클릭 후 어떤 필드를 채울지
(4) 출력	다음 노드가 받는 데이터 형식	후속 노드의 입력 매핑 시
(5) 흔한 실수	학습자가 자주 막히는 패턴	트러블슈팅 시 가장 먼저 확인

본 5필드 포맷이 이 교재의 차별화 포인트 중 첫 번째 입니다. 영문 docs 는 각 노드 페이지 구조가 자유로워 비교 학습이 어렵지만, 이번 교재는 동일 포맷을 강제하여 노드 간 차이를 표 단위로 한눈에 비교하실 수 있게 합니다. 특히 (5) 흔한 실수가 운영 단계에서 가장 자주 인용되는 필드입니다.

7.2.2 LLM vs Agent — 가장 중요한 구별 한 줄

이 교재의 가장 중요한 노드 구별점은 LLM Node 와 Agent Node 의 차이입니다. 단일 기준은 다음 한 줄입니다.

> 💡 ***LLM vs Agent 한 줄 구별*** *— 도구 호출 여부. 도구가 필요하면 Agent, 그렇지 않으면 LLM.*

단순 응답 · 요약 · 분류 · 번역 · 페르소나 챗봇은 모두 LLM Node 면 충분합니다. 날씨 API · 사내 DB 검색 · 외부 검색 엔진 호출처럼 도구를 동적으로 선택해야 하는 경우는 Agent Node 입니다. 두 노드를 잘못 선택하시면, 도구가 필요한 시나리오에 LLM 노드를 쓰셨을 때 모델이 "이런 도구가 있다면 좋겠다" 고 텍스트로만 응답하고 실제 호출은 안 합니다.

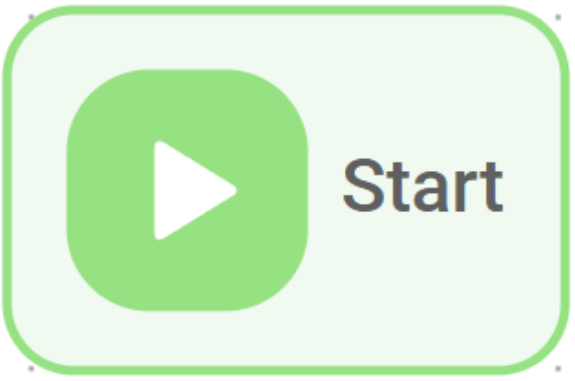
본 7장 cheatsheet 학습이 끝나시면 다음 8~11장의 각 노드 깊이 학습을 5필드 포맷으로 일관되게 진행하실 수 있게 됩니다.

8장: 핵심 5노드 깊이 보기 — Start · LLM · Agent · Human Input · Direct Reply

8.1 Start · LLM · Direct Reply — Hello World 골격 3노드 깊이

본 절은 6장 Hello World 의 3노드를 5필드 포맷으로 1번 더 정리합니다. 6장에서 다룬 기본 입력 필드 외에 Memory · JSON Schema · Update Flow State 같은 추가 옵션도 함께 학습합니다.

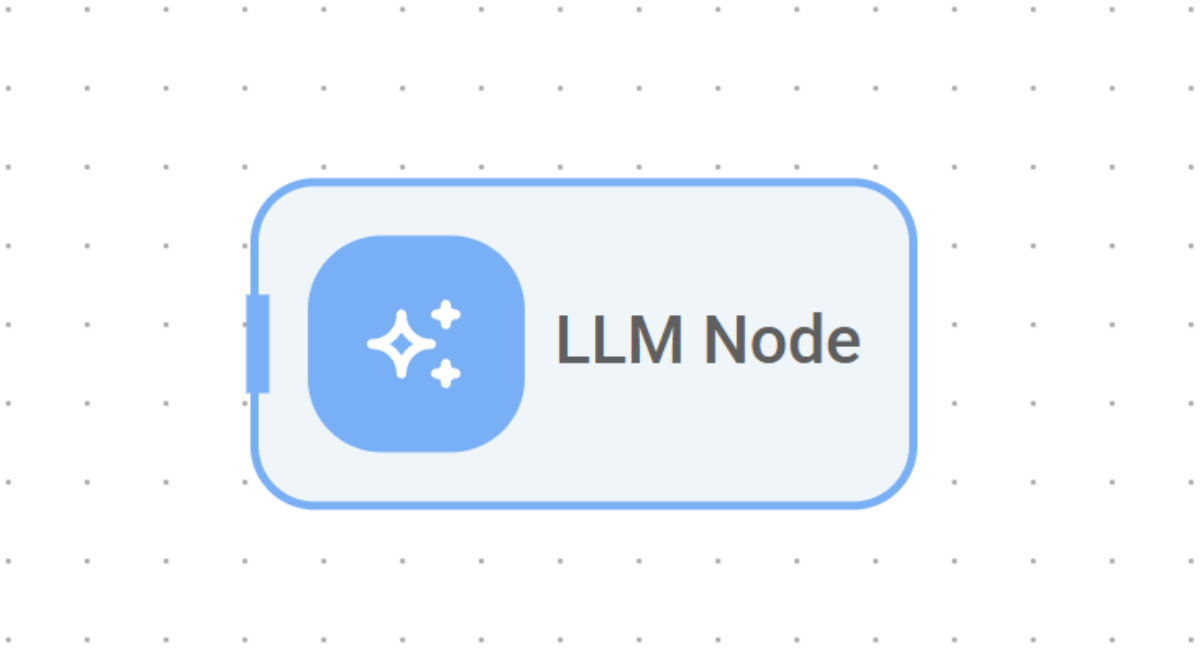
8.1.1 Start Node 5필드 — Input Type · Ephemeral Memory · Flow State



필드	의미 / 선택 가이드
(1) 정의	워크플로 실행의 시작점. 입력 방식 (Chat/Form) 정의 + `$flow.state` 키 선언·초기화
(2) 언제	모든 AgentFlow V2 워크플로에 단 하나만 존재해야 함 (옵션 아님)
(3) 입력	`Input Type` (Chat/Form), `Ephemeral Memory` (세션 한정 메모리), `Flow State` (키-값 사전 선언)
(4) 출력	초기 입력값 + 초기화된 `$flow.state`
(5) 흔한 실수	Flow State 키를 Start 에서 빠뜨리고 뒤쪽 노드에서 처음 등장시키면 동작하지 않음

Ephemeral Memory 를 켜시면 한 세션 안에서만 메모리가 유지되고 세션 종료 시 사라집니다. 영구 메모리가 필요하시면 LLM/Agent 노드의 `Memory=Vector` 옵션으로 외부 VectorStore 에 적재하셔야 합니다. Flow State 의 상세는 12장에서 다루지만, "Start 에서 모든 키를 선언한다" 는 규칙은 본 항에서 미리 새기십시오.

8.1.2 LLM Node 5필드 — Model · Messages · Memory · JSON Schema · Update Flow State



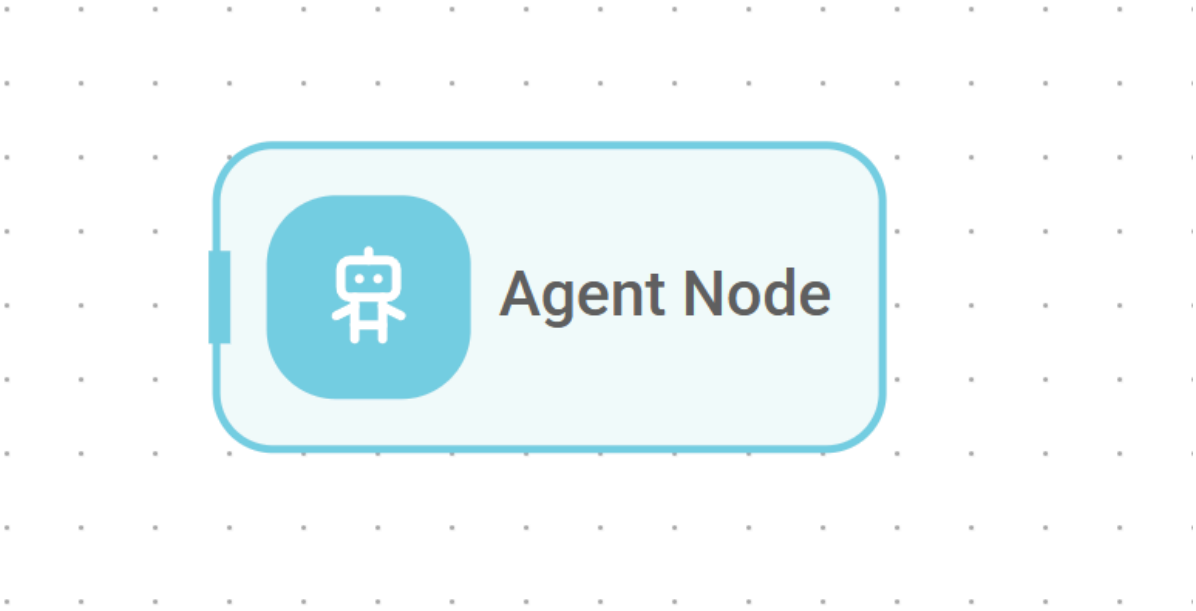
필드	의미 / 선택 가이드
(1) 정의	도구 없이 LLM 을 한 번 호출해 텍스트 또는 JSON 구조 생성
(2) 언제	단순 응답·요약·분류·번역·구조화 추출 등 "도구 호출이 필요 없는" 작업
(3) 입력	`Model`, `Messages`, `Memory`, `JSON Schema` (선택), `Update Flow State` (선택)
(4) 출력	생성된 텍스트 또는 JSON 객체
(5) 흔한 실수	도구 호출이 필요한 시나리오에 LLM 을 쓰면 모델이 텍스트로만 응답하고 실제 호출은 안 함

`Memory` 필드는 4종 전략 (Buffer · Window · Summary · Vector) 중 하나를 선택합니다. 각 전략의 차이는 다음과 같습니다.

Memory 전략	한 줄 비교
Buffer	전체 대화 이력을 그대로 누적 (가장 단순, 토큰 비용 ↑)
Window	최근 N 턴만 유지 (토큰 비용 절감, 장기 기억 손실)
Summary	LLM 으로 대화 요약 압축 (토큰 절감 + 핵심 보존)
Vector	VectorDB 에 임베딩 저장 후 유사 청크 검색 (장기 기억, 외부 의존)

JSON Schema 옵션은 다음 노드가 정형 데이터를 받아야 할 때 사용합니다. 예를 들어 LLM 으로 사용자 의도를 `{"intent": "refund", "confidence": 0.92}` 같은 구조로 추출하시려면 Schema 를 명시하시면 됩니다.

8.2 Agent Node 깊이 보기 + LLM vs Agent 명확화



8.2.1 Agent Node 5필드 — Tools · Document Stores · Memory · Max Iterations

필드	의미 / 선택 가이드
(1) 정의	도구 목록과 지식 소스를 받아 LLM 이 ReAct 추론으로 도구를 동적 선택
(2) 언제	사용자 의도에 따라 호출할 도구·검색 문서가 달라지는 경우
(3) 입력	`Tools` (도구 목록), `Document Stores` (RAG 저장소), `Memory` (4종 전략), `Max Iterations`
(4) 출력	최종 응답 텍스트 + 중간 도구 호출 trace
(5) 흔한 실수	도구를 너무 많이 등록하면 에이전트가 헤맴 (특히 도구 설명이 비슷한 경우)

> ***⚠ 자주 막히는 곳*** — *`Tools` 슬롯의 도구 개수는 5~7개 이내로 시작하시기를 권합니다. 각 도구의 자연어 설명을 차별화하는 데 시간을 투자하시는 것이 ReAct 추론 정확도의 1차 결정 요인입니다.*

`Document Stores` 슬롯은 RAG 용 문서 저장소를 등록하시면 자동으로 검색 도구로 변환됩니다. 각 저장소에 자연어 설명 (예: "사내 인사 정책 문서" / "제품 매뉴얼") 을 붙여두시면 에이전트가 그 설명을 보고 적절한 저장소를 고릅니다. `Max Iterations` 는 추론 루프 최대 반복 횟수로 무한루프 방지용입니다 (기본 5~10).

8.2.2 ReAct 추론 — 한 라운드의 흐름

Agent Node 의 내부는 ReAct (Reason + Act) 패턴으로 동작합니다. LLM 이 "Thought → Action → Observation → Thought ..." 루프로 도구 호출을 동적 결정합니다.

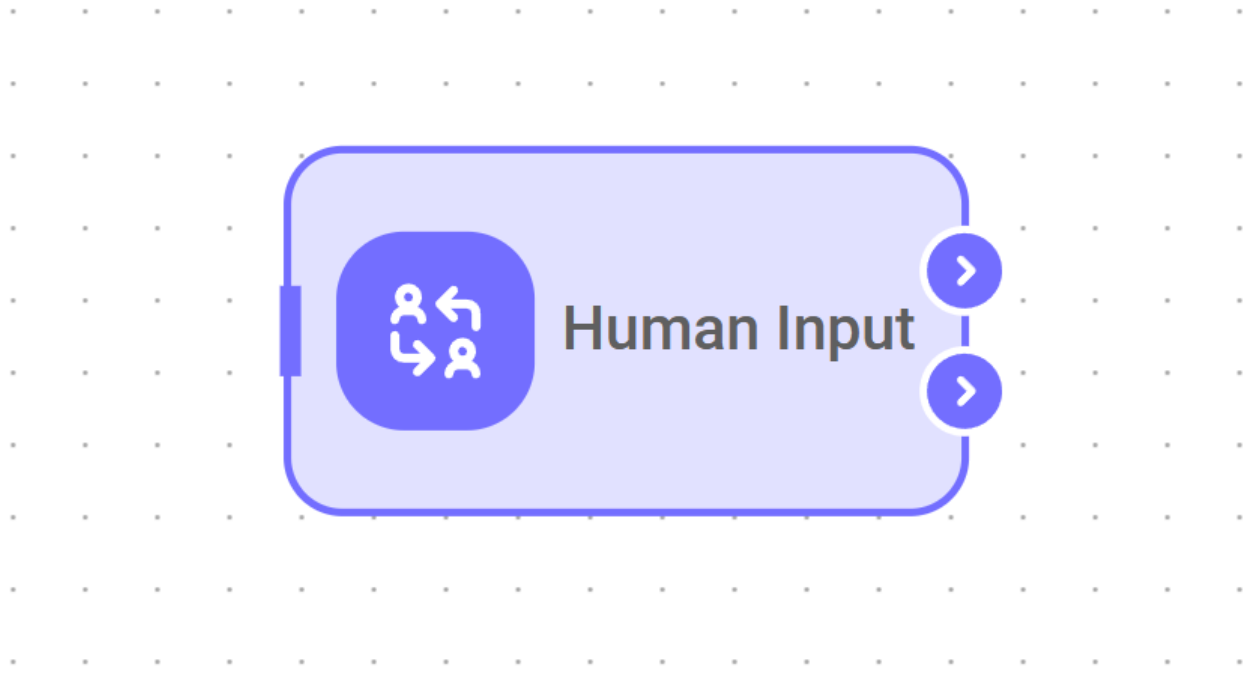
```
[ReAct 추론 trace 예시]

Thought: 사용자가 "내일 서울 날씨" 를 묻고 있다. Weather API 도구를 호출해야겠다.
Action: weather_api(city="Seoul", date="2026-05-27")
Observation: {"temperature": 22, "condition": "맑음", "humidity": 55}
Thought: 도구 응답을 받았다. 사용자에게 한국어로 정리해서 답해야겠다.
Final Answer: 내일 서울은 맑음, 기온 22도, 습도 55% 예상됩니다.
```

본 trace 가 Agent Node 의 디버깅 출력으로 그대로 노출됩니다. 워크플로가 의도와 다르게 동작할 때 본 trace 를 1번 보시면 어느 단계에서 잘못된 도구가 선택되었는지 즉시 파악하실 수 있습니다.

> 💡 ***팁*** *— Agent Node 의 trace 가 너무 길어진다면* `Max Iterations` *가 너무 큰 것입니다. 5~10 사이에서 시작하시고, 필요 시 점진적으로 늘리시는 것이 토큰 비용 통제의 표준입니다.*

8.3 Human Input Node — HITL 의 1급 시민



8.3.1 Human Input 5필드 — Description Type · Description

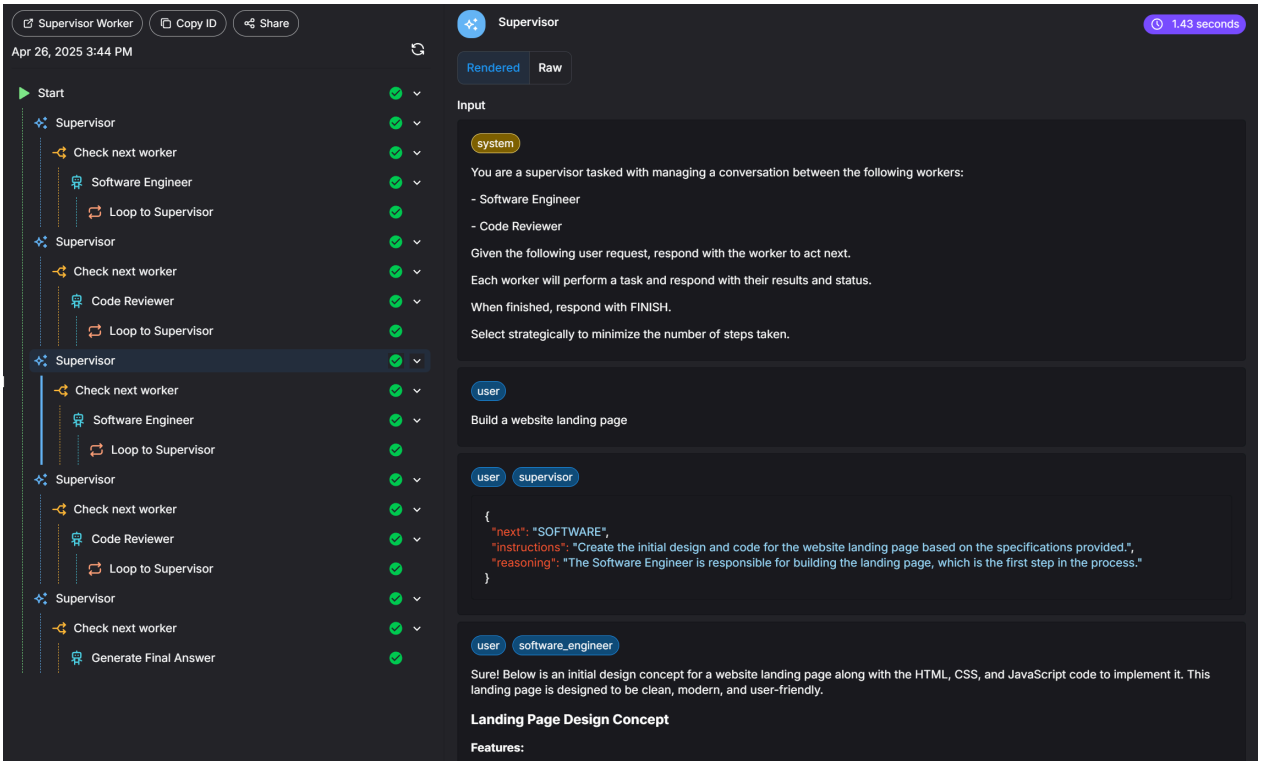
필드	의미 / 선택 가이드
(1) 정의	워크플로 일시정지 + 사람의 입력·승인·피드백 대기
(2) 언제	AI 작성 콘텐츠 발송 전 검수, 외부 결제·송금 직전 승인, 민감 정보 접근 확인
(3) 입력	`Description Type` (Fixed/Dynamic), `Description` (사용자 안내 문구)
(4) 출력	두 분기 앵커 — `proceed` (승인) / `reject` (반려)

필드	의미 / 선택 가이드
(5) 흔한 실수	체크포인트 영속을 위해 SQLite/Postgres 가 영속 볼륨에 마운트되어야 함

`Description Type=Fixed` 는 모든 호출에서 동일 문구를 표시합니다. `Dynamic` 은 LLM 이 현재 상황에 맞춰 매번 안내 문구를 생성합니다. 후자가 사용자 경험은 더 자연스럽지만 토큰 비용이 추가됩니다.

> ✅ ***체크포인트*** *— Docker 1줄 실행 시* `-v ~/.flowise:/root/.flowise` *볼륨 마운트가 적용되어 있다면 체크포인트 영속이 자동 보장됩니다. 사내 Kubernetes 운영 시 PVC (Persistent Volume Claim) 설정 필요.*

8.3.2 HITL 패턴 D — Draft → 사람 승인 → 발송 / 재작성



HITL 의 표준 패턴은 다음과 같습니다.

> Start → LLM (draft) → Human Input → {proceed: HTTP send | reject: Loop back to LLM}

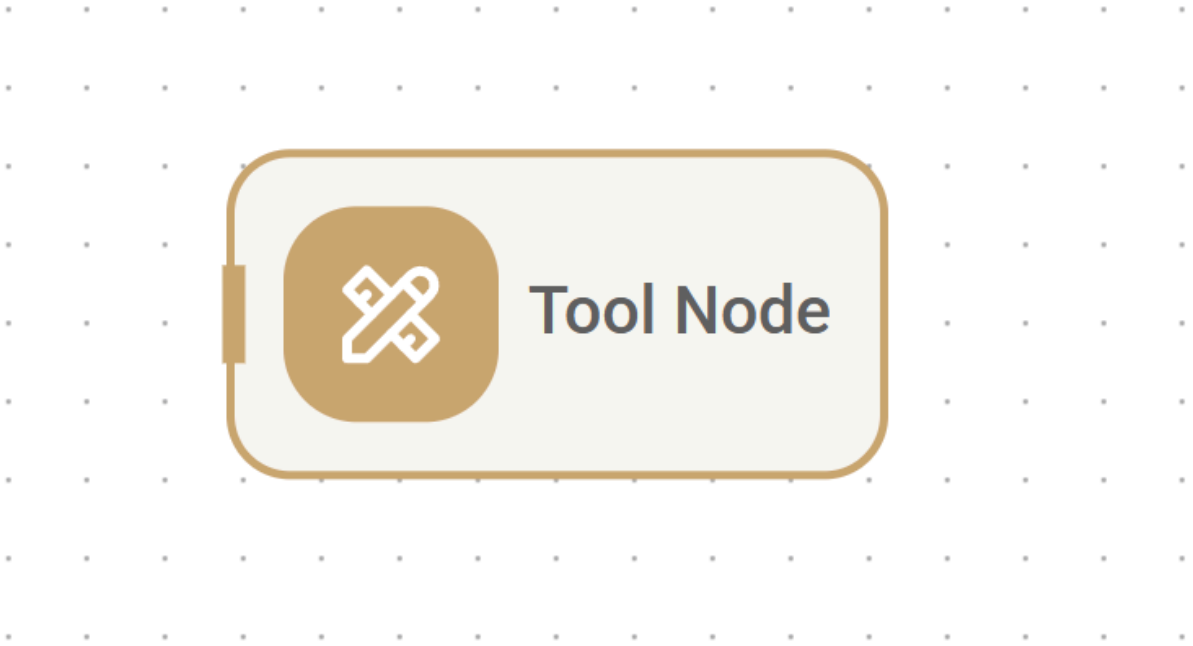
마케팅 이메일 발송 전 매니저 검수, 외부 송금 직전 책임자 승인, 사내 결재 시나리오 등에 즉시 적용 가능한 패턴입니다. `proceed` 경로는 HTTP Node 로 외부 서비스 호출, `reject` 경로는 Loop Node 로 LLM 단계로 회귀하여 다시 draft 작성하도록 설계합니다.

본 패턴이 13장 5 패턴의 패턴 D 로 1번 더 강화됩니다. 두 번째 보시는 시점에 이미 HITL 의 본질을 흡수하시게 됩니다.

여기까지로 8장의 핵심 5노드 (Start · LLM · Agent · Human Input · Direct Reply — 12번 노드는 6장에서 다룸) 깊이 학습이 끝났습니다. 본 5개 노드만 잘 다루시면 단일 에이전트 챗봇·HITL 워크플로의 90% 를 구축하실 수 있습니다. 다음 9장부터 외부 자원 연결 노드 3종으로 들어갑니다.

9장: Tool · Retriever · HTTP — 외부 자원 연결 3노드

9.1 Tool Node — 결정적 도구 호출 vs Agent



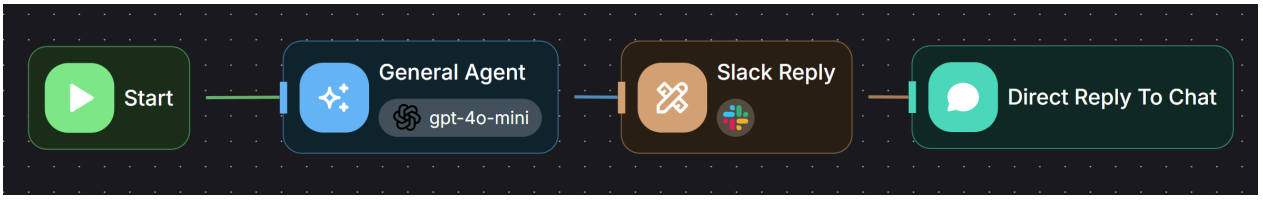
9.1.1 Tool 5필드 — Tool · Input Arguments · Update Flow State

필드	의미 / 선택 가이드
(1) 정의	워크플로 설계자가 "이 도구를 이 인자로 호출해라" 라고 지정해 실행
(2) 언제	흐름이 정해져 있어 LLM 판단이 필요 없을 때 (예: 무조건 Slack 알림)
(3) 입력	`Tool` , `Input Arguments` , `Update Flow State` (선택)
(4) 출력	도구 호출 원시 결과 (API 의 JSON 문자열·텍스트·수치 값 등)
(5) 흔한 실수	Agent Node 와 혼동해 도구 후보 묶음을 넣으려는 경우 (Tool 은 단 하나만)

Tool vs Agent 를 표 한 장으로 비교하면 다음과 같습니다.

항목	Tool Node	Agent Node
도구 선택	설계자가 사전 지정 (단일)	LLM 이 ReAct 추론으로 동적 결정 (다수)
LLM 호출	없음 (결정적)	있음 (각 라운드마다)
비유	"이거 하나만 정확히 처리해" (작업 지시서)	"재량껏 알아서 일하라" (직원)

9.1.2 MCP 도구 등록 + Agent · Tool 두 노드에서 동일 사용



좌측 사이드바의 `Tools` 메뉴에서 MCP 호환 도구를 등록하시면, Agent 노드의 `Tools` 슬롯과 Tool 노드의 `Tool` 드롭다운 양쪽에 자동 노출됩니다. 별도 커넥터 개발 없이 벤더 공식 도구를 그대로 끌어 쓰실 수 있다는 것이 MCP 1급 통합의 실용적 결과입니다.

> 💡 ***GitHub MCP 등록 예시*** — *`Tools` → `Add Tool` → `Custom MCP` 선택 → MCP Server URL 입력 (예: `https://github-mcp.example.com`) → 인증 토큰 등록 → Save. 등록 즉시 `Tools` 슬롯 드롭다운에 `github-search`, `github-create-issue` 같은 도구들이 자동 노출됩니다.*

Anthropic MCP 공식 카탈로그 에서 사내 사용 가능한 벤더 공식 MCP 를 미리 검색해 보시는 것도 도입 정당성 검증에 좋습니다.

9.2 Retriever Node — RAG 의 핵심 부품

9.2.1 Retriever 5필드 — Document Store · Query · Output Format

필드	의미 / 선택 가이드
(1) 정의	등록된 Document Store 에서 쿼리와 의미적으로 가까운 문서 조각 검색
(2) 언제	LLM 의 일반 지식만으로 답할 수 없는 사내 문서·매뉴얼·정책 기반 응답
(3) 입력	`Document Store` , `Retriever Query` , `Output Format` (Text / Text with Metadata)
(4) 출력	검색된 텍스트 청크 (또는 청크 + 출처 메타데이터)
(5) 흔한 실수	검색 결과를 곧바로 사용자에게 노출 (다음 LLM 의 컨텍스트로 받아 답하는 패턴이 표준)

`Output Format=Text` 는 본문만 반환합니다. 출처 인용이 필요하시면 `Text with Metadata` 를 선택하셔서 다음 LLM 노드의 시스템 프롬프트에 출처를 함께 노출하시면 됩니다.

RAG 의 표준 패턴은 다음과 같이 LLM 노드의 시스템 프롬프트로 검색 결과를 받습니다.

> [LLM 노드 System Message 예시]
>
> 당신은 사내 정책 안내 어시스턴트입니다. 다음 문맥을 참고해 사용자의 질문에 답하세요.
>
> {{ $retrieverAgentflow_0.output }}
>
> 문맥에 답이 없으면 "관련 정책 문서가 없습니다" 라고 답하세요.

이런 패턴이 표준이며, 검색 결과를 사용자에게 직접 보여드리지 않고 LLM 의 응답 생성 컨텍스트로만 활용하시는 것이 권장됩니다.

9.2.2 외부 VectorStore (Qdrant) 영속화 1줄

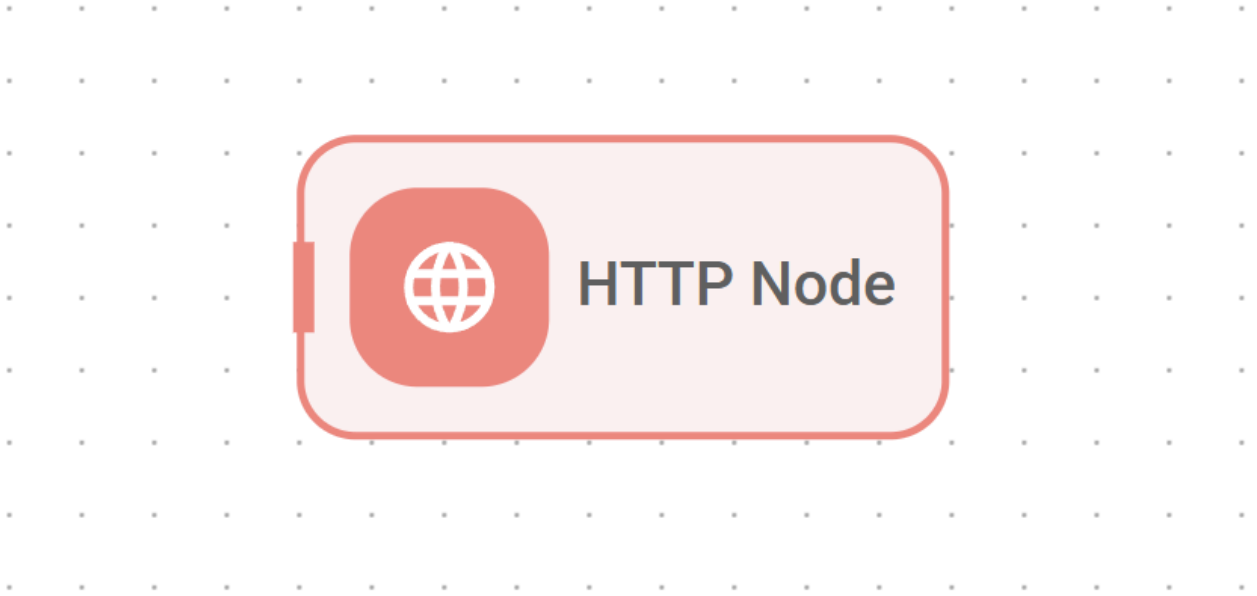
Document Store 의 백엔드를 Qdrant 같은 외부 VectorStore 로 변경하시면 Flowise 인스턴스 재시작에도 인덱스가 영속됩니다.

```
docker run -d -p 6333:6333 --name qdrant qdrant/qdrant
```

위 1줄로 Qdrant 인스턴스를 띄우신 뒤, Flowise 의 Document Store 의 백엔드 드롭다운에서 `Qdrant` 를 선택하시고 URL 에 `http://localhost:6333` 을 입력하시면 됩니다.

> 💡 **팁** — *사내 PoC 가 운영 단계로 이관될 때 외부 VectorStore 가 사실상 필수입니다. 상세는 자매 자료 (Flowise Intermediate) 에서 다룹니다.*

9.3 HTTP Node — 범용 REST 통합



9.3.1 HTTP 5필드 — Method · URL · Headers/Body · Credentials · Response Type

필드	의미 / 선택 가이드
(1) 정의	GET/POST/PUT/DELETE/PATCH HTTP 요청을 보내고 응답 수신
(2) 언제	사내 시스템·외부 SaaS·자체 API 와 연동 (별도 도구 등록 없이 즉시 호출)
(3) 입력	`Method` , `URL` , `Headers/Body` , `HTTP Credentials` , `Response Type`
(4) 출력	응답 본문 (파싱된 객체 또는 문자열) + 상태 코드
(5) 흔한 실수	인증 토큰을 직접 헤더에 평문으로 박으면 워크플로 JSON 내보내기 시 노출

JSON Body 예시는 다음과 같습니다.

```
{
 "channel": "#alerts",
 "text": "AI Agent 가 새 결재 요청을 처리했습니다: {{ $flow.state.requestId }}",
 "username": "Flowise Bot"
}
```

본문 안의 `{{ $flow.state.requestId }}` 같은 변수가 실행 시점에 치환됩니다. `Response Type=JSON` 으로 설정하시면 응답이 객체로 파싱되어 다음 노드가 `.output.status` 같은 점 접근으로 필드를 읽으실 수 있습니다.

9.3.2 인증 토큰 평문 노출 차단 — Credentials 메뉴 사용

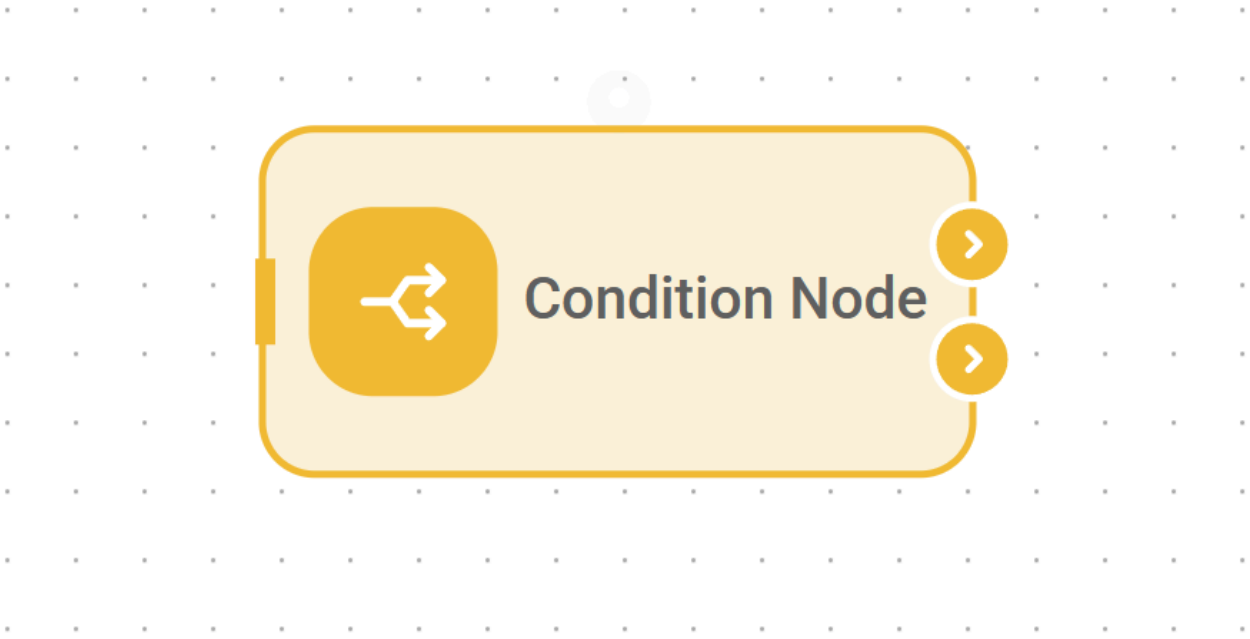
> ***⚠ 자주 막히는 곳 — 인증 토큰 평문 노출 사고***
>
> *인증 토큰을 직접 Headers 필드에 `Authorization: Bearer sk-xxxxx...` 같이 박지 마십시오. 워크플로 JSON 내보내기 시 토큰이 그대로 노출되어 GitHub 에 실수로 push 되면 즉시 보안 사고로 이어집니다. 반드시 좌측 `Credentials` 메뉴에서 `Authorization` 타입 자격증명으로 사전 등록한 뒤 본 노드의 `HTTP Credentials` 드롭다운에서 참조하시는 것이 표준입니다.*

본 권고가 사내 보안 정책 위반 1순위 사고의 가장 흔한 패턴이라, 본 절에서 1번 더 강조합니다.

여기까지로 9장 외부 자원 연결 3노드 학습이 끝났습니다. 다음 10장에서 흐름 제어 4노드 (Condition · Condition Agent · Iteration · Loop) 로 들어갑니다.

10장: Condition · Condition Agent · Loop · Iteration — 흐름 제어 4노드

10.1 Condition vs Condition Agent — 결정적 vs AI 분기



본 절은 두 분기 노드의 5필드를 동시 비교합니다. 단일 구별 기준은 "규칙으로 코드 가능한가, 자연어 의도인가" 입니다.

10.1.1 Condition 5필드 + 데이터 타입 가이드

필드	의미 / 선택 가이드
(1) 정의	사전 정의된 논리 규칙으로 워크플로 분기
(2) 언제	분기 조건이 명확히 코드 가능한 규칙일 때 (예: 점수 80 이상 합격)
(3) 입력	비교 대상 1 / 연산자 / 비교 대상 2 + 데이터 타입 (String/Number/Boolean)
(4) 출력	각 분기 평가 결과에 대응하는 다중 출력 앵커
(5) 흔한 실수	데이터 타입을 잘못 지정해 비교가 의도와 다르게 동작

연산자는 다음 5종을 자주 사용하시게 됩니다.

연산자	사용 예
equals	`userIntent equals "refund"`
notEqual	`priority notEqual "low"`
contains	`message contains "긴급"`
larger	`score larger 80` (Number 타입)
isEmpty	`userInput isEmpty` (빈 입력 처리)

> ⚠ ***자주 막히는 곳*** *— 점수 비교 시 데이터 타입을* `String` *으로 두시면* `"80"` *과* `"100"` *의 문자열 비교가 일어나서* `"80"` *이* `"100"` *보다 크다고 판정됩니다. 숫자 비교는 반드시* `Number` *타입을 선택하십시오.*

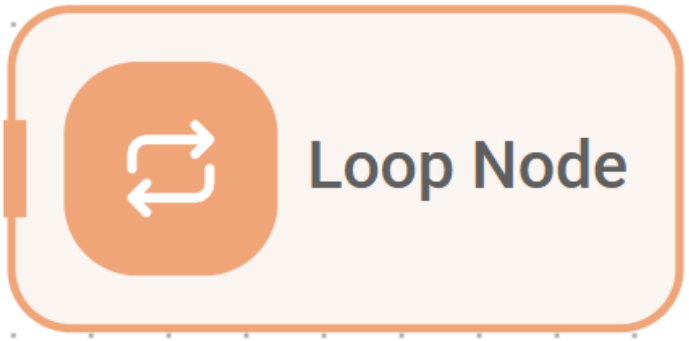
10.1.2 Condition Agent 5필드 + Scenarios 작성 가이드

필드	의미 / 선택 가이드
(1) 정의	LLM 이 입력을 분석해 사용자 정의 시나리오 중 하나로 라우팅
(2) 언제	분기 기준이 자연어 의도·감정·주제 등 규칙으로 코드하기 어려운 경우
(3) 입력	`Model` , `Instructions` , `Input` , `Scenarios` (이름·설명·출력 경로)
(4) 출력	각 Scenario 마다 하나씩 출력 앵커
(5) 흔한 실수	시나리오 설명이 모호하거나 서로 겹쳐 LLM 분류 일관성 ↓

> 💡 ***Scenarios 작성 팁*** *— 각 시나리오 설명은 짧고 구별이 분명하게 작성하십시오. 예시:*
>
> - ✓ *"환불 요청 — 사용자가 환불·반품·취소를 명시적으로 요청"*
> - ✓ *"기술 지원 — 사용자가 제품 사용 중 오류·작동 불량을 보고"*
> - × *"고객 문제 — 무언가 잘못된 상황" (너무 모호)*

분류용 LLM 은 빠른 모델 (`gpt-4o-mini`) 로 충분합니다. 응답 정확도보다 응답 속도가 중요하기 때문에, 분류 모델은 가벼운 모델로 두시고 본 응답 생성용 모델만 더 큰 모델로 분리하시는 것이 비용 대비 효과 측면에서 권장됩니다.

10.2 Loop vs Iteration — while vs for-each



본 절은 두 반복 노드의 5필드를 동시 비교합니다. 단일 구별 기준은 "배열 원소 순회 (for-each) 인가, 조건 만족 시까지 회귀 (while) 인가" 입니다.

10.2.1 Loop 5필드 + Max Loop Count 비용 가이드

필드	의미 / 선택 가이드
(1) 정의	워크플로 실행 흐름을 이미 실행된 노드 중 하나로 되돌림
(2) 언제	"사용자 입력 만족할 때까지 LLM 재시도", "에이전트가 정답 찾을 때까지 추론 반복"
(3) 입력	`Loop Back To` (어느 노드로 회귀), `Max Loop Count` (무한루프 방지, 기본 5)
(4) 출력	표준 forward 출력 없음. `Loop Back To` 로 흐름 회귀
(5) 흔한 실수	Max Loop Count 너무 크게 잡아 토큰 비용 폭증

> ***⚠ Max Loop Count 권장값*** *— 5~10 사이에서 시작하십시오. 사용자 입력 검증 같은 가벼운 시나리오는 3, ReAct 추론 보강 같은 무거운 시나리오는 10 정도가 안전합니다.*

10.2.2 Iteration 5필드 + 동시성 함정

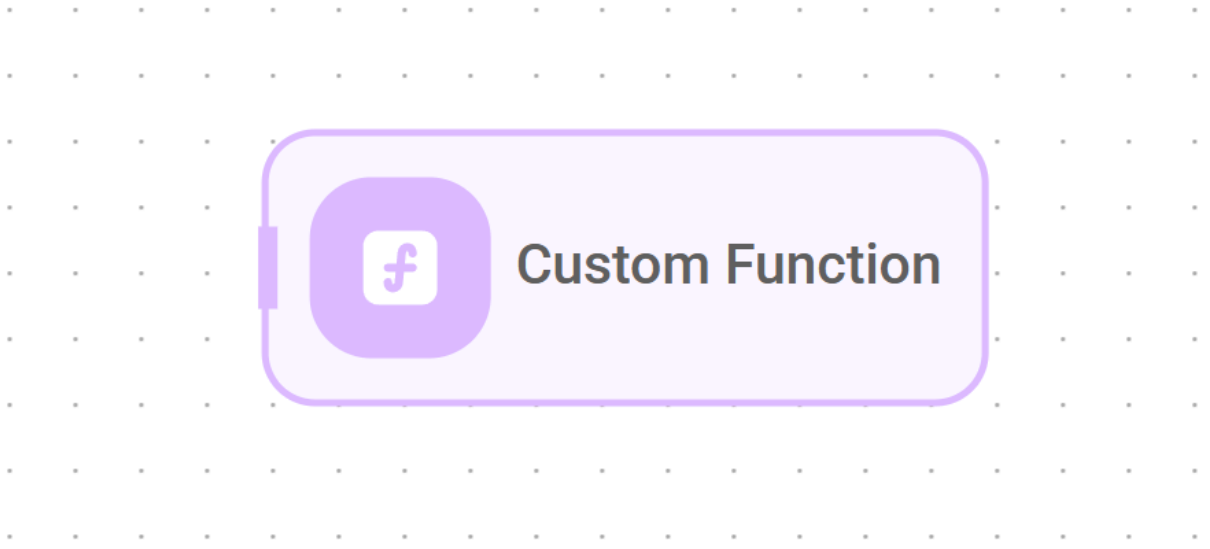
필드	의미 / 선택 가이드
(1) 정의	배열을 받아 각 원소에 대해 내부 서브플로를 한 번씩 실행 (for-each)
(2) 언제	"검색 결과 10건 각각에 대해 LLM 요약" 같은 배치 작업
(3) 입력	`Array Input` (반복 대상 배열, 예: `{{ $flow.state.itemList }}`)
(4) 출력	각 원소별 처리 결과의 집계 또는 서브플로 내 변수 최종 상태
(5) 흔한 실수	내부 서브플로가 외부 `$flow.state` 를 수정하면 동시성 문제로 결과가 꼬임

> ***⚠ 동시성 함정*** *— Iteration 안의 서브플로가 외부 `$flow.state.counter` 같은 키를 갱신하면 마지막 반복의 값만 남습니다 (덮어쓰기). 합산·누적이 필요하시면 Custom Function Node 로 명시 처리하시는 것이 안전합니다. 본 함정은 12장 Flow State 완전 이해의 함정 3번에서 1번 더 강조됩니다.*

여기까지로 10장 흐름 제어 4노드 학습이 끝났습니다. 다음 11장에서 확장 노드 2종 (Custom Function · Execute Flow) 으로 들어갑니다.

11장: Custom Function · Execute Flow — 확장 노드 2종

11.1 Custom Function — 서버사이드 JavaScript



11.1.1 Custom Function 5필드 + 컨텍스트

필드	의미 / 선택 가이드
(1) 정의	Node.js 환경에서 임의의 JavaScript 코드를 실행
(2) 언제	표준 노드로 표현하기 어색한 로직 (정규식·날짜 계산·복잡한 JSON 가공)
(3) 입력	`Input Variables` 패널 + 코드 에디터
(4) 출력	반환값 (문자열). JSON 객체 반환 시 `JSON.stringify` 후 다음 노드에서 파싱
(5) 흔한 실수	동기 함수만 가능한 줄 알고 async/await 회피 (실제로는 정상 동작)

코드 에디터에서 사용 가능한 컨텍스트는 다음과 같습니다.

컨텍스트 변수	의미
`$flow.sessionId`	채팅 세션 ID
`$flow.chatId`	채팅 ID
`$flow.input`	사용자 입력
`$flow.state`	Flow State 키-값 객체
`$vars.*`	워크스페이스 전역 변수

간단한 예시 — 사용자 입력에서 이메일 주소만 추출:

```
 const input = $flow.input;
const emailMatch = input.match(/[\w.-]+@[\w.-]+\.\w+/);
```

```
return emailMatch ? emailMatch[0] : "이메일 없음";
```

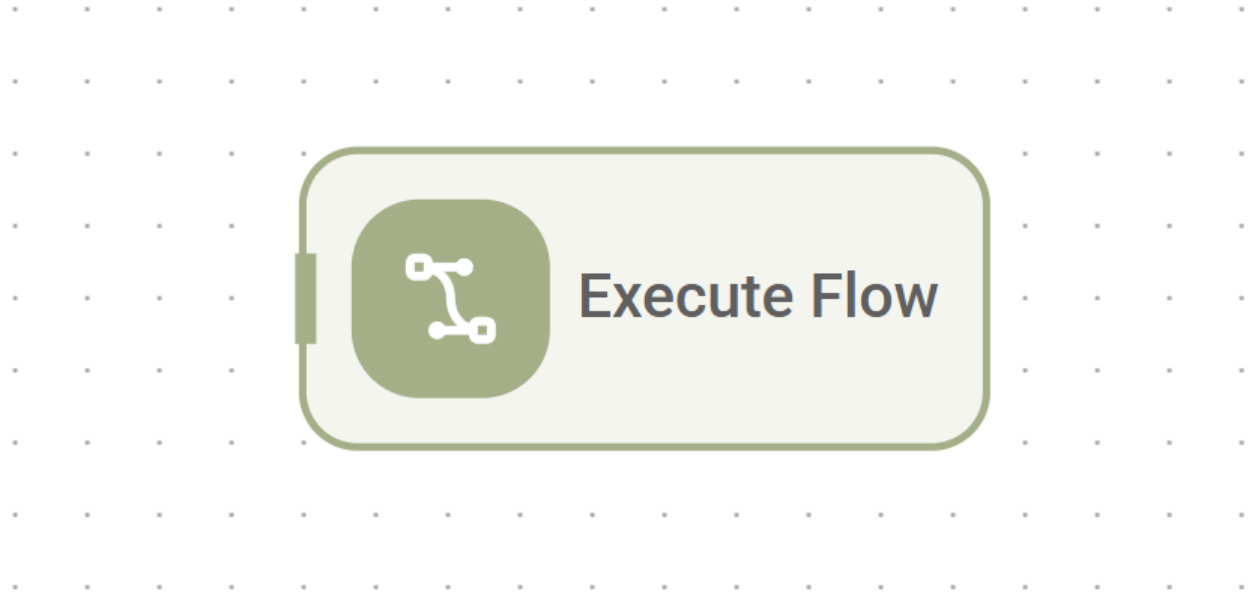
11.1.2 async/await 정상 동작 — 비동기 호출 가능

Custom Function 노드는 동기 함수만 가능한 것으로 오해되기 쉽지만, 실제로는 `async/await` 패턴이 정상 동작합니다. `await fetch(...)` 같은 외부 API 호출도 가능합니다.

```
const response = await fetch("https://api.example.com/data", {
 headers: { "Authorization": `Bearer ${$vars.API_TOKEN}` }
});
const data = await response.json();
return JSON.stringify(data);
```

> 💡 ***팁*** *— Custom Function 으로 외부 API 호출이 가능하지만, 1회성 호출이면 HTTP Node 가 더 명시적이고 가시적입니다. Custom Function 은 표준 노드로 표현이 어색한 변환 로직에 한정하시는 것이 가독성에 좋습니다.*

11.2 Execute Flow — 서브 워크플로 호출



11.2.1 Execute Flow 5필드 — Target Flow · Input · Config Overrides · Base URL

필드	의미 / 선택 가이드
(1) 정의	같은 Flowise 인스턴스의 다른 Chatflow / AgentFlow 를 모듈처럼 호출
(2) 언제	한 워크플로가 너무 커져 가독성 ↓ 또는 공통 로직 재사용
(3) 입력	`Target Flow` , `Input` , `Config Overrides` (JSON), `Base URL` , `Update Flow State` (선택)

필드	의미 / 선택 가이드
(4) 출력	서브 워크플로의 최종 응답
(5) 흔한 실수	서브 워크플로 응답 형식이 바뀌면 부모 워크플로 깨짐

필드	사용 예
Target Flow	동일 인스턴스의 다른 워크플로 선택
Input	전달할 입력 (`{{ $flow.input }}` 또는 가공된 값)
Config Overrides	호출 시점에 동적으로 덮어쓸 설정 (JSON)
Base URL	동일 인스턴스가 아닌 외부 Flowise 호출 시

`Config Overrides` 옵션이 강력한 기능입니다. 동일 서브 워크플로를 다른 모델·도구·프롬프트로 호출하실 수 있습니다.

11.2.2 서브 워크플로 = 공개 API 운영 권장

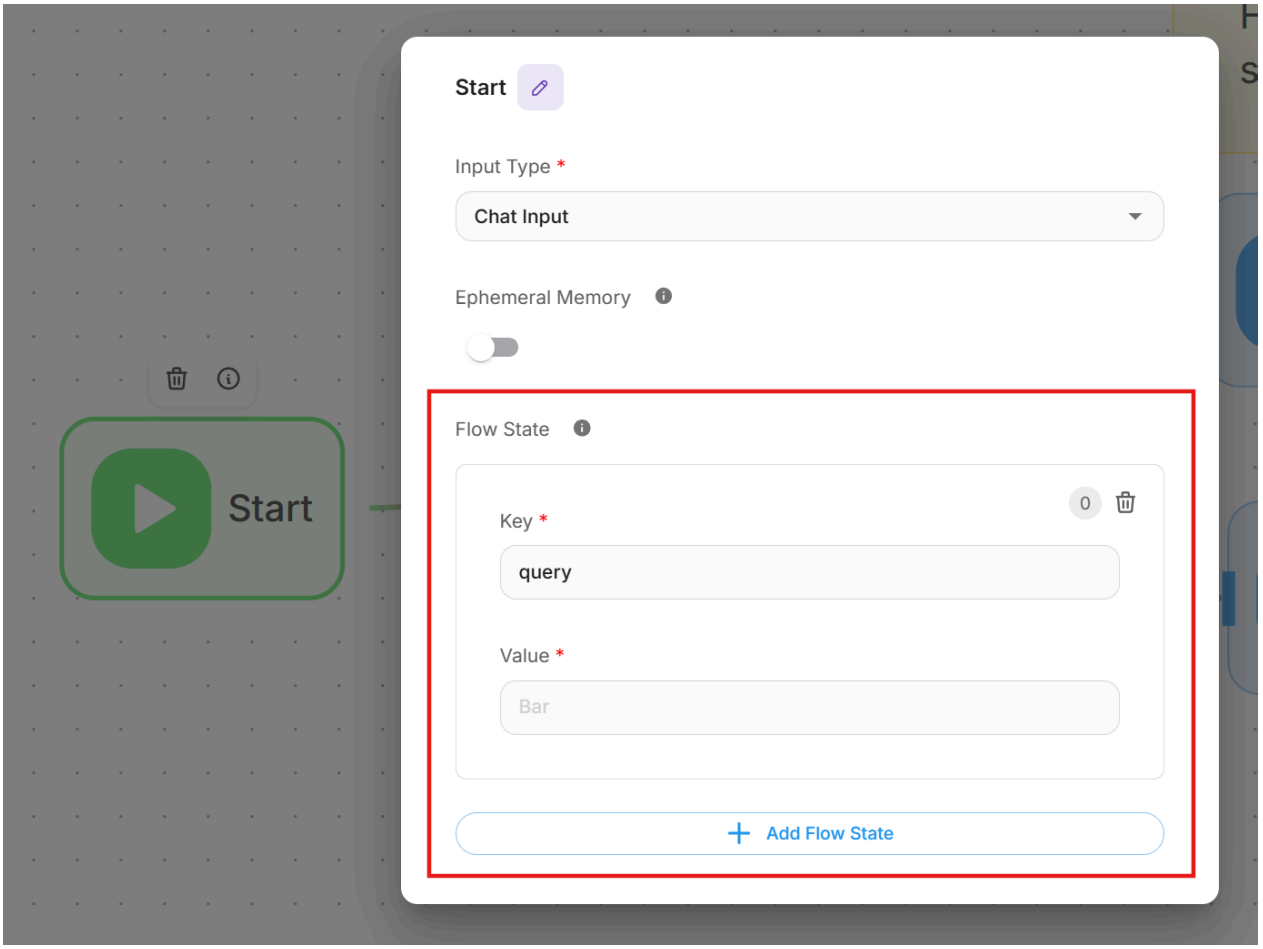
> 💡 ***서브 워크플로 = 공개 API 운영 룰*** *— 서브 워크플로의 응답 형식이 바뀌면 부모 워크플로가 깨집니다. 따라서 서브 워크플로는 "공개 API" 처럼 다음 두 가지를 고정하시는 것이 운영 거버넌스의 표준입니다.*
>
> 1. ***응답 스키마 고정*** *— 응답 JSON 의 키·타입을 변경하지 않음*
> 2. ***응답 스키마 문서 동봉*** *— 서브 워크플로마다 응답 스키마 문서 1개를 README 로 두기*

본 룰이 이 교재의 운영 단계에서 가장 자주 인용되는 거버넌스 권고입니다. 서브 워크플로 1개를 처음 만드실 때부터 본 룰을 적용하시면 사내 운영 단계의 사고를 사전 차단하실 수 있습니다.

여기까지로 11장 확장 노드 2종 학습이 끝났습니다. 다음 12장에서 Flow State 의 완전한 동작을 다룹니다.

12장: Flow State 완전 이해 — 초기화·읽기·갱신·함정

12.1 Flow State 수명·공유 범위·저장 대상



`$flow.state` 의 한 줄 정의와 비유는 3.3.1 에서 미리 보셨습니다. 이번 장에서는 그 위에 수명·공유 범위·저장 대상 3축과 초기화·갱신·읽기 3단계, 그리고 함정 3가지를 완비합니다.

12.1.1 수명 = 1회 실행 · 공유 범위 = 실행 내 모든 노드

축	정의
수명	한 번의 워크플로 실행이 시작될 때 생성, 끝나면 사라짐
공유 범위	같은 실행 안의 모든 노드가 읽고 쓸 수 있음
격리	다른 사용자의 동시 실행과는 완전 독립 (사용자 A 와 사용자 B 의 `$flow.state` 는 서로 격리)

비유하자면 한 회의에서 화이트보드에 적은 임시 메모입니다. 회의가 끝나면 지워지지만, 회의 도중에는 누구나 보고 고칠 수 있고, 다른 회의실의 메모와는 섞이지 않습니다.

12.1.2 저장 대상 = 임시 메모리, 영구 필요 시 외부

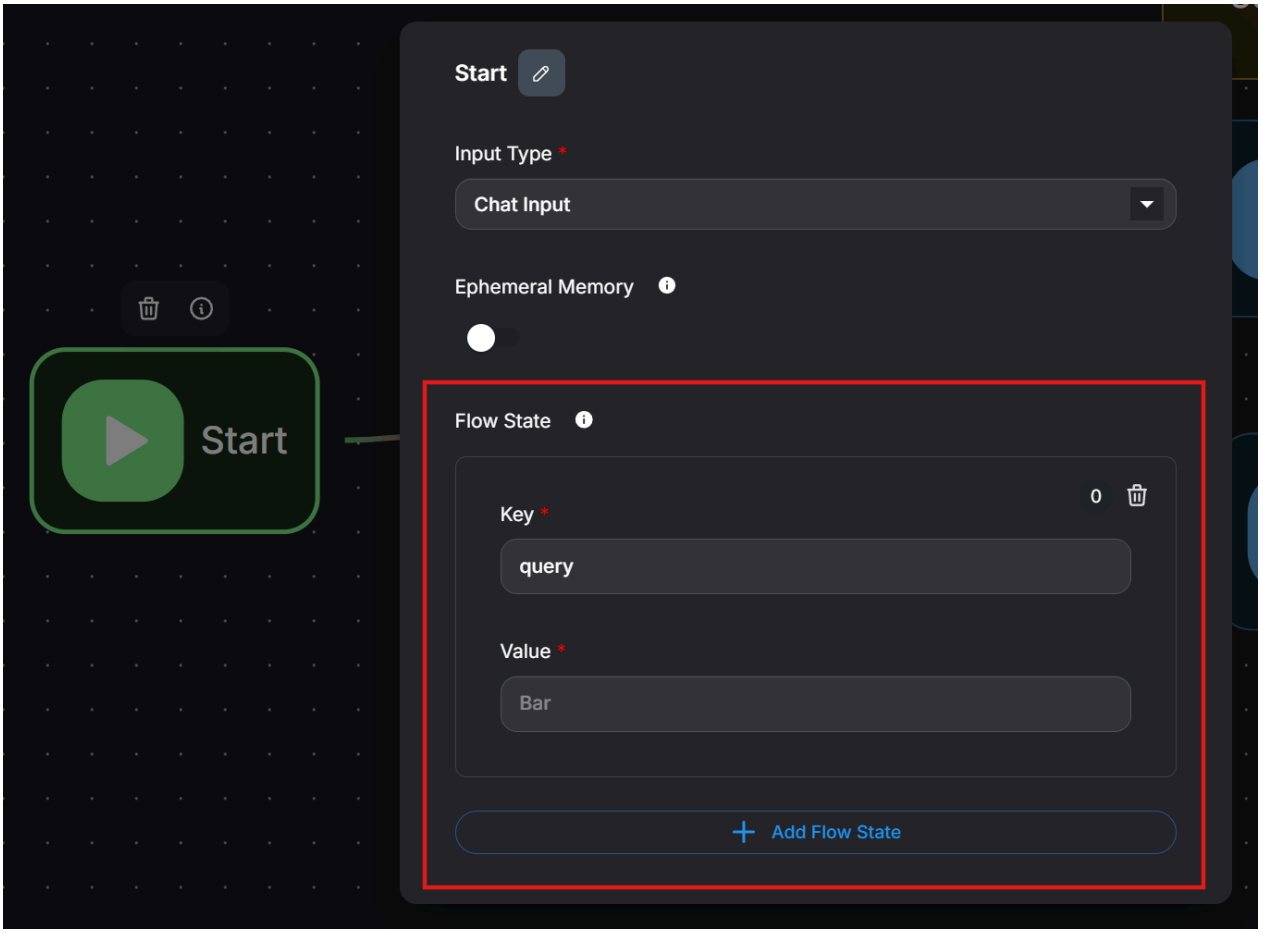
Flow State 는 임시 작업 메모리입니다. 영구 저장이 필요한 데이터 (사용자 프로필 · 이력 · 누적 카운터) 는 외부 DB 나 VectorStore 에 별도 적재하셔야 합니다.

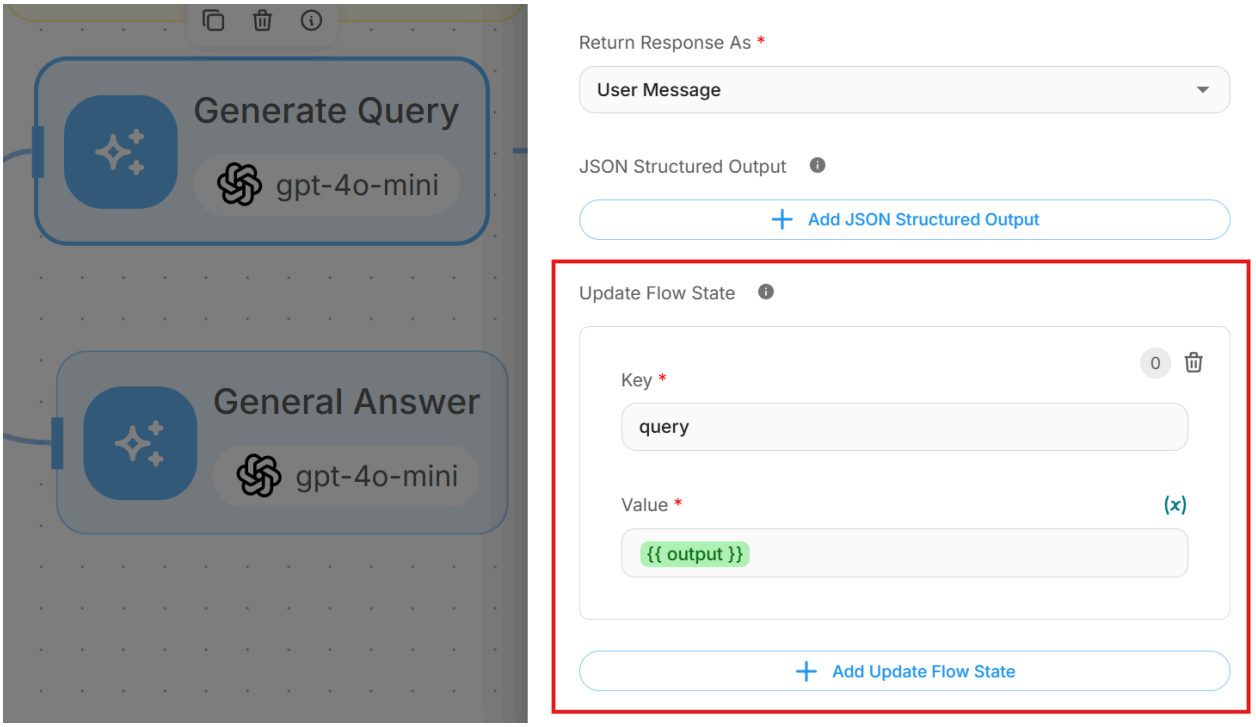
> 💡 ***영구 필요 시 외부 DB*** *— 사용자별 영구 메모리가 필요하시면 다음 패턴 중 선택하십시오.*

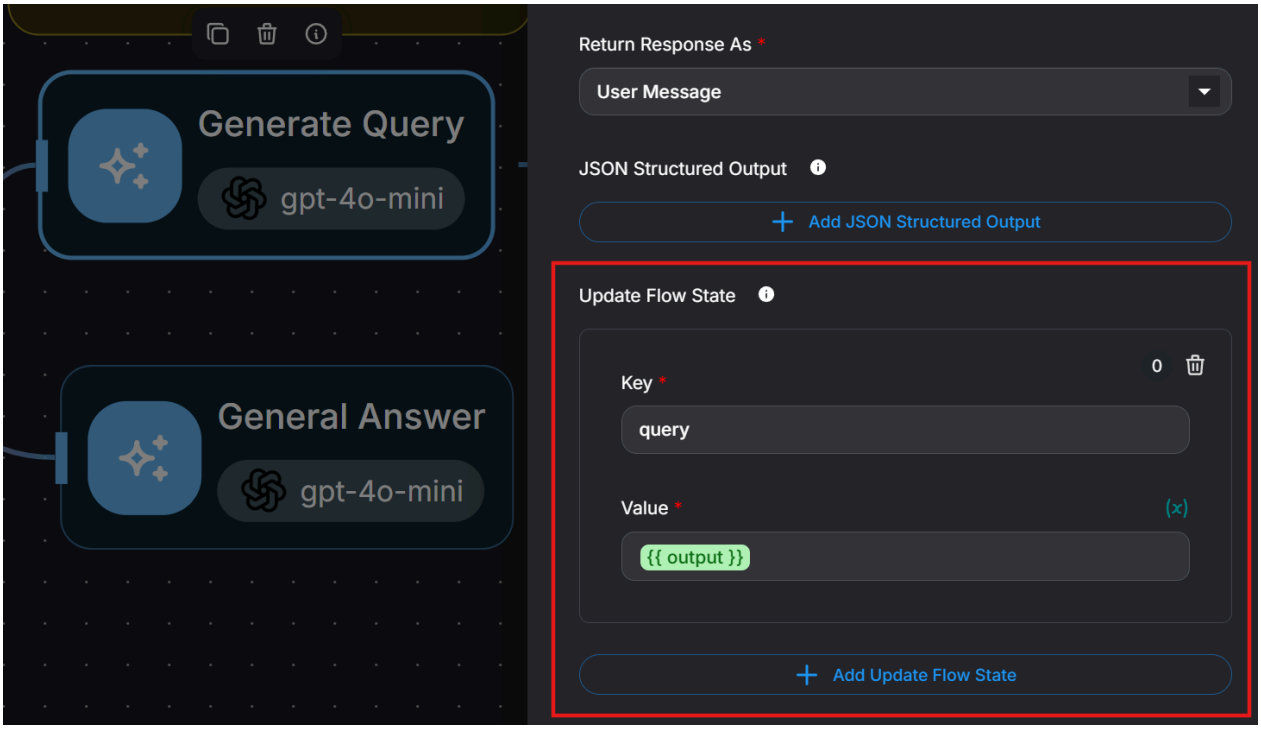
> - ***외부 DB + Custom Function 노드*** — *`await fetch(...)` 로 사내 DB 조회·저장*
> - ***LLM/Agent 의 Memory=Vector*** — *VectorStore 기반 장기 기억 자동 관리*
> - ***Redis + HTTP Node*** — *Redis REST 인터페이스로 키-값 영속*

Flow State 는 항상 "한 실행의 화이트보드" 라고 기억하시면, 영구 데이터와의 경계가 명확해집니다.

12.2 초기화 — Start Node 단독 선언 + 갱신 가능 노드 7개







12.2.1 Start Node 의 Flow State 패널에서 모든 키 선언

Flow State 의 모든 키는 Start Node 의 `Flow State` 패널에서만 사전 선언 가능합니다. 이후 어떤 노드도 새 키를 만들 수 없습니다. 오직 Start 에서 선언된 키만 갱신 가능합니다.

```
 # Start Node > Flow State 패널 예시
currentStep: "init"
totalScore: 0
```

```
userIntent: ""
customerId: ""
```

이 제약 덕분에 워크플로의 상태 스키마가 한 곳에서 명시적으로 관리됩니다. 본 단계는 사실상 워크플로의 `$flow.state` 스키마를 선언하는 행위라고 보시면 됩니다.

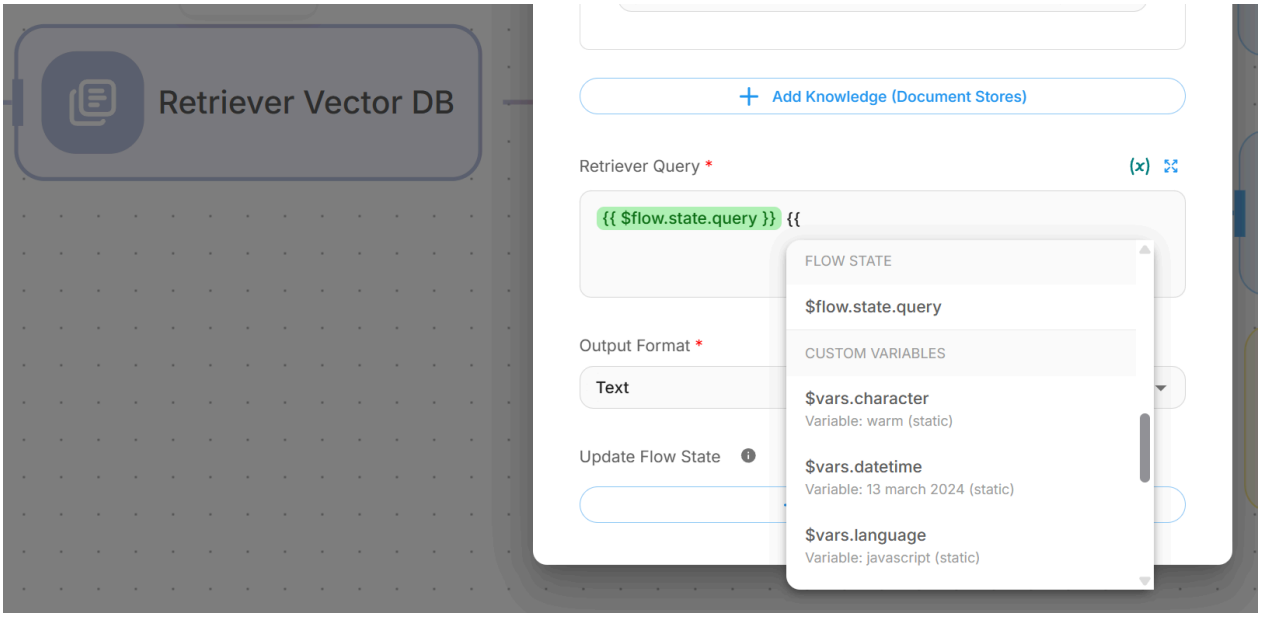
12.2.2 갱신 가능 7개 노드 + Update Flow State 옵션

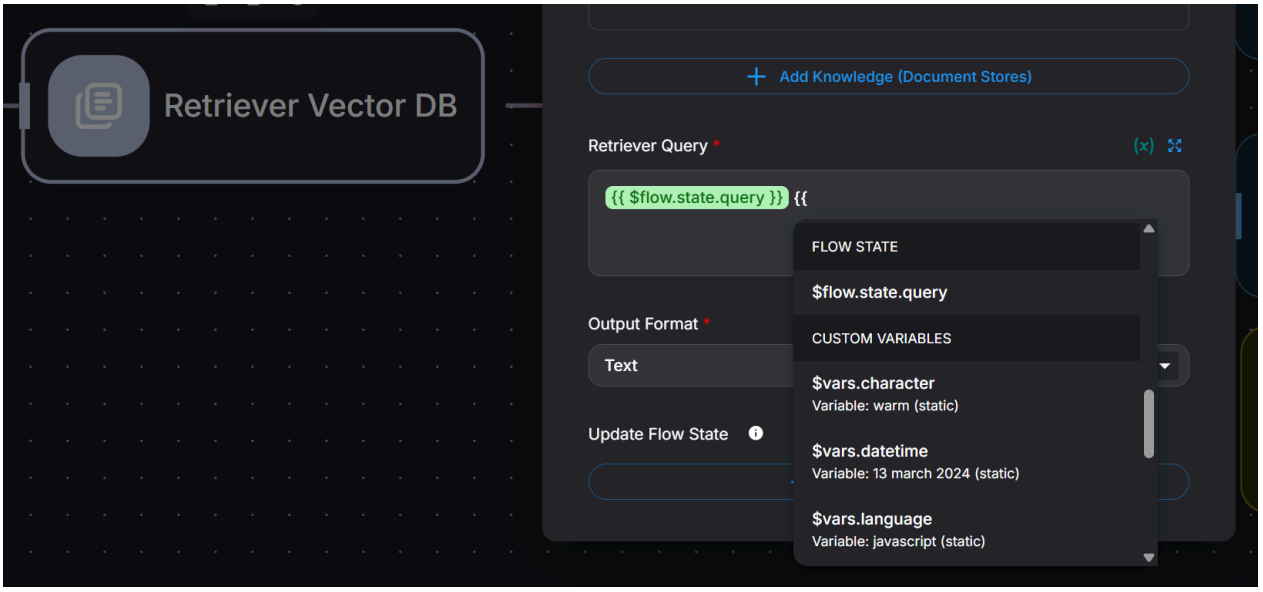
다음 7개 노드만 `Update Flow State` 옵션을 통해 기존 키 값을 갱신할 수 있습니다.

갱신 가능 노드 7개	갱신 불가 노드 7개
LLM	Start (선언 위치)
Agent	Direct Reply (분기 종료)
Tool	Condition (분기 평가만)
HTTP	Condition Agent (분기 평가만)
Retriever	Iteration (서브플로 관리)
Custom Function	Loop (회귀 제어)
Execute Flow	Human Input (사람 입력 대기)

갱신 패턴은 보통 "이번 노드 출력의 일부를 추출해 특정 키에 저장" 형태입니다. 예를 들어 LLM Node 가 분류 결과 `"refund"` 를 반환하면, 그것을 `userIntent` 키에 저장하는 식입니다. 값 입력란에서 `{{` 를 타이핑하시면 자동완성이 사용 가능한 모든 변수 (이전 노드 출력 · `$flow.input` · `$vars` · 기존 `$flow.state` 키) 를 보여드립니다.

12.3 읽기 + 함정 3가지





12.3.1 읽기 — `{{ $flow.state.키이름 }}` 한 줄

읽기는 더 단순합니다. 변수를 받아들이는 어떤 입력 필드에든 다음 문법을 그대로 적으시면 됩니다.

```
{{ $flow.state.userIntent }}
```

이 문법은 LLM 메시지 · HTTP URL · Condition 비교 대상 등 모든 텍스트 필드에서 동일하게 작동합니다. 예시:

```
[LLM Node System Message 예시]
당신은 친절한 어시스턴트입니다.
사용자의 현재 상태: {{ $flow.state.customerStatus }}
이 상태를 참고하여 적절한 응답을 생성해 주세요.
```

실행 시점에 `{{ $flow.state.customerStatus }}` 가 현재 값으로 치환됩니다. 읽기는 어디서나 동일, 갱신은 7개 노드만 — 이 규칙을 기억하시면 됩니다.

12.3.2 함정 3가지 — 미선언 키 · 세션 간 공유 오해 · Iteration 안에서의 갱신

> ***⚠ 함정 1 — 선언하지 않은 키 사용***
>
> *Start 에서 선언하지 않은 키를 뒤쪽 노드에서 처음 등장시키면 갱신이 무시됩니다. 디버그도 어렵습니다. 항상 Start 에서 모든 키를 사전 선언하시는 것이 표준입니다.*

> ***⚠ 함정 2 — 세션 간 공유 오해***
>
> *Flow State 는 세션 간에 공유되지 않습니다. 사용자 A 와 사용자 B 가 동시에 같은 워크플로를 실행하면 각자 독립된 `$flow.state` 를 가집니다. 영구 사용자 프로필이 필요하시면 외부 DB 를 사용하십시오.*

> ***⚠ 함정 3 — Iteration 안에서의 갱신***

> *Iteration 서브플로가 외부 `$flow.state.counter` 같은 키를 갱신하면 마지막 반복의 값만 남습니다 (덮어쓰기). 합산·누적이 필요하시면 Custom Function Node 로 명시 처리하시는 것이 안전합니다.*

```
// Iteration 의 마지막 반복 결과 합산 예시 (Custom Function)
const items = $flow.state.itemList;
let total = 0;
for (const item of items) {
  total += item.value;
}
return total.toString();
```

본 3가지 함정이 학습자가 운영 단계에서 만날 가장 자주 인용되는 사고 패턴입니다. 12장 학습 종료 시점에 본 3가지를 자기 언어로 한 줄씩 설명 가능하시다면 Flow State 완전 이해의 최종 도달점입니다.

13장: 5가지 표준 패턴 — 단일 LLM · Tool Agent · RAG · HITL · Multi-agent

이번 장에서는 14 노드 학습을 5 패턴으로 통합합니다. 본 5 패턴이 본 자료 입문 단계의 90% 를 차지하며, 학습자가 자기 사내 사례에 1:1 으로 대입하실 수 있는 표준 골격이 됩니다.

13.1 패턴 A·B·C — 단일 LLM · Tool Agent · RAG

13.1.1 패턴 A — 단일 LLM 챗봇 (Start → LLM → Direct Reply)

가장 단순한 3노드 워크플로입니다. 6장 Hello World 와 동일 골격이고, 본 자료 첫 실습이 본 패턴이었습니다.

> Start → LLM → Direct Reply

적합 시나리오	메모리
페르소나 챗봇 (사내 상담봇)	Buffer 또는 Window
요약봇 (텍스트 입력 → 요약 출력)	메모리 불필요
번역봇 (텍스트 입력 → 번역 출력)	메모리 불필요
FAQ 챗봇 (정해진 답변 응대)	Window (최근 N 턴)

> 💡 ***팁** — 6장에서 만드신 Hello World 가 본 패턴 A 의 가장 단순한 형태입니다. Memory 옵션만 Buffer 로 켜 주시면 대화 이력 누적이 가능한 챗봇이 됩니다.*

13.1.2 패턴 B·C — Tool Agent + RAG

패턴 B 는 도구 사용 에이전트입니다.

> Start → Agent (with Tools[*]) → Direct Reply

날씨 · 검색 · 계산 등 외부 정보가 필요한 챗봇에 적합합니다. Agent Node 의 `Tools` 슬롯에 등록된 도구들 중 LLM 이 자율 선택합니다.

패턴 C 는 RAG (사내 문서 기반 응답) 입니다.

> Start → Retriever → LLM (with context) → Direct Reply

Retriever 가 사용자 질문에 가까운 청크를 뽑고, LLM 이 그 청크를 컨텍스트로 받아 답변합니다.

항목	패턴 B (Tool Agent)	패턴 C (RAG)
핵심 부품	Agent Node	Retriever Node + LLM Node
동적 결정 주체	LLM (도구 자율 선택)	검색 결과 기반 응답 생성
적합 시나리오	외부 API 호출 (날씨·검색·DB)	사내 문서·매뉴얼·정책 기반 응답
출처 인용	trace 출력에 도구 호출 기록	Retriever 의 `Text with Metadata` 옵션

두 패턴 모두 본 자료 14장 marketplace 예제에서 실제로 import 해 보실 수 있습니다.

13.2 패턴 D·E — HITL · Multi-agent

13.2.1 패턴 D — HITL 승인 (`LLM → Human Input → {proceed | reject: Loop}`)

> Start → LLM (draft) → Human Input → {proceed: HTTP send | reject: Loop back to LLM}

본 패턴은 8.3.2 에서 이미 한 번 보셨습니다. AI 가 작성한 초안을 사람이 검수해 발송하거나 재작성하는 표준 HITL 패턴이며, 마케팅 이메일 · 결재 · 외부 송금 시나리오에 즉시 적용 가능합니다.

`reject` 경로의 Loop Node 가 LLM 단계로 회귀하면, LLM 이 사람의 피드백을 받아 다시 draft 를 생성합니다. `proceed` 경로의 HTTP Node 가 실제 외부 서비스를 호출합니다 (이메일 발송 · Slack 메시지 · 결재 시스템 등).

> ✅ ***체크포인트*** *— 본 패턴이 자동화 플랫폼과 V2 의 가장 큰 차별점입니다. Human Input 의 체크포인트 영속 덕분에 매니저가 휴가 중이어도 워크플로가 며칠이고 일시정지 상태로 대기할 수 있습니다.*

13.2.2 패턴 E — Supervisor-Worker Multi-agent

> Start → Condition Agent (intent 분류) →
> ├ refund 경로: Agent(refund tools) → Direct Reply
> ├ tech 경로: Agent(tech tools, Document Store) → Direct Reply
> └ etc 경로: LLM → Direct Reply

슈퍼바이저 역할의 Condition Agent 가 사용자 의도를 분류하고, 각 도메인 워커 에이전트로 위임합니다. Flow State 에 `userIntent`, `customerId` 같은 키를 두어 모든 분기에서 공유합니다.

본 패턴이 사내 고객 응대 챗봇의 표준이며, 14장 marketplace 의 Supervisor 예제와 정합합니다. Condition Agent 의 Scenarios 는 4~5개 이내가 권장됩니다 (10개 이상이면 LLM 분류 정확도가 급격히 떨어짐).

여기까지로 13장 5 패턴 학습이 끝났습니다. 다음 14장에서 본 5 패턴을 marketplace 예제 5종으로 실제 import 하고 변형해 보겠습니다.

14장: Marketplace 5예제 — import · 변형 실습

14.1 Marketplace JSON import 기본 흐름

본 장의 목표는 Flowise 공식 marketplace 의 입문 예제 5종을 import 해, 자기 환경 (모델·도구·프롬프트) 에 맞춰 변형하는 것입니다. Marketplace 예제 1개만 import 해서 모델 1개만 갈아 끼우셔도 이 교재의 실습 산출 1개를 확보하실 수 있습니다.

14.1.1 marketplace 메뉴 진입 + JSON 다운로드

> ***steps*** *— Flowise UI 의 Marketplace 메뉴 또는 GitHub 직접 다운로드 두 경로.*

좌측 사이드바의 `Marketplace` 메뉴를 클릭하시면 예제 카탈로그가 열립니다. 카테고리 드롭다운에서 `Agentflows` 를 선택하시면 V2 예제만 필터됩니다. 각 예제 카드를 클릭하시면 `Use Template` (즉시 import) 또는 `Download JSON` (파일 다운로드) 두 옵션이 노출됩니다.

UI 메뉴 외에 GitHub 에서 직접 다운로드도 가능합니다. 다음 URL 의 디렉터리에서 원하는 JSON 을 직접 받으실 수 있습니다.

> https://github.com/FlowiseAI/Flowise/tree/main/packages/server/marketplaces/agentflowsv2

> 💡 ***GitHub 직접 다운로드 시 commit hash 메모*** *— JSON 스키마가 마이너 버전마다 바뀔 수 있으므로, 다운로드 시점의 commit hash 를 메모해 두시면 추후 비교가 용이합니다.*

14.1.2 빈 캔버스에 JSON import + Credential 재매핑

> ***steps*** *— `Agentflows → Add New → 우측 상단 ... → Import` → JSON 파일 선택 → Credential 수동 재매핑.*

빈 캔버스 진입 후 우측 상단의 옵션 메뉴 (...) → `Import` 를 클릭하시면 파일 선택 다이얼로그가 열립니다. 다운로드한 JSON 을 선택하시면 즉시 노드 그래프가 캔버스에 펼쳐집니다.

> ✅ ***체크포인트 — Credential 매핑 확인***

> *import 직후 각 모델 노드 (LLM · Agent · Condition Agent 등) 의 Credential 드롭다운이 빈 상태입니다. JSON 안에는 Credential 매핑이 박혀 있지 않으므로, 5장에서 등록하신 `openai-default` 를 각 노드에 수동으로 재매핑해 주셔야 합니다. 재매핑 누락 시 첫 실행에서 모델 호출 에러가 발생합니다.*

본 절차는 모든 import 마다 반복되며, 한번 익숙해지시면 30초 안에 처리 가능합니다.

14.2 5예제 카탈로그 + 변형 포인트

본 절은 marketplace 입문 예제 5종을 한 표로 정리하고, 각 예제의 가장 단순한 변형 포인트를 제안합니다.

14.2.1 예제 1·2·3 — Memory 4종 비교 · RAG · Tool Agent

#	예제	핵심 노드 구성	변형 포인트	예상 소요
1	Memory 4종 비교 (E5 시리즈)	`Start → LLM (각 Memory) → Direct Reply` × 4	Memory 옵션 1종만 시도 (예: Window만)	30분
2	PDF Agentic RAG (E3)	`Start → Retriever (Qdrant) → Agent → Direct Reply`	Qdrant 1줄 (`docker run qdrant/qdrant`) + PDF 업로드	1시간
3	Tool Agent ReAct (E4)	`Start → Agent (Custom Tool) → Direct Reply`	Custom Tool 의 사내 REST URL 1줄 변경	40분

예제 1 (Memory 4종) 은 4개의 LLM 워크플로를 동일 10턴 대화로 테스트하면서 Buffer · Window · Summary · Vector 의 응답 차이를 직접 측정하시게 됩니다. 학습자는 처음 1개 (Buffer 또는 Window) 만 시도하셔도 본 자료 산출로 충분합니다.

예제 2 (PDF Agentic RAG) 는 사내 매뉴얼 PDF 1개를 Document Store 에 업로드해 RAG 챗봇을 만드는 시나리오입니다. Qdrant 외부 VectorStore 와 결합하시면 인스턴스 재시작에도 인덱스가 영속됩니다.

예제 3 (Tool Agent ReAct) 은 Custom Tool 노드에서 사내 REST API 1개를 호출하는 가장 단순한 Tool Agent 패턴입니다. 사내 사례와 가장 가까운 1개를 시작점으로 고르시는 것이 학습 효과 측면에서 좋습니다.

14.2.2 예제 4·5 — Route Agent · Supervisor

#	예제	핵심 노드 구성	변형 포인트	예상 소요
4	Route Agent (E6)	`Start → Condition Agent → {분기 3개: LLM/Agent} → Direct Reply`	분기 4개로 확장 (예: refund/tech/billing/general)	1시간
5	Supervisor & Workers (E7)	`Start → Condition Agent → {worker A/B/C: Agent + To`	워커 1개의 도구 슬롯에 사내 도구 1개 추가	1.5시간

#	예제	핵심 노드 구성	변형 포인트	예상 소요
		ols} → Direct Repl y		

예제 4 (Route Agent) 는 13.2 의 패턴 E 와 정합합니다. 분류 LLM 은 빠른 모델 (`gpt-4o-mini`) 로 두시고, 분기 후의 응답 생성 모델만 큰 모델로 분리하시는 것이 비용 대비 효과 측면에서 권장됩니다.

예제 5 (Supervisor & Workers) 는 멀티 에이전트 패턴의 가장 표준적인 구현입니다. 사내 고객 응대 챗봇의 1차 골격으로 그대로 적용 가능합니다.

> 💡 ***Supervisor 작동 검증 팁*** *— 예제 5 의 Condition Agent 의 trace 출력을 1번 확인해 보십시오. 의도 분류 결과와 위임된 워커가 일치하는지 점검하시면 분류 정확도를 정량적으로 측정하실 수 있습니다.*

본 5예제 중 1개만 import 해서 변형해 보셔도 이 교재의 모든 학습이 사내 시스템과 연결되는 첫 산출이 됩니다.

15장: FAQ · Troubleshooting · 자가 진단 · 다음 단계

15.1 FAQ Top 10 · Troubleshooting Top 5

15.1.1 FAQ Top 10

#	질문	한 줄 답변
Q1	V1 Chatflow 를 그대로 두고 V2 로 가도 되나요	가능. 사이드바 메뉴 수준에서 분리·공존. 신규 워크플로만 V2 로 권장
Q2	Agent Node 와 LLM Node 를 어떻게 구별하나요	도구 호출 여부 단일 기준. 도구 필요하면 Agent, 그렇지 않으면 LLM
Q3	Loop 와 Iteration 의 차이는	Iteration = 배열 for-each (각 원소 1회), Loop = 조건 만족 시까지 회귀 (while)
Q4	Human Input 노드는 동기 차단인가요	아니오. 워크플로는 영속 일시정지, Flowise 프로세스는 다른 요청을 계속 처리
Q5	Flow State 는 어디까지 영속되나요	1회 실행 동안만. 영구는 외부 DB
Q6	MCP Tools 는 어떻게 등록하나요	좌측 `Tools` 메뉴에서 등록 → Agent/Tool 노드 드롭다운에 자동 노출
Q7	이 교재의 이미지가 보이지 않으면	`images_agentflowv2/` 폴더가 같이 옮겨졌는지 확인. 상대 경로 기반

#	질문	한 줄 답변
		이라 폴더만 있으면 정상
Q8	Marketplace JSON 의 스키마 호환은	마이너 버전마다 미세 변경 가능. 다운로드 시점 commit hash 메모 권장
Q9	사내 LLM 자격증명이 자꾸 실패합니다	BaseURL 끝에 `/v1` 누락 확인 (가장 흔한 원인)
Q10	첫 응답이 안 나옵니다	Direct Reply 누락 / 엣지 누락 / Credential 미선택 3가지 점검

본 10개 질문이 학습자의 빈도순 상위 사고 매트릭스입니다. 본 표를 PDF 부록 페이지로 출력하셔서 책상 옆에 두시면 자가 진단 시간이 절반으로 줄어듭니다.

15.1.2 Troubleshooting Top 5

#	증상	해결 1단계
T1	첫 응답이 안 나옴	Direct Reply Node 배치 + 엣지 연결 확인
T2	모델 호출 에러 (401 / 403)	LLM/Agent 노드의 Credential 드롭다운에서 자격증명 선택
T3	사내 LLM 호출 404	BaseURL 끝에 `/v1` 정확히 붙였는지 확인
T4	`$flow.state.x` 값이 갱신 안 됨	Start Node 의 Flow State 패널에서 `x` 키 사전 선언
T5	Agent 가 도구를 자꾸 헷갈림	Tools 슬롯의 도구 개수 5~7개로 축소 + 도구 설명 차별화

본 5가지가 학습자의 첫 1주 사고 95% 를 차지합니다. 본 표만으로도 대부분의 운영 단계 문제가 해결됩니다.

15.2 자가 진단 체크리스트 + 다음 학습 단계

15.2.1 5 질문 자가 진단 체크리스트

> ✅ ***본 자료 1회독 후 자가 진단 — 5개 모두 답할 수 있으시면 본 자료 완주***
>
> - *[ ] **Q1** — Flowise 의 한 줄 정의를 자기 언어로 발화 가능하시다 (FlowiseAI Inc. + LangChain.js + GUI 캔버스 + Apache 2.0)*
> - *[ ] **Q2** — Workflow 형 도구가 Code Framework 와 자동화 플랫폼 사이의 어떤 골을 메우는지 5가지 이유로 설명 가능하시다*

> - *[ ] **Q3** — `Start → LLM → Direct Reply` 3노드 워크플로 1개를 처음부터 손으로 만들어 첫 응답을 받으실 수 있다*
> - *[ ] **Q4** — 14개 표준 노드를 5필드 포맷 (정의 · 언제 · 입력 · 출력 · 흔한 실수) 으로 자기 언어로 설명 가능하시다*
> - *[ ] **Q5** — Flow State 의 초기화·읽기·갱신·함정 3가지를 설명 가능하시고, Marketplace 예제 5종 중 ≥ 1개를 import·변형해 본 산출이 있으시다*

5개 모두 체크하시면 본 자료 1주차 학습 목표를 100% 달성하신 것입니다. 4개 이하시면 부족한 영역의 해당 장으로 한 번 더 돌아가셔서 보강하시면 됩니다.

15.2.2 다음 학습 단계 + 자매 자료 안내

이 교재를 완주하셨다면 다음 학습 단계로 자연스럽게 진입하실 수 있습니다.

다음 단계	자료	다루는 영역
1주차 보강	본 자료 Ch 8~12 재정독	14 노드 5필드 + Flow State
2주차 (Intermediate)	Flowise Intermediate 자매 자료	외부 VectorStore 운영 · MCP 도구 자체 개발 · Embed 위젯 · Public API
3주차 (Advanced)	Flowise Advanced 자매 자료	자체 노드 개발 · 멀티 테넌트 운영 · 권한 관리

> 💡 ***다음 단계 결정 가이드** — 자가 진단 체크리스트의 5개 모두 통과하셨다면 2주차로 진입 권장. 4개 이하라면 본 자료 1주차 보강 후 진입. 사내 운영 단계 진입 시 외부 VectorStore 와 MCP 도구 자체 개발이 가장 자주 필요해지므로 2주차의 1·2장이 최우선입니다.*

MSAP.ai 의 AI Agent 워크플로 운영 모듈은 이 교재의 학습 단계 다음으로 사내 PoC 가 운영으로 진입하실 때 평가하실 수 있는 후속 옵션 중 하나입니다. 이 교재의 학습 자체는 어떤 운영 모듈 선택과도 독립적으로 가치를 제공합니다.

여기까지로 본 자료 15장 전체 학습이 끝났습니다. 1주 동안 이 교재와 함께 손으로 실습하신 학습자께서는 사내 AI Agent 도입 평가 · PoC 설계 · 운영 이관의 첫 3단계에서 자기 언어로 의견을 내실 수 있는 단계에 도달하셨습니다.

Appendix A: 14 노드 한 줄 요약 치트시트

A.1 14 노드 cheatsheet 표

A.1.1 14 노드 한 줄 요약 + 상세 장 매핑

#	노드	한 줄 정의	상세 장
1	Start	워크플로 진입점, Flow State 키 선언	8장
2	LLM	도구 없는 단일 LLM 호출	8장
3	Agent	도구를 자율 선택하는 ReAct 에이전트	8장
4	Tool	지정된 단일 도구를 결정적으로 호출	9장
5	Retriever	Document Store 에서 의미 검색	9장
6	HTTP	범용 REST API 호출	9장
7	Condition	규칙 기반 결정적 분기	10장
8	Condition Agent	LLM 기반 동적 분기 (자연어 시나리오)	10장
9	Iteration	배열 for-each 반복	10장
10	Loop	이전 노드로 회귀 (while 루프)	10장
11	Human Input	워크플로 일시정지 · HITL 체크포인트	8장
12	Direct Reply	최종 응답 전송, 분기 종료	8장
13	Custom Function	임의 JavaScript 실행	11장
14	Execute Flow	다른 Flowise 워크플로를 모듈로 호출	11장

본 표를 PDF 페이지 1장으로 인쇄하셔서 책상 옆에 두시면 학습 내내 가장 자주 참조하시게 됩니다.

A.1.2 5필드 포맷 셀프 진단 체크박스

본 자료 8~11장 학습 후 14 노드 각각의 5필드를 자기 언어로 발화 가능한지 체크박스로 자가 진단해 보십시오. 14 × 5 = 70 체크박스 중 50 이상 통과하시면 본 자료 완주의 정량 기준입니다.

노드	정의	언제	입력	출력	실수
Start	☐	☐	☐	☐	☐
LLM	☐	☐	☐	☐	☐
Agent	☐	☐	☐	☐	☐

노드	정의	언제	입력	출력	실수
Tool	☐	☐	☐	☐	☐
Retriever	☐	☐	☐	☐	☐
HTTP	☐	☐	☐	☐	☐
Condition	☐	☐	☐	☐	☐
Condition Agent	☐	☐	☐	☐	☐
Iteration	☐	☐	☐	☐	☐
Loop	☐	☐	☐	☐	☐
Human Input	☐	☐	☐	☐	☐
Direct Reply	☐	☐	☐	☐	☐
Custom Function	☐	☐	☐	☐	☐
Execute Flow	☐	☐	☐	☐	☐

Appendix B: v1 → v2 마이그레이션 매핑표

B.1 v1 노드 ↔ v2 매핑 + 신설 노드 4개

B.1.1 v1 ↔ v2 매핑 표

사내에서 v1 Chatflow 를 운영 중이시고 v2 로 점진 마이그레이션을 검토하시는 경우의 1:1 노드 매핑입니다.

V1 Chatflow 노드	V2 AgentFlow 대응	비고
Conversational Retrieval QA Chain	`Retriever` + `LLM`	RAG 패턴 명시적 분해
OpenAI Function Agent	`Agent` (with Tools)	도구 슬롯으로 통합
LangChain ReAct Agent	`Agent` (with Tools)	동일 패턴, 노드명 변경
Buffer Memory	`LLM` / `Agent` 의 Memory=Buffer	노드 옵션으로 흡수
Window Memory	`LLM` / `Agent` 의 Memory=Window	노드 옵션으로 흡수
Summary Memory	`LLM` / `Agent` 의 Memory=Summary	노드 옵션으로 흡수
Vector Store Retriever	`Retriever`	옵션 통합

V1 Chatflow 노드	V2 AgentFlow 대응	비고
Custom Tool	`Tool` 또는 `Agent.Tools`	동일 도구를 두 노드 모두에서 사용 가능

> 💡 ***마이그레이션 권장 방식*** *— 1:1 매핑보다* ***패턴 단위 재설계*** *가 권장됩니다. V2 의 표현력이 더 풍부하므로, V1 에서 "에이전트 내부 마법" 으로 처리하던 흐름을 캔버스 상에 노드로 펼치시는 것이 가독성·유지보수성·디버깅 모두에 유리합니다.*

B.1.2 v2 에 신설된 노드 4개

v1 에서 우회 패턴으로만 표현 가능했던 4개 흐름이 v2 에서 1급 노드로 신설되었습니다.

신설 v2 노드	v1 시기 우회 패턴	v2 노드의 가치
`Loop`	에이전트 내부 추론 루프 또는 외부 큐	while 루프 캔버스 가시화
`Iteration`	Custom Code 의 forEach	for-each 캔버스 가시화
`Condition Agent`	LLM Node + 다음 단계 텍스트 파싱	자연어 의도 분류 1급 시민
`Human Input`	외부 큐 + Slack 알림 우회	체크포인트 영속 일시정지
`Execute Flow`	별도 워크플로 + 외부 API 호출	모듈 호출 1급 시민

신설 5개 노드가 v2 의 표현력 확장 핵심입니다. v1 에서 우회 패턴으로 운영 중이시던 시나리오를 v2 로 옮기시면 캔버스 단순화 + 디버깅 가시성 + 거버넌스 측면 모두에서 즉각적인 이득이 있습니다.

Appendix C: References

C.1 1차 자료

C.1.1 로컬 1차 자료 4종

이 교재의 본문 인용 표면의 95% 가 본 4종 자료입니다. 본 자료 ZIP 다운로드 시 본 4종이 동일 디렉터리에 동봉됩니다.

ID	파일명	위치	인용 범위
S1	Flowise Agent flow v2 소개.md	동일 디렉터리	14 노드 5필드 + 5 패턴 + Flow State
S2	Getting Started — Flowise & AgentFlow V2 이해 (Basic).md	동일 디렉터리	1·2장 정체·5가지 이유 + Hello World + marketplace 5예제

ID	파일명	위치	인용 범위
S3	Flowise Getting Started.md	동일 디렉터리	설치 · 자격증명 · Memory 4종 · RAG · ReAct · Route · Supervisor
S4	images_agentflowv2/ (30개 PNG/GIF)	동일 디렉터리	14 노드 캡처 + 차별점 7개 + Flow State 6개 + 표지 3개

C.1.2 이미지 자산 30개 카탈로그

이 교재가 사용하는 모든 이미지는 `images_agentflowv2/` 폴더에 다운로드되어 있습니다. 본문 임베드 위치는 다음과 같습니다.

파일명	본 자료 어느 장
`hero-collage.png`	표지
`diff-agent-to-agent.png`	3.2
`diff-hitl-1.png` · `diff-hitl-2.png`	3.2, 8.3.2
`diff-shared-state.png`	3.2
`diff-streaming.gif`	2.3, 3.2
`diff-mcp-1.png` · `diff-mcp-2.png`	2.3, 9.1.2
`node-overview.png` · `node-overview-dark.png`	7.1
`flowstate-overview.png` · `flowstate-init.png`	12.1, 12.2
`flowstate-update-1.png` · `flowstate-update-2.png`	12.2
`flowstate-read-1.png` · `flowstate-read-2.png`	12.3
`node-01-start.png` ~ `node-14-execute-flow.png`	8~11장

C.2 2차 자료 — 외부 URL 카탈로그

C.2.1 Flowise 공식 docs · GitHub · Marketplace

학습자가 본 자료 너머로 진입하실 때 가장 자주 인용되는 출구 3 URL 입니다.

ID	URL	액세스 일자	라이선스
S5	https://docs.flowiseai.com/using-flowise/agentflowv2	2026-05-26	(벤더 공식)

ID	URL	액세스 일자	라이선스
S6	https://github.com/FlowiseAI/Flowise	2026-05-26	Apache 2.0
S7	https://github.com/FlowiseAI/Flowise/tree/main/packages/server/marketplaces/agentflowsv2	2026-05-26	Apache 2.0

C.2.2 MCP · LangChain.js · Qdrant · SSE MDN

이 교재의 깊이 학습을 위한 보조 출구 4 URL 입니다.

ID	URL	액세스 일자	라이선스
S8	https://modelcontextprotocol.io/	2026-05-26	(Anthropic 공식)
S9	https://js.langchain.com/docs/	2026-05-26	(LangChain 공식)
S10	https://qdrant.tech/documentation/	2026-05-26	(Qdrant 공식)
S11	https://developer.mozilla.org/docs/Web/API/Server-sent_events	2026-05-26	CC-BY-SA

<!-- LINT_RUN: honorific_consistency=0.96, writer_self_declarative=0, reader_density=8.1/1000, reader_explicit_count=57, reader_addressing_extended=137, phase1_block_vocab=0, forbidden_analyst_vocab=0, c10_signal=1.00, c2_total_chars=107237, c2_total_lines=1586, c2_total_words_estimate=24000, image_embeds=33, figure_marker_residual=0, external_links=21, h2_count=19, h3_count=44, h4_count=88, h4_to_h2_ratio=4.63, ts=2026-05-26T17:35:00Z -->

처음 새 도구를 만나실 때 가장 답답한 것은 "내가 이걸 다 읽고 나면 무엇을 할 수

CONTACT

WEB
msap.ai
www.msap.ai/

EMAIL
hello@msap.ai

TEL
02-6953-5427

YOUTUBE
@msaptv
www.youtube.com/@msaptv

LINKEDIN
linkedin.com/showcas...
www.linkedin.com/showcase/msap-ai/

FACEBOOK
facebook.com/opennaru
www.facebook.com/opennaru

SCAN