AI Agent

Google ADK 시작하기

1. Google ADK 시작하기

LangGraph에서 ADK로의 전환

실제 서비스(Production) 환경에서는 더 많은 기능이 필요

- Agent가 24/7 중단 없이 안정적으로 실행될 수 있는가?
- 수천 명의 사용자가 동시에 요청해도 처리할 수 있는가?
- Agent 활동과 비용을 실시간으로 알 수 있는가?
- Agent가 예상과 다르게 동작했을 때, 그 원인을 어떻게 추적할 것인가?
- 회사의 중요한 데이터에 접근하는 Agent의 권한을 어떻게 제어할 것인가?

LangGraph에서 ADK로의 전환

LangGraph

- Graph-first philosophy
- Agent의 흐름을 저수준에서 제어
- Langchain의 구조를 다각화하기 위함
- State를 기반으로 복잡한 워크플로우를 유연하게 설계 가능

LangGraph에서 ADK로의 전환

Google ADK (Agent Development Kit)

- Model-first philosophy
- 실제 서비스 운영 Production에 필요한 기능들을 통합 제공
- LangGraph에 비해 고수준 추상화 기능들이 제공
- Google Cloud의 공식 개발 및 운영 프레임워크
- Google Cloud(Vertex AI)의 인프라 위에서 안정적으로 개발, 배포, 운영

ADK Agent

핵심 구성 요소

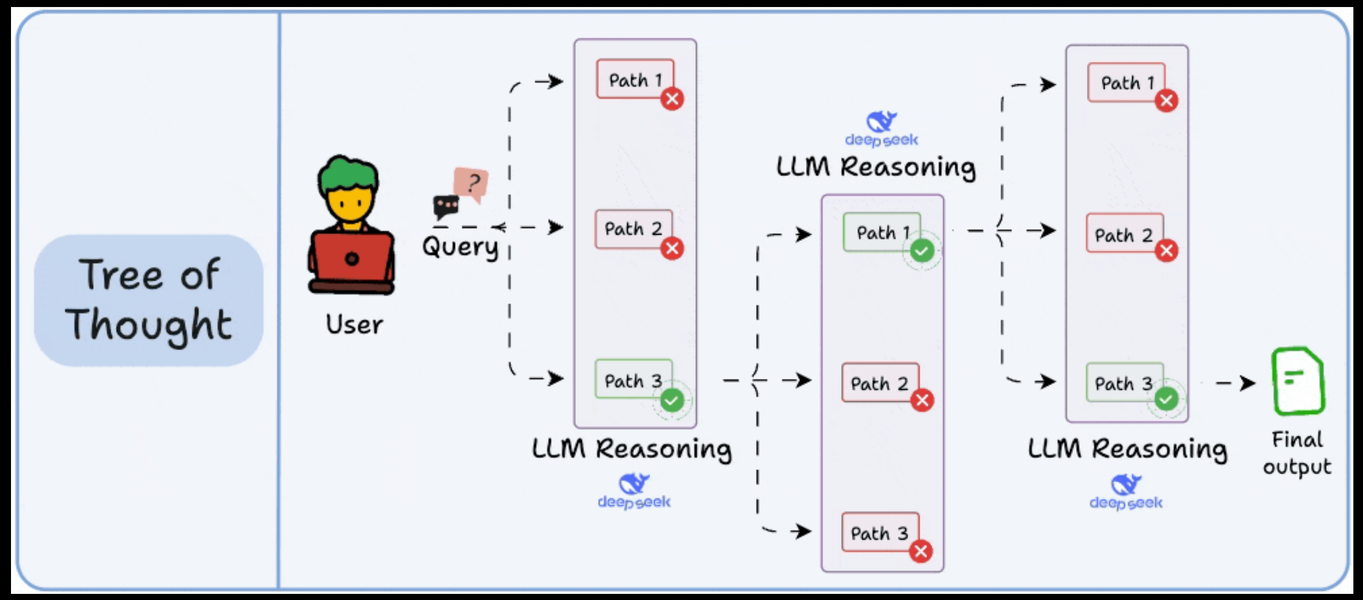
- Agent
  - Model
  - Description
  - Instruction
  - Tools
- Session
- Callbacks

```python
weather_agent = Agent(
    name="weather_agent_v1",
    model=AGENT_MODEL, # Gemini 모델 또는 LiteLLM
    description="특정 도시에 대한 날씨 정보를 알려줍니다.",
    instruction="You are a helpful weather assistant. "
                "When the user asks for the weather in a specific city, "
                "use the 'get_weather' tool to find the information. "
                "If the tool returns an error, inform the user politely. "
                "If the tool is successful, present the weather report clearly.",
    tools=[get_weather], # 함수를 직접 입력
)
```

Observability

관측 가능성 (Observability)

- LLM 기반 Agent는 비결정적(Non-deterministic) 특성으로 디버깅이 어려움
- 동일한 입력에도 다른 결과나 추론 과정이 나올 수 있음
- Agent 생각의 흐름 전체를 기록한 장부가 필요
- ADK는 Trajectory라는 구조화된 로그로 자동 기록하여 이 문제를 해결



https://blog.dailydoseofds.com/p/3-prompting-techniques-for-reasoning

Observability

Trajectory

- Agent 세션 동안 발생한 모든 이벤트(Event)의 순차적인 기록
- 각 단계의 목적과 데이터가 명확히 구분된 구조화된 데이터
- ADK의 SessionService는 이 모든 이벤트를 메모리나 데이터베이스에 기록
- 개발자는 세션 ID만 알면 언제든지 이 전체 Trajectory를 조회 가능

Observability

주요 이벤트 유형

- User Input: 사용자의 최초 요청
- LLM Request: Agent가 LLM에게 보낸 프롬프트
- LLM Response (Tool Call): LLM이 Tool 사용을 제안한 내용
- Tool Request: ADK가 실제 Tool 함수를 호출한 기록
- Tool Response: Tool 함수가 반환한 결과
- LLM Response (Final Answer): 최종 사용자 답변
- State Delta: state가 어떻게 변경되었는지에 대한 기록

Observability

Trajectory 기반 Action

- 디버깅 (Debugging)
  - Agent가 예상과 다른 Tool을 호출했거나, 잘못된 답변을 생성했을 때
  - Trajectory를 단계별로 따라가며 어디서 잘못된 추론이 시작되었는지 탐색 가능
- 성능 분석
  - 각 LLM 호출과 Tool 실행에 걸린 시간을 분석
  - 워크플로우의 병목 지점을 찾아내고 최적화

Observability

Trajectory 기반 Action

- 비용 추적
  - LLM 호출 횟수와 사용된 토큰 수를 기록
  - 특정 작업에 드는 API 비용을 정확하게 계산하고 관리
- 감사 및 규정 준수 (Auditing & Compliance)
  - 민감한 데이터를 다루는 Agent의 경우, 모든 결정 과정을 기록
  - 나중에 감사에 대응하거나 규정을 준수했음을 증명

GCP Integration

GCP Integration

- ADK는 GCP의 다양한 서비스들과 통합되도록 설계
- 수백만 명이 사용하는 서비스 수준으로 확장할 수 있는 기반
- 실제 서비스 하려면 데이터 저장, 모델 배포, 보안 관리 등 수많은 인프라 작업이 필요
- ADK는 GCP의 솔루션들을 활용하여 이러한 복잡성을 해결

GCP Integration

Vertex AI:

- Google Cloud의 통합 MLOps(Machine Learning Operations) 플랫폼
- 데이터 관리, 모델 훈련, 배포, 모니터링 등 머신러닝 모델의 전체 수명 주기를 관리

- How ADK integrates:
  - ADK로 개발된 Agent는 Vertex AI Endpoints 형태로 안정적 배포 가능
  - 자동 확장(Auto-scaling) 기능을 통해 트래픽 변화에 유연하게 대응
  - Vertex AI Experiments와 같은 도구를 사용
  - 여러 버전의 Agent 프롬프트나 로직을 A/B 테스트하고 성능을 추적, 관리

GCP Integration

GCS & Cloud Logging:

- Agent의 외부 기억장치와 일지
- Google Cloud Storage (GCS):
  - 대용량 파일을 저장하는 확장성이 뛰어난 객체 스토리지 서비스
- Cloud Logging:
  - 대규모 로그 데이터를 수집, 검색, 분석, 모니터링

- How ADK integrates:
  - InMemorySessionService 대신 GCS나 Firestore 기반의 SessionService를 사용
  - Agent의 State를 GCS에 안전하게 영속적 저장
  - 서버가 재시작되어도 대화 내용이 보존
  - Agent의 모든 Trajectory 이벤트를 Cloud Logging으로 자동으로 스트리밍
  - 개발자는 강력한 쿼리 언어를 사용하여 특정 사용자의 대화 기록을 검색
  - 특정 종류의 오류가 얼마나 자주 발생하는지 분석하고 대시보드 작성 가능