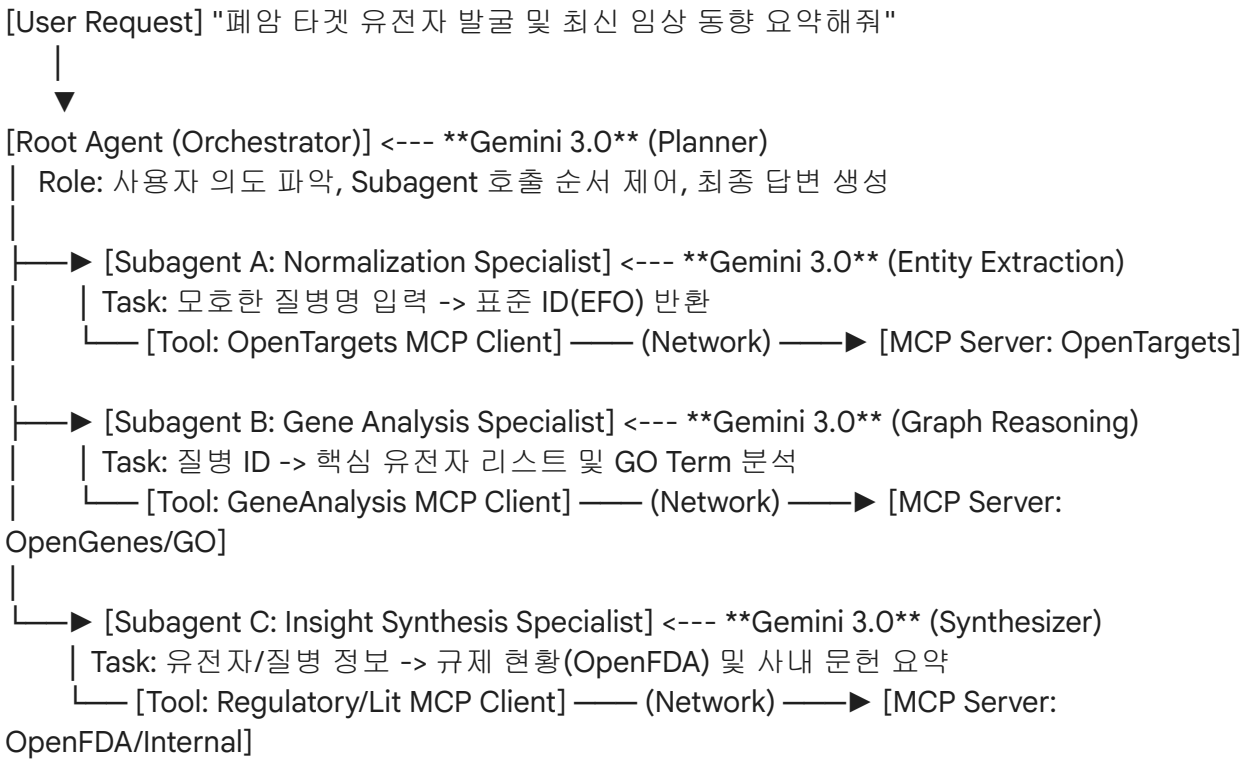
Technical Architecture Definition: 제약 정보 검색 Agent (ADK & MCP 기반)

1. 기술적 목표 (Technical Scope)

본 문서는 제약 R&D의 핵심 워크플로우를 계층형 에이전트**(Layered Agent Architecture)** 구조로 구현하기 위한 기술 아키텍처를 정의한다. **Main Agent**가 전체 흐름을 지휘하고, **Subagents**가 각 단계의 전문 작업을 수행하며, 각 Subagent는 **MCP Client**를 도구(Tool)로 사용하여 외부 시스템과 통신한다.

2. 시스템 아키텍처 (System Architecture)

2.1 High-Level Topology (Hierarchical Agent Structure)



2.2 에이전트 계층 상세 정의 (Agent Stack)

A. Root Agent (The Manager)

- 모델**:** Gemini 3.0
- 역할**:**
  - 사용자와의 대화 인터페이스(Chat).

- 복잡한 작업(Task)을 논리적 단계로 분해.
- 적절한 Subagent에게 작업을 위임(Delegation)하고 결과를 취합.
- 예시: "Subagent A에게 질병명 정규화를 맡기고, 그 결과를 받아 Subagent B에게 전달."

B. Subagents (The Workers)

각 Subagent는 독립적인 Instructions와 Tools를 가진 Vertex AI Agent 인스턴스이다.

1. **Normalization Subagent (Subagent A)**
   - 목표: 정확한 질병/약물 엔티티 추출 및 표준화.
   - 특화 모델: **Gemini 3.0**
   - 보유 도구: OpenTargetsClient (Python Function wrapping MCP Protocol).
   - 동작: 사용자 입력 텍스트 -> TxGemma로 엔티티 추출 -> Tool 호출 -> 표준 ID 반환.
2. **Gene Analysis Subagent (Subagent B)**
   - 목표: 질병-유전자 연관성 분석 및 생물학적 기작(Mechanism) 파악.
   - 모델: Gemini 3.0
   - 보유 도구: OpenGenesClient, GeneOntologyClient.
   - 동작: 표준 ID 수신 -> 연관성 높은 유전자 필터링 -> GO Term 분석 -> 유전자 리스트 반환.
3. **Insight Synthesis Subagent (Subagent C)**
   - 목표: 규제 정보 확인 및 전문 문헌 기반 인사이트 도출.
   - 모델: Gemini 3.0 (요약) + TxGemma (논문 초록 해석 로직 포함 가능).
   - 보유 도구: OpenFDAClient, InternalVectorSearchClient(별도로 구현함).
   - 동작: 유전자/질병 정보 수신 -> 승인 약물 조회 + 사내 논문 RAG -> 종합 리포트 생성.

3. 타겟 시나리오 상세 설계: 질병-유전자-문헌 분석 파이프라인

3.1 단계별 데이터 소스 및 MCP 매핑

Subagent	담당 역할	활용 데이터 **(Source)**	연결 도구 **(MCP Client)**
Normalization Agent	**Step 1:** 용어 정규화	**Open Targets**	**tool_normalize_term** (Calls OpenTargets MCP Server)
Gene Analysis Agent	**Step 2:** 타겟 발굴	**OpenGenes** **Gene Ontology**	**tool_get_associations** (Calls

			OpenGenes/GO MCP Server)
Insight Agent	**Step 3:** 정보 종합	**OpenFDA** (Public) **Internal DB** (Private, 추후 구현 범위)	**tool_get_regulatory_info** **tool_search_internal_docs** (Calls Literature MCP Server)

3.2 Agent Orchestration Logic (Delegation Flow)

Root Agent가 Subagent들을 제어하는 논리적 흐름은 다음과 같다.

1. **Start:** 사용자 질문 수신 ("폐암 관련 유전자와 약물 알려줘").
2. **Delegation to Subagent A (Normalization):**
   - Root Agent: "입력 문장에서 질병명을 찾아 정규화해줘."
   - **Subagent A:** (TxGemma) tool_normalize_term("폐암") 호출 -> {"id": "EFO_0000384", "name": "Non-small cell lung carcinoma"} 반환.
3. **Context Passing & Delegation to Subagent B (Gene):**
   - Root Agent: "EFO_0000384 질병의 주요 타겟 유전자를 찾아줘."
   - **Subagent B:** tool_get_associations("EFO_0000384") 호출 -> ["EGFR", "KRAS", "ALK"] 반환.
4. **Delegation to Subagent C (Insight):**
   - Root Agent: "NSCLC 질병과 EGFR, KRAS 유전자에 대한 FDA 승인 정보와 우리 사내 문헌을 조사해줘."
   - **Subagent C:**
     - tool_get_regulatory_info("NSCLC", "EGFR") 호출 (OpenFDA).
     - tool_search_internal_docs("EGFR mutation resistance") 호출 (Internal).
     - 결과를 종합하여 텍스트로 요약 후 반환.
5. **Final Response:**
   - Root Agent: Subagent C의 리포트를 사용자 친화적인 말투로 다듬어 최종 답변 출력.

4. 인프라 및 보안 (Infra & Security)

4.1 MCP Client-Server 구조

기존에는 Agent가 직접 HTTP 요청을 보냈다면, 이제는 **Subagent 내부에 등록된 Function(Client)**이 실제 통신을 담당한다.

- **MCP Client (in Agent):** Python 코드로 작성된 Vertex AI Function Tool. MCP 프로토콜(JSON-RPC)에 맞춰 Request Body를 생성하고 Cloud Run Endpoint로 전송.

- **MCP Server (in Cloud Run):** 실제 DB 접속 및 쿼리 수행.

4.2 보안 격리

- **Subagent 권한 분리:**
  - Insight Agent만 사내 문서(Internal DB) 접근 권한을 가진 Service Account를 사용.
  - Normalization Agent는 외부망(OpenTargets) 접근만 허용.
- **TxGemma 배포:**
  - Normalization Subagent가 사용하는 TxGemma는 Vertex AI Endpoint에 Private하게 배포되어 데이터 유출 방지.

5. 개발 및 배포 환경 (DevOps Specs)

5.1 Repository Structure (Modularized)

```
/
├── agent-root/            # Main Agent (Orchestrator) 정의
├── subagents/
│   ├── normalization/     # [Subagent A]
│   │   ├── instructions.md  # TxGemma 프롬프트
│   │   └── tools/         # MCP Client Code (OpenTargets)
│   ├── gene-analysis/     # [Subagent B]
│   │   ├── instructions.md
│   │   └── tools/         # MCP Client Code (OpenGenes)
│   └── insight-synthesis/   # [Subagent C]
│       ├── instructions.md
│       └── tools/         # MCP Client Code (OpenFDA/Internal)
├── mcp-servers/           # Backend Servers (Dockerized)
│   ├── opentargets-mcp/
│   ├── opengenes-mcp/
│   ├── OpenFDA-MCP-Server/
│   └── GeneOntology-MCP-Server/
└── infra/                 # Terraform (Vertex AI Agents, Cloud Run)
```

6. 참고 자료 (Reference Repositories)

각 MCP Server 구현 시 참조할 수 있는 오픈소스 리포지토리 목록입니다.

구분	단계 **(Step)**	**GitHub URL**	용도
OpenTargets	Step 1 (Normalization)	https://github.com/nickzren/opentargets-mcp	질병명 정규화 및 EFO ID 검색 (GraphQL 연동

			예제)
OpenGenes	Step 2 (Gene Analysis)	https://github.com/longevity-genie/opengenes-mcp	질병-유전자 연관성 및 수명 관련 유전자 데이터 조회
Gene Ontology	Step 2 (Gene Analysis)	https://github.com/Augmented-Nature/GeneOntology-MCP-Server	유전자의 생물학적 기능(GO Terms) 분석
OpenFDA	Step 3 (Insight Synthesis)	https://github.com/Augmented-Nature/OpenFDA-MCP-Server	FDA 승인 약물 정보, 라벨 및 부작용 데이터 검색