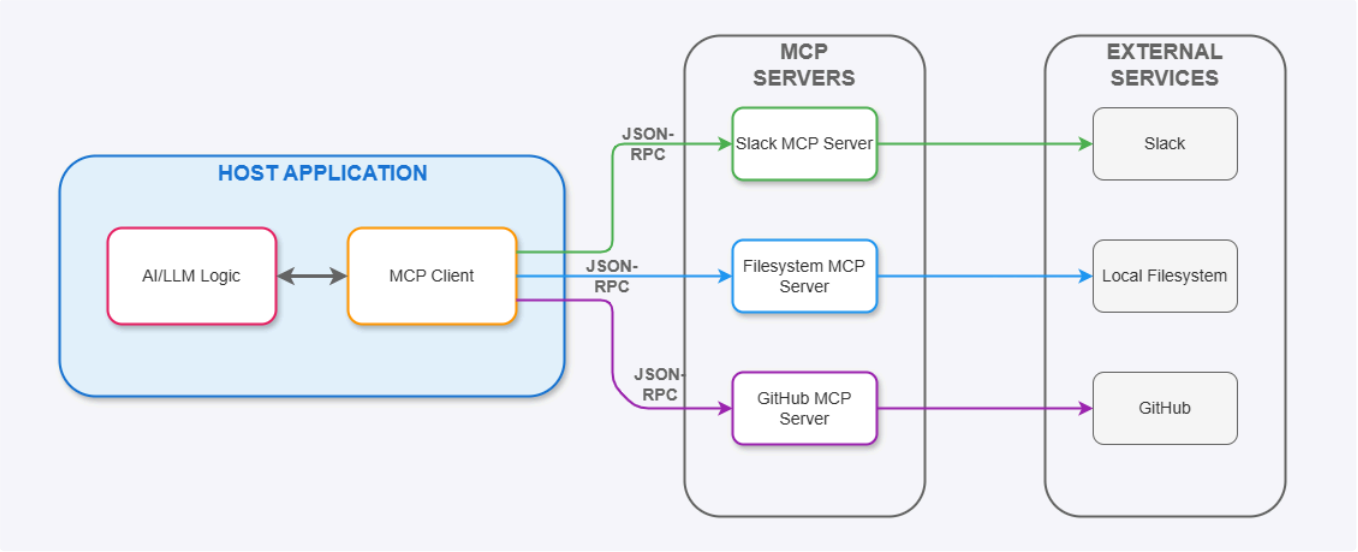
Chapter 4. MCP 전환

Chapter 4. MCP로 전환하기

> ⓘ 이 챕터에서는...
>
> Chapter 2에서 만든 Custom Component를 **Model Context Protocol (MCP)** 서버로 전환합니다.
> 같은 기능을 MCP 표준으로 제공하여 Langflow뿐만 아니라 Claude Desktop, Cline 등 다양한 AI 도구에서 재사용할 수 있습니다.

---

1. MCP란? (복습)



Model Context Protocol은 AI 모델이 외부 도구와 통신하는 **표준 프로토콜**입니다.

왜 MCP를 써야 할까?

Before (Custom Component):

```
GDELT Component → Langflow 전용
NewsExtractor Component → Langflow 전용
```

- ❌ Langflow에서만 사용 가능
- ❌ Claude Desktop에서 쓰려면 다시 작성
- ❌ 각 도구마다 다른 인터페이스

After (MCP Server):

```
MCP Server → Langflow, Claude Desktop, Cline, 모두 사용 가능
```

- ✅ 한 번 작성, 어디서나 사용
- ✅ 표준 프로토콜로 호환성 보장
- ✅ 코드 재사용성 극대화

---

2. 프로젝트 구조 확인

```
cd ~/dev-root/ai-agent-mcp/llm-agent-workshop
git checkout 02-news-agent-with-mcp
```

프로젝트 구조

```
llm-agent-workshop/
├── core_services/              # 📦 공통 비즈니스 로직 (DRY!)
```

```
│   ├── gdelt_service.py        # GDELT API 핵심 로직
│   └── content_extractor_service.py  # 본문 추출 핵심 로직
│
├── custom_components/          # 🎨 Langflow 전용 컴포넌트
│   ├── gdelt_doc_search_component.py  # Original
│   ├── gdelt_doc_search_component_with_core.py  # core_services 사용
│   ├── news_content_extractor.py  # Original
│   └── news_content_extractor_with_core.py  # core_services 사용
│
└── mcp_news_server.py          # 🚀 MCP 서버 (core_services 사용)
```

🔥 핵심 설계 원칙: **DRY (Don't Repeat Yourself)**

- `core_services` : 순수한 비즈니스 로직 (Python 함수)
- `custom_components` : Langflow 래퍼
- `mcp_news_server.py` : MCP 래퍼

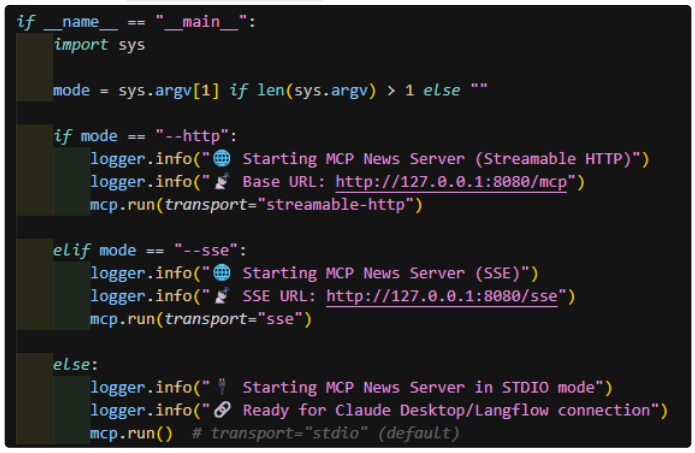
같은 로직을 세 번 작성하지 않고, 한 곳(`core_services`)에만 작성!

---

3. MCP 서버 살펴보기

3-1. 코드 구조

VS Code에서 `mcp_news_server.py` 열기:



```python
if __name__ == "__main__":
    import sys

    mode = sys.argv[1] if len(sys.argv) > 1 else ""

    if mode == "--http":
        logger.info("🌐 Starting MCP News Server (Streamable HTTP)")
        logger.info("📡 Base URL: http://127.0.0.1:8080/mcp")
        mcp.run(transport="streamable-http")

    elif mode == "--sse":
        logger.info("🌐 Starting MCP News Server (SSE)")
        logger.info("📡 SSE URL: http://127.0.0.1:8080/sse")
        mcp.run(transport="sse")

    else:
        logger.info("🖥️ Starting MCP News Server in STDIO mode")
        logger.info("🔗 Ready for Claude Desktop/Langflow connection")
        mcp.run()  # transport="stdio" (default)
```

```python
from mcp.server.fastmcp import FastMCP
from core_services.gdelt_service import GDELTService
from core_services.content_extractor_service import ContentExtractorService

# MCP 서버 초기화
mcp = FastMCP(name="news-research")

@mcp.tool()
async def search_gdelt_news(
    query: str,
    max_results: int = 10,
    # ... 파라미터들
) -> str:
    """GDELT 뉴스 검색"""
    df = GDELTService.search_news(...)  # ← core_services 사용!
    return format_results(df)

@mcp.tool()
async def extract_article_content(
    urls: str,
    max_length: int = 5000
) -> str:
    """기사 본문 추출"""
    results = ContentExtractorService.extract_content(...)  # ← core_services 사용!
    return format_results(results)
```

✓ 포인트

- `@mcp.tool()` : MCP 도구로 등록
- `core_services` 호출: 실제 로직은 재사용
- `async def` : MCP는 비동기 처리

---

4. Transport 방식 이해하기

MCP는 두 가지 통신 방식을 지원합니다:

Transport 비교

항목	SSE	STDIO
연결 방식	HTTP (포트 8080)	표준 입출력 (파이프)
서버 실행	수동 시작 필요	클라이언트가 자동 시작
로그 확인	터미널에 실시간 출력	Langflow 로그에 섞임
상태	Legacy (하위 호환용)	표준 방식

ⓘ SSE vs STDIO 선택 기준

- **SSE**: 서버를 띄워두고 여러 클라이언트가 접속하는 경우, 실시간 로그 디버깅이 필요한 경우
- **STDIO**: 데스크톱 앱(Claude Desktop 등), 로컬 도구 통합, 클라이언트가 서버 생명주기를 관리하는 경우

참고: FastMCP 문서에 따르면 SSE는 "legacy transport"입니다. 프로덕션 배포 시에는 Streamable HTTP 사용이 권장되지만, 로컬 개발과 디버깅에는 여전히 유용합니다.

---

5. 실습: STDIO 모드로 실행하기

STDIO 모드는 가장 간단한 방식입니다. 서버를 수동으로 실행하지 않아도 Langflow가 자동으로 관리합니다.

5-1. MCP Server 설정 JSON

```
{
  "news-research-stdio": {
    "command": "uv",
    "args": [
      "run",
      "python",
      "mcp_news_server.py"
    ],
    "transport": "stdio"
  }
}
```

상대 경로 사용 가능

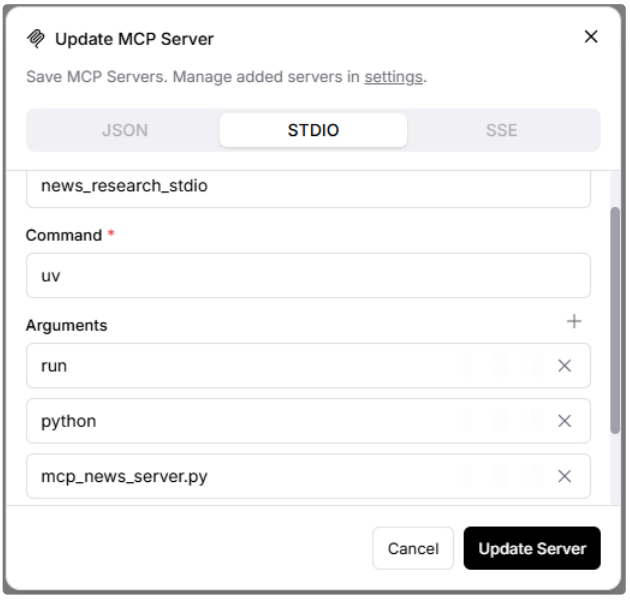
Langflow를 프로젝트 루트에서 실행 중이라면 `args` 에 절대 경로 불필요!

5-2. Langflow에서 설정

1. Langflow 열기 (`http://localhost:7860`)
2. **Settings** (⚙) → **MCP Servers** 클릭
3. **Import from JSON** 버튼 클릭
4. 위의 JSON 붙여넣기 → **Save**

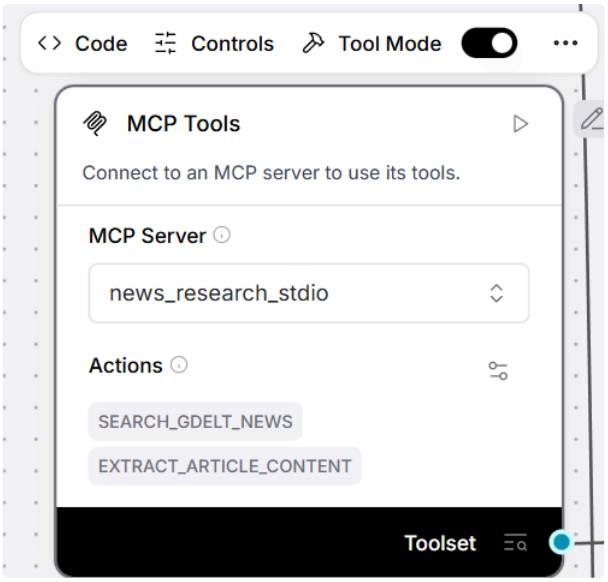
5-3. MCP Tools 컴포넌트 추가

1. 새 플로우 생성 또는 기존 플로우 열기
2. 컴포넌트 검색: **MCP Tools**
3. 드래그하여 캔버스에 추가
4. **MCP Server** 드롭다운에서 `news-research-stdio` 선택
5. **Tools** 섹션에 도구 목록 자동 로드 확인:
   - ✅ `search_gdelt_news`
   - ✅ `extract_article_content`



5-4. Agent와 연결하여 테스트

기존 Agent 플로우에 MCP Tools 연결:



테스트 쿼리:

```
"Samsung SDS 종목 분석하려고"
```

실행 결과:

- ✅ 서버를 수동으로 띄우지 않아도 자동 실행
- ✅ Langflow가 서버 시작/종료 자동 관리
- ✅ Agent가 GDELT 검색 → 본문 추출 자동 수행

> ✓ STDIO 모드의 장점
>
> 별도 서버 관리 없이 바로 사용 가능! 가장 간편한 방식입니다.

---

6. 실습: SSE 모드로 실행하기

SSE 모드는 서버를 직접 실행하여 **실시간 로그**를 확인할 수 있는 방식입니다. 디버깅과 강의 시연에 유용합니다.

6-1. MCP 서버 실행

터미널 1 (WSL):

```
cd ~/dev-root/ai-agent-mcp/llm-agent-workshop
uv run python mcp_news_server.py --sse
```

예상 출력:

```
2025-11-23 ... - INFO - 🌐 Starting MCP News Server (SSE)
2025-11-23 ... - INFO - 📡 SSE URL: http://127.0.0.1:8080/sse
```

6-2. 서버 동작 확인

터미널 2 (WSL):

```
curl http://127.0.0.1:8080/sse
```

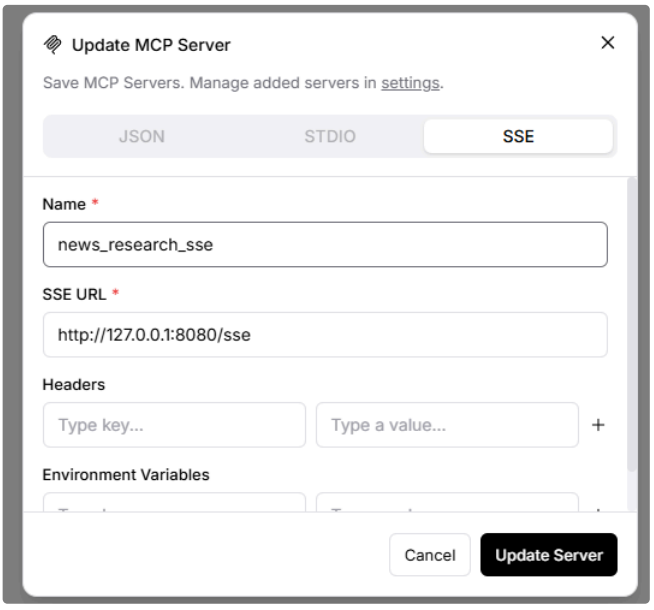
응답이 오면 성공!

6-3. Langflow에서 설정

1. Langflow에서 **Settings** → **MCP Servers**
2. **Import from JSON** 클릭
3. 다음 JSON 붙여넣기:

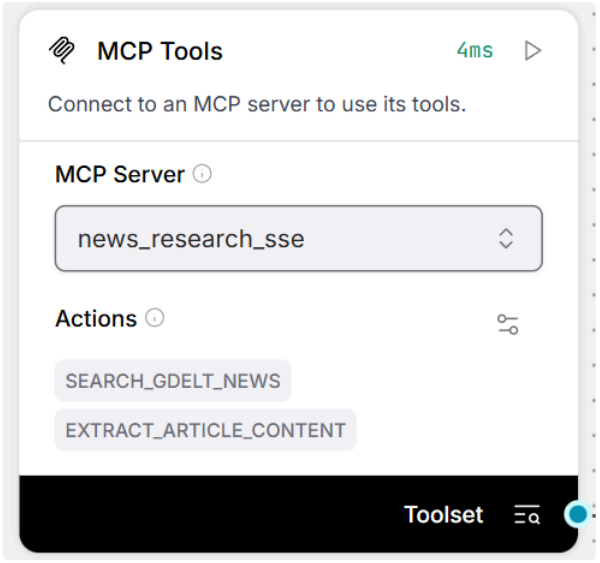
```
{
  "news-research-sse": {
    "url": "http://127.0.0.1:8080/sse",
    "transport": "sse"
  }
}
```

4. **Save** 클릭



6-4. MCP Tools 컴포넌트에서 SSE 서버 선택

1. MCP Tools 컴포넌트의 **MCP Server** 드롭다운
2. `news-research-sse` 선택
3. 도구 목록 표시 확인



6-5. Agent 테스트 및 실시간 로그 관찰

테스트 쿼리:

```
"Samsung SDS 종목 분석하려고"
```

터미널 1에서 실시간 로그 확인:

```
2025-11-23 ... - INFO - Searching GDELT for: Samsung SDS
2025-11-23 ... - INFO - Found 10 articles
2025-11-23 ... - INFO - Extracting content from: https://...
2025-11-23 ... - INFO - Extracted 3 articles successfully
```

"Agent 실행 로그" 이(가) 아직 생성되지 않았습니다. 클릭해서 생성해보세요.

✓ SSE 모드의 장점

터미널 로그를 보면서 Agent가 어떤 도구를 어떻게 사용하는지 실시간으로 확인 가능!
디버깅과 교육 목적으로 최적입니다.

---

7. STDIO 모드 주의사항 ⚠️

표준 입출력 오염 금지!

STDIO는 `stdin` / `stdout` 으로 JSON-RPC 메시지를 주고받습니다:

```
Langflow <--stdin/stdout (JSON-RPC)--> MCP Server
          <--stderr (로그)----------
```

❌ **절대 금지:**

```
print("디버그 메시지")  # stdout 오염 → 통신 깨짐!
```

✅ **안전한 방법:**

```
logger.info("디버그 메시지")  # stderr로 출력
```

"STDIO 통신 다이어그램" 이(가) 아직 생성되지 않았습니다. 클릭해서 생성해보세요.

우리 코드(`mcp_news_server.py`)는 이미 안전하게 설정되어 있습니다:

```
logging.basicConfig(
    level=logging.DEBUG,
    handlers=[
        logging.StreamHandler()  # sys.stderr로 출력 (기본값)
    ]
)
```

---

8. STDIO vs SSE: 언제 뭘 쓸까?

상황별 추천

상황	추천	이유
🚀 **빠른 로컬 테스트**	STDIO	서버 띄울 필요 없음, 바로 시작
🖥️ **데스크톱 앱 연동**	STDIO	Claude Desktop, Cline 등 표준 방식
🔧 **개발/디버깅**	SSE	실시간 로그 확인 가능
🎓 **강의 시연**	SSE	로그 보면서 설명하기 좋음
🌐 **다중 클라이언트**	SSE	한 서버에 여러 클라이언트 동시 접속

실전 팁

- **처음 시작**: STDIO로 간편하게
- **디버깅/개발**: SSE로 로그 관찰
- **교육용 시연**: SSE가 흐름 보여주기 좋음
- **데스크톱 도구 통합**: STDIO 사용 (Claude Desktop, Cline 등)

---

9. 정리 및 다음 단계

이번 챕터에서 배운 것

✅ MCP의 개념과 장점 이해
✅ `core_services` 기반 코드 재사용 (DRY 원칙)
✅ MCP 서버 실행 (STDIO/SSE 두 가지 방식)
✅ Langflow에서 MCP Tools 연결
✅ STDIO 모드의 주의사항 (stdout 오염 금지)

핵심 아키텍처

실습 과제 💪

1. **STDIO 모드로 다양한 쿼리 테스트**
   - "NVIDIA 최근 뉴스"
   - "Apple 실적 관련 기사"
   - 서버 자동 시작/종료 확인
2. **SSE 모드로 전환하여 로그 관찰**
   - 같은 쿼리로 테스트
   - 터미널에서 어떤 로그가 출력되는지 확인
   - Agent의 도구 사용 흐름 파악
3. **(도전) Claude Desktop 연결해보기**
   - `claude_desktop_config.json` 설정
   - Claude에서 자연어로 뉴스 검색해보기
   - STDIO 방식으로 연결

✓ 축하합니다! 🎉

이제 여러분은 Langflow Custom Component를 MCP 표준 서버로 전환했습니다.
같은 코드를 다양한 AI 도구에서 재사용할 수 있는 진정한 "도구 제작자"가 되셨습니다!