아래는 로컬 개발 환경과 Streamlit Cloud 배포 환경에서 발생할 수 있는 문제들을 항목별로 자세하게 설명한 내용입니다.

**로컬 개발 환경**

**1. 가상환경 및 필수 도구 설치**

* **가상환경 활성화:**
  + **설명:** 로컬에서 작업할 때, 프로젝트 별로 격리된 환경(venv)을 사용하면 패키지 버전 충돌이나 의존성 문제를 방지할 수 있습니다.
  + **확인 명령어:**
  + which python
  + echo $VIRTUAL\_ENV

위 명령어들을 사용해 현재 활성화된 가상환경 경로와 Python 인터프리터 위치를 확인합니다.

* + **필수 패키지 설치:**
    - 설치되어야 하는 패키지:
      * mcp: MCP 서버/클라이언트 기능 제공
      * openai-agents: 에이전트 관련 기능 제공
      * streamlit: 앱 UI 제공
      * youtube-transcript-api: 유튜브 자막 추출
      * python-dotenv: .env 파일로부터 환경 변수 로드
      * requests: HTTP 요청 처리
      * 기타 비동기 관련 패키지(nest-asyncio 등)
    - **확인 방법:**
    - pip list
    - **권장 사항:**  
      프로젝트 실행에 필요한 최소한의 패키지를 requirements.txt 파일에 포함시키고,
    - pip install -r requirements.txt

로 재설치할 수 있도록 관리합니다.

**2. 환경 변수 설정**

* **시스템 환경 변수 사용:**
  + **설명:** 로컬에서는 .env 파일을 사용하거나 직접 터미널에서 환경 변수를 설정할 수 있습니다.  
    예를 들어:
  + export YOUTUBE\_API\_KEY="your\_actual\_youtube\_api\_key"
  + export OPENAI\_API\_KEY="your\_actual\_openai\_api\_key"
  + **코드 확인:**  
    코드에서 os.getenv("YOUTUBE\_API\_KEY")와 os.getenv("OPENAI\_API\_KEY")가 올바른 값을 불러오는지 확인합니다.
  + **확인 방법:**
  + echo $YOUTUBE\_API\_KEY
  + echo $OPENAI\_API\_KEY

**3. 경로 오류 및 파일 존재 여부 확인**

* **mcp.json 설정:**
  + **설명:** 로컬에서는 Windows 경로를 사용할 수 있으며, 2\_mcp\_server.py 파일이 정확한 위치에 있어야 합니다.  
    예를 들어:
  + {
  + "mcpServers": {
  + "mcp-test": {
  + "command": "C:\\mcp-agent-phoenix\\python\_mcp\_agent-main\\venv\\Scripts\\python.exe",
  + "args": [
  + "C:\\mcp-agent-phoenix\\python\_mcp\_agent-main\\2\_mcp\_server.py"
  + ]
  + }
  + }
  + }
  + **확인 명령어:**
  + ls -l "C:/mcp-agent-phoenix/python\_mcp\_agent-main/2\_mcp\_server.py"

Git Bash에서 파일이 있는지와 올바른 권한이 있는지 확인합니다.

**4. MCP 서버와 에이전트 간 연결 점검**

* **디버깅 로그 추가:**
  + **설명:** 4\_mcp\_client.py의 setup\_mcp\_servers() 함수에 디버깅 로그를 추가하여 MCP 서버와의 연결 상태를 확인합니다.
  + **예시 코드:**
  + print(f"DEBUG: Attempting to connect to MCP server: {server\_name}")
  + # MCP 서버 연결 시도
  + await mcp\_server.connect()
  + print(f"DEBUG: Successfully connected to MCP server: {server\_name}")
* **이벤트 스트림 로그 확인:**
  + **설명:** async for event in result.stream\_events(): 구문에 로그를 추가하여 이벤트가 제대로 도착하는지 확인합니다.
  + **예시 코드:**
  + async for event in result.stream\_events():
  + print("DEBUG: Received event:", event)
* **개별 구성요소 테스트:**
  + **설명:** 3\_openai\_agents\_basics.py 파일 등을 사용해 동기 방식(Runner.run\_sync 등)으로 모델 응답을 테스트하여, 에이전트가 정상적으로 응답을 생성하는지 확인합니다.

**5. 코드 문법 오류 및 오타 수정**

* **오류 확인:**
  + **설명:** 에디터나 Git 상태(git status)를 통해 파일이 붉게 표시되는 경우 문법 오류나 오타가 있을 수 있습니다.
  + **예시:** 파일 첫 줄에 "pyfrom" 대신 "from" 등 오타가 있는 경우 수정합니다.
* **Lint 도구 활용:**
  + Pylint, flake8 등 도구를 사용해 코드 스타일과 문법 오류를 자동으로 점검할 수 있습니다.

**6. API 호출 테스트**

* **테스트 방법:**
  + **설명:** 터미널에서 curl 명령어를 통해 유튜브 API 호출을 테스트합니다.
  + **예시 명령어:**
  + curl "https://www.googleapis.com/youtube/v3/search?part=snippet&q=test&key=YOUR\_ACTUAL\_API\_KEY"
  + **목적:** API 키가 올바르게 작동하는지, 네트워크 연결에 문제가 없는지 확인합니다.

**Streamlit Cloud (클라우드 배포 환경)**

**1. 환경 변수 (Secrets) 설정**

* **Secrets 기능 사용:**
  + **설명:** Streamlit Cloud에서는 GitHub에 .env 파일을 업로드하지 않고, 안전하게 Secrets 페이지를 통해 API 키를 관리합니다.
  + **설정 방법:**
    1. Streamlit Cloud 대시보드에서 앱을 선택합니다.
    2. 'Manage app'을 클릭한 후 'Settings' 또는 'Configuration' 탭으로 이동합니다.
    3. 'Secrets' 섹션에서 아래와 같이 키-값 쌍으로 추가합니다.
    4. YOUTUBE\_API\_KEY = "your\_actual\_youtube\_api\_key"
    5. OPENAI\_API\_KEY = "your\_actual\_openai\_api\_key"
  + **코드 연결:**  
    코드에서는 os.getenv("YOUTUBE\_API\_KEY")를 사용하면 Secrets에 저장된 값이 자동으로 로드됩니다.

**2. mcp.json 설정 변경**

* **배포 환경에 맞게 경로 수정:**
  + **설명:** 배포 환경은 보통 Linux 기반이므로, Windows 전용 경로 대신 상대 경로와 기본 Python 인터프리터를 사용합니다.
  + **예시:**
  + {
  + "mcpServers": {
  + "mcp-test": {
  + "command": "python",
  + "args": [
  + "./2\_mcp\_server.py"
  + ]
  + }
  + }
  + }
  + **검증:**
    - 배포 환경의 터미널에서 ls -l ./2\_mcp\_server.py로 파일 존재 여부를 확인합니다.
    - which python으로 기본 Python 인터프리터 위치를 확인할 수 있습니다.

**3. 필수 패키지 설치 및 요구사항 관리**

* **requirements.txt 관리:**
  + **설명:** 배포 환경에서는 requirements.txt 파일에 명시된 패키지들이 자동으로 설치됩니다.
  + **권장 사항:**
    - 위에서 추천한 최소한의 패키지를 포함하는 파일을 업데이트합니다.
    - 예시:
    - streamlit==1.44.1
    - mcp==1.6.0
    - openai==1.70.0
    - openai-agents==0.0.8
    - youtube-transcript-api==1.0.3
    - python-dotenv==1.1.0
    - requests==2.32.3
    - nest-asyncio==1.6.0
  + **확인:**  
    배포 로그를 통해 설치된 패키지 목록을 확인합니다.

**4. 디버깅 및 로그 확인**

* **디버깅 로그 출력:**
  + **설명:** 4\_mcp\_client.py에 추가한 디버깅 로그를 통해 MCP 서버 연결 및 이벤트 스트림 상태를 확인합니다.
  + **예시 로그:**
    - "DEBUG: Attempting to connect to MCP server: ..."
    - "DEBUG: Received event: ..."
  + **확인 방법:**  
    Streamlit Cloud의 로그 페이지에서 해당 로그가 출력되는지 확인합니다.

**5. 네트워크 및 외부 API 호출**

* **API 키 유효성 및 네트워크 연결:**
  + **설명:** Secrets에 설정된 API 키로 외부 API (유튜브, OpenAI 등)가 정상적으로 호출되는지 확인합니다.
  + **검증:**
    - 배포 환경에서 curl이나 다른 HTTP 클라이언트로 API 호출 테스트를 진행해 봅니다.
    - 예를 들어:
    - curl "https://www.googleapis.com/youtube/v3/search?part=snippet&q=test&key=YOUR\_ACTUAL\_API\_KEY"
    - 올바른 JSON 응답이 오면 API 호출은 정상입니다.

**6. 응답 지연 원인 점검**

* **LLM 응답 시간 및 이벤트 스트림:**
  + **설명:** 배포 환경에서는 네트워크 지연, API 호출 타임아웃 등이 있을 수 있으므로, 디버깅 로그를 통해 어느 단계에서 지연이 발생하는지 확인합니다.
  + **추가 점검:**
    - 에이전트 모델("gpt-4o-mini")이 응답을 생성하는 데 걸리는 시간을 측정합니다.
    - 이벤트 스트림에 대한 타임아웃 설정이나 재시도 로직이 필요한지 검토합니다.

**종합 의견**

* **환경에 따른 설정 분리:**
  + 로컬에서는 .env 파일이나 시스템 환경 변수를 활용해 API 키와 경로를 관리하고, Windows 전용 경로를 사용할 수 있습니다.
  + 클라우드 배포 환경에서는 Streamlit Cloud의 Secrets 기능과 Linux/Unix 스타일의 상대 경로를 사용해야 합니다.
* **디버깅 로그 활용:**
  + 각 단계(가상환경, 환경 변수, 파일 경로, MCP 서버 연결, 이벤트 스트림, 외부 API 호출 등)에 디버깅 로그를 추가해 문제의 원인을 단계별로 파악하는 것이 중요합니다.
* **필수 패키지 및 의존성 관리:**
  + 로컬에서 설치된 모든 패키지 중 실제로 앱 실행에 필요한 최소한의 패키지를 requirements.txt에 포함시켜 배포 환경에서의 설치 문제를 해결합니다.

이와 같이 각 항목을 자세하게 점검하고, 환경별로 설정을 분리하여 관리하면 로컬 개발과 클라우드 배포 환경 모두에서 안정적으로 앱을 운영할 수 있습니다.  
문제가 지속될 경우 각 단계별 디버깅 로그와 함께 구체적인 오류 메시지를 다시 검토해보시기 바랍니다.