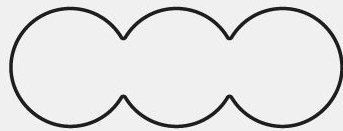
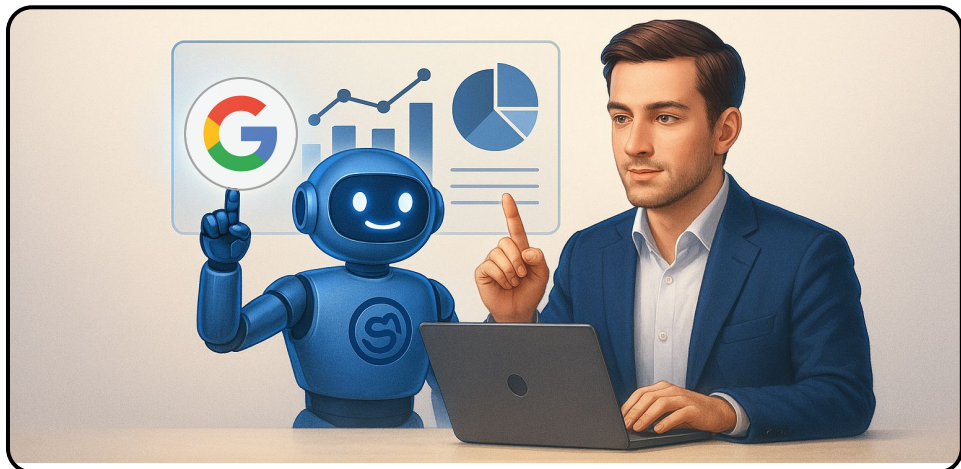





Google Developer Group  
Incheon

# Gemini API를 활용한 데이터 분석 에이전트 만들기



**{ Build  with AI }**

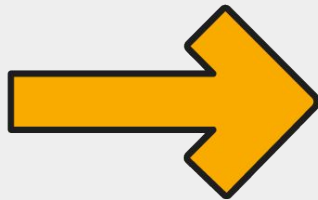
# 목차

- 1 Agent란?
- 2 우리 주변의 Agent
- 3 Agent를 위한 기술들
- 4 Agent 실습
- 5 아이디어 공유 및 회고



Chapter One

# Agent란?





Google Developer Group  
Incheon

## 방승욱

### 주과목

Machine  
Learning

Deep  
Learning

RAG

Data  
Analytics

Data  
Visualization

LLM

### 약력

모두의연구소 교육퍼실리테이터팀 재직

모두의연구소 쏘카 과정 퍼실리테이터

모두의연구소 데이터사이언티스트 과정 퍼실리테이터

모두의연구소 프로덕트 데이터 분석가 과정 퍼실리테이터

고등학교 진로특강 및 AI/Data 세미나 진행 경험 다수



{ Build  with AI }

**Agent란?**



# Agent란?

Agent란 스스로 판단하고 행동하는 인공지능 시스템

사용자의 요청에 대해 주변 환경을 인식하고, 그에 맞는 행동을 선택해 문제를 해결하거나 목표를 달성하기 위한 적절한 행동을 수행합니다.

예를 들어, 문서를 요약하거나 정리된 내용을 PDF로 저장하는 등의 작업을 스스로 수행할 수 있어 실생활에서도 다양하게 활용되고 있습니다.

# Agent란?

- **Single-step Agent**

- 한 번의 명령 또는 입력에 대해 즉시 결과를 반환
- 별도의 계획 없이, 단순하거나 반복적인 작업에 적합
- 예: "이 문장을 영어로 번역해줘" → 번역 결과 출력

- **Multi-step Agent**

- 복잡한 목표를 달성하기 위해 여러 단계를 계획하고 실행
- 예: "이 주식 데이터를 분석해보고 리포트로 만들어줘"
- → 데이터 다운로드 → 분석 코드 실행 → 시각화 → 리포트 작성 → PDF 저장
- LLM 기반 AI 에이전트들이 주로 사용하는 방식

# Agent란?

- **Single-hop 추론**

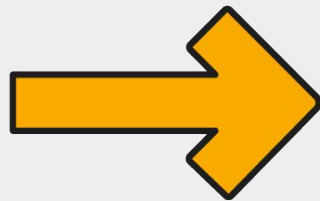
- 하나의 정보(문서, 문장)만으로 질문에 답할 수 있음
- 질문 → 바로 답
- 질문: "구글의 **CEO**는 누구야?"
- 답변: "순다 피차이" (단일 문장에서 바로 추론 가능)

- **Multi-hop 추론**

- 여러 개의 정보 조각을 연결해서 답을 유도해야 함
- 질문 → 정보 **A** → 정보 **B** → 최종 답
- 질문: "순다 피차이가 **CEO**인 회사가 만든 모바일 운영체제는?"
- 추론 과정: [**A**] 순다 피차이 → 구글 **CEO** [**B**] 구글 → 안드로이드 개발
  - 정답: 안드로이드



# 우리 주변의 Agent



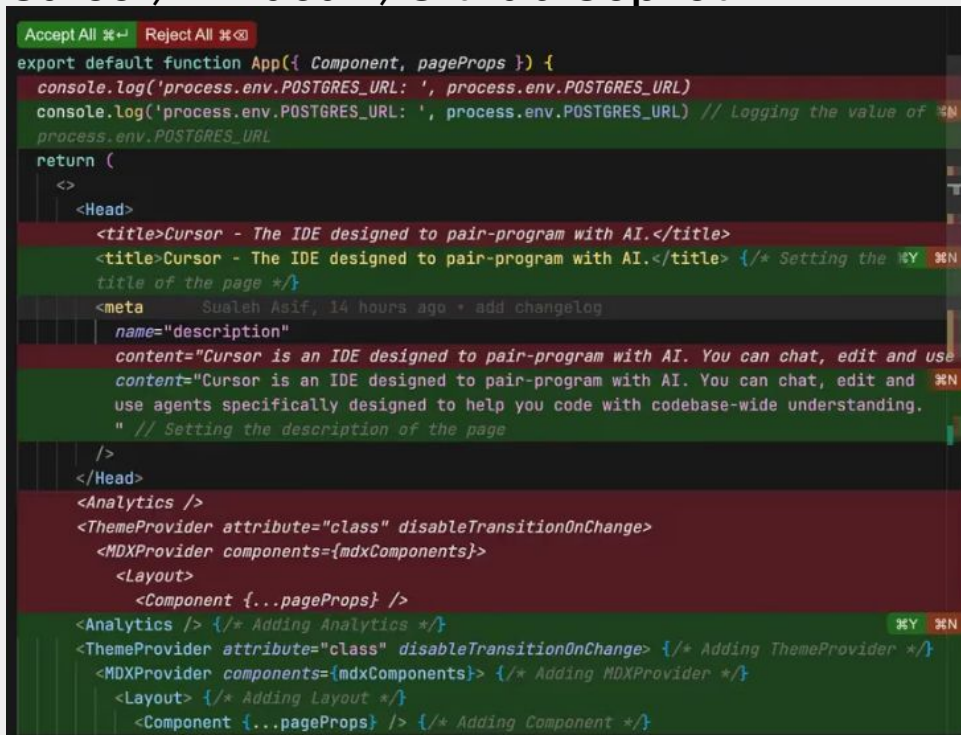
# 우리 주변의 Agent

- 개발 보조 도구
  - Cursor, WindSurf, Github Copilot

```
vscode > src > vs > workbench > contrib > aiCpp > electron-sandbox > TS cppActions.ts > AcceptCppSuggestionAction > run
216 export class AcceptCppSuggestionAction extends Action2 {
220   constructor() {
228     menu: {
229       id: CppDiffPeekView.TitleMenu,
230       order: 1,
231     },
232   });
233 }
234 // KL to chat, MK to generate
235 }
236
237 registerAction2(AcceptCppSuggestionAction);
238
239 function getGhostTextControllerFromAccessor(accessor: ServicesAccessor): GhostTextController | undefined {
240   const codeEditorService = accessor.get(ICodeEditorService);
241   const currentEditor = codeEditorService.getActiveCodeEditor() || codeEditorService.getFocusedCodeEditor();
242   if (!currentEditor) {
243     return;
244   }
245   const controller = GhostTextController.get(currentEditor);
246   return controller || undefined;
247 }
```

# 우리 주변의 Agent

- 개발 보조 도구
  - Cursor, WindSurf, Github Copilot



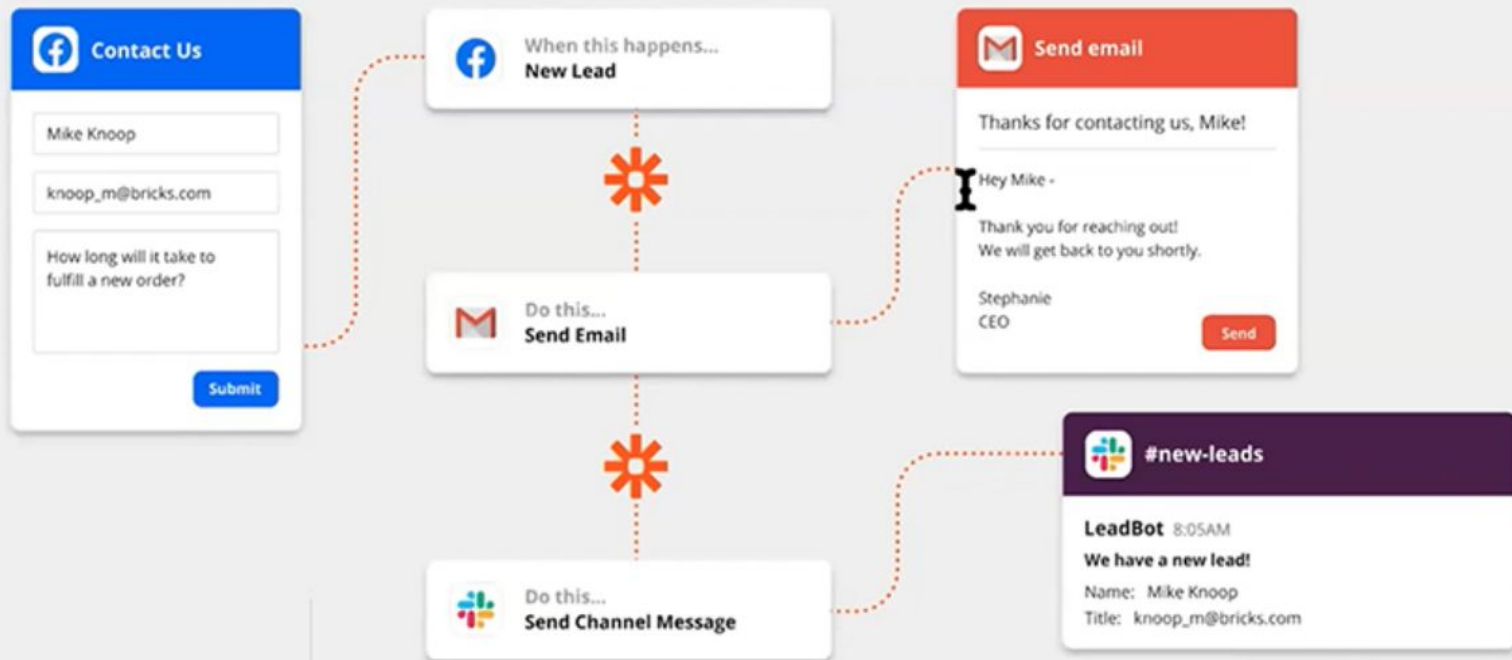
```
Accept All ✖ Reject All ✖
export default function App({ Component, pageProps }) {
  console.log('process.env.POSTGRES_URL: ', process.env.POSTGRES_URL)
  console.log('process.env.POSTGRES_URL: ', process.env.POSTGRES_URL) // Logging the value of %M
  process.env.POSTGRES_URL
  return (
    <>
    <Head>
      <title>Cursor - The IDE designed to pair-program with AI.</title>
      <title>Cursor - The IDE designed to pair-program with AI.</title> {/* Setting the %Y %N
        title of the page */}
      <meta      Sualah Asif, 14 hours ago • add changelog
        name="description"
        content="Cursor is an IDE designed to pair-program with AI. You can chat, edit and use
        content="Cursor is an IDE designed to pair-program with AI. You can chat, edit and %N
        use agents specifically designed to help you code with codebase-wide understanding.
        " // Setting the description of the page
      />
    </Head>
    <Analytics />
    <ThemeProvider attribute="class" disableTransitionOnChange>
      <MDXProvider components={mdxComponents}>
        <Layout>
          <Component {...pageProps} />
        </Layout>
      </MDXProvider>
    </ThemeProvider>
  )
}
</>
```

# 우리 주변의 **Agent**

- 업무 자동화 도구
  - Zapier
    - 수천 개의 앱과 서비스를 연결해, 특정 조건이 발생하면 자동으로 작업을 수행하도록 설정할 수 있는 노코드 자동화 플랫폼

# 우리 주변의 Agent

- 업무 자동화 도구
  - Zapier

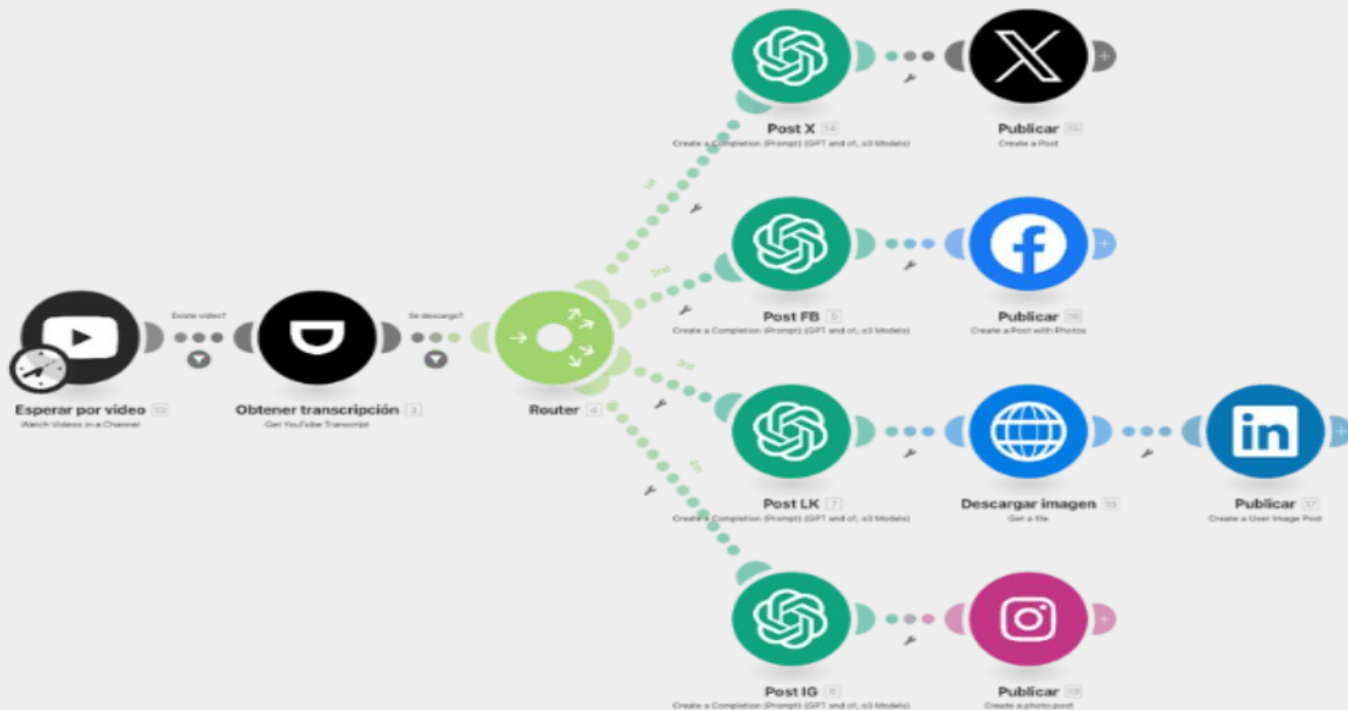


# 우리 주변의 **Agent**

- 업무 자동화 도구
  - Make
    - 앱과 서비스를 연결해 복잡한 다단계 작업을 시각적으로 설계할 수 있는 노코드 워크플로우 플랫폼

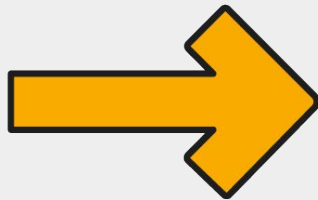
# 우리 주변의 Agent

- 업무 자동화 도구
  - Make



## Chapter Three

# Agent를 위한 기술들





# Agent를 위한 기술들

- LangChain

- LangGraph

# Agent를 위한 기술들

- **LangChain**

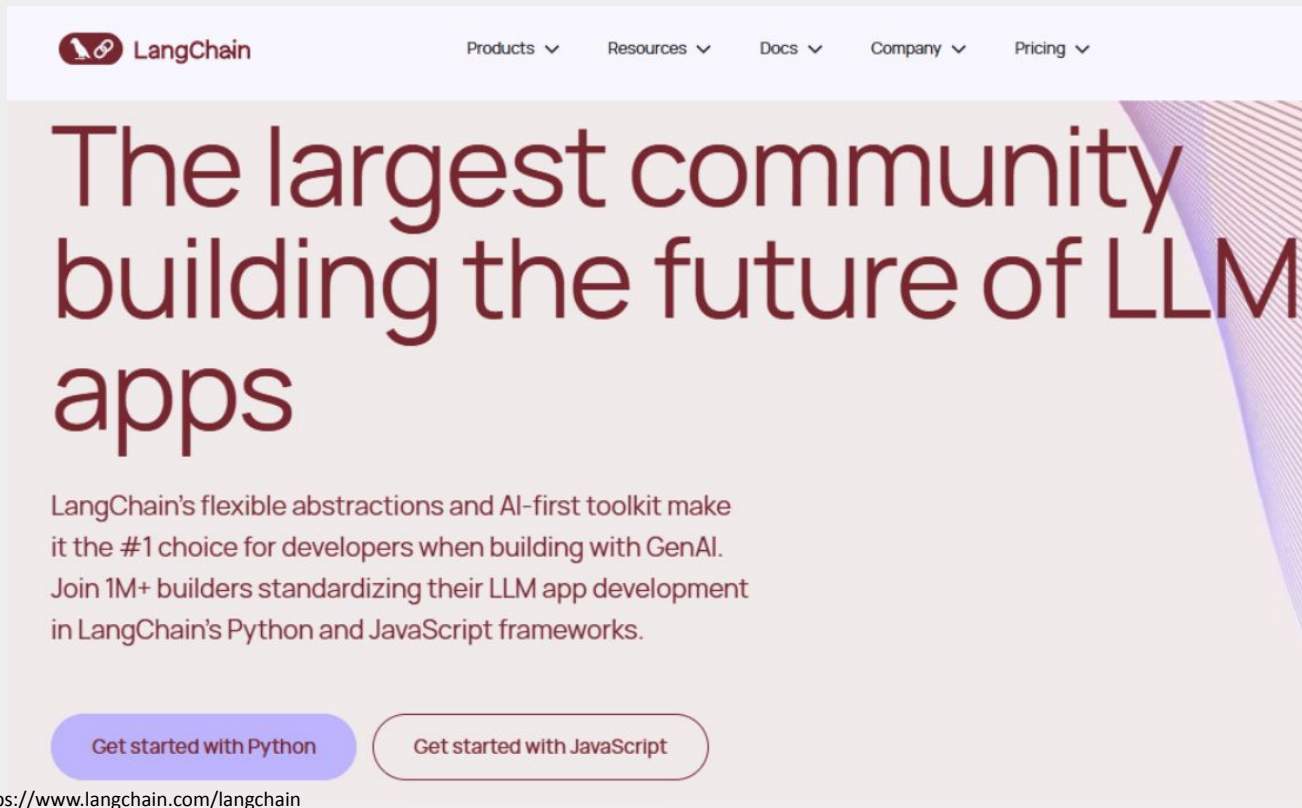
**LangChain**은 **LLM**을 활용한 에이전트를 쉽게 만들 수 있게 해주는 대표적인 오케스트레이션 도구입니다.

**AI** 개발자들이 가장 먼저 선택할 수 있는 대표적인 툴킷 중 하나로, 검색, 데이터베이스 조회, 코드 실행, 웹 브라우징 등 다양한 기능을 ‘체인(**chain**)’ 구조로 연결하여 **Agentic**한 동작을 구현하기에 최적화되어 있습니다.

**Python**과 **JavaScript** 기반으로 작동하며, 다양한 **LLM**과 외부 도구를 유연하게 지원합니다.

# Agent를 위한 기술들

- LangChain



이미지 출처 : <https://www.langchain.com/langchain>

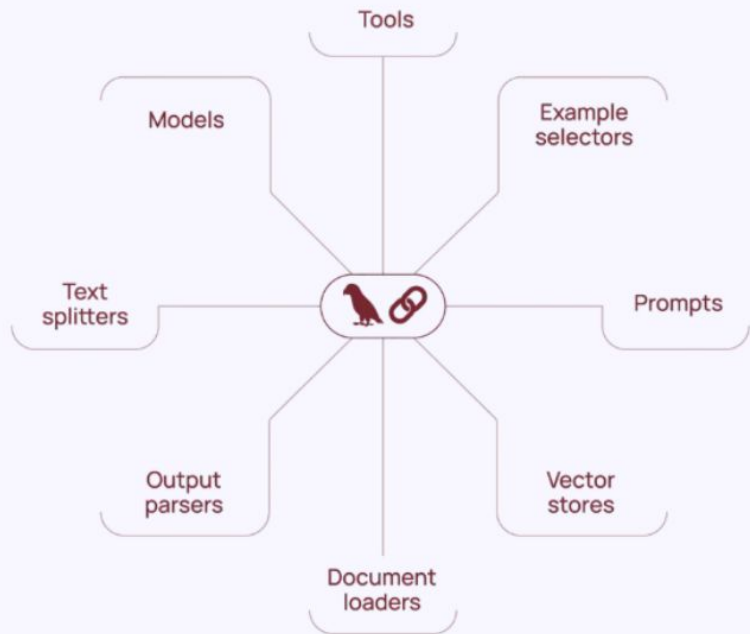
# Agent를 위한 기술들

- LangChain

## A complete set of interoperable building blocks

Build end-to-end applications with an extensive library of components. Want to change your model? Future-proof your application by incorporating vendor optionality into your LLM infrastructure design.

[Read the docs ↗](#)



# Agent를 위한 기술들

- **LangChain**

**LCEL**이라는 자체 문법을 통해 구성 요소를 손쉽게 조합할 수 있습니다.

이를 이용해 유연하고 커스터마이징이 간단한 모델을 손쉽게 개발할 수 있습니다.

# Agent를 위한 기술들

- LangChain

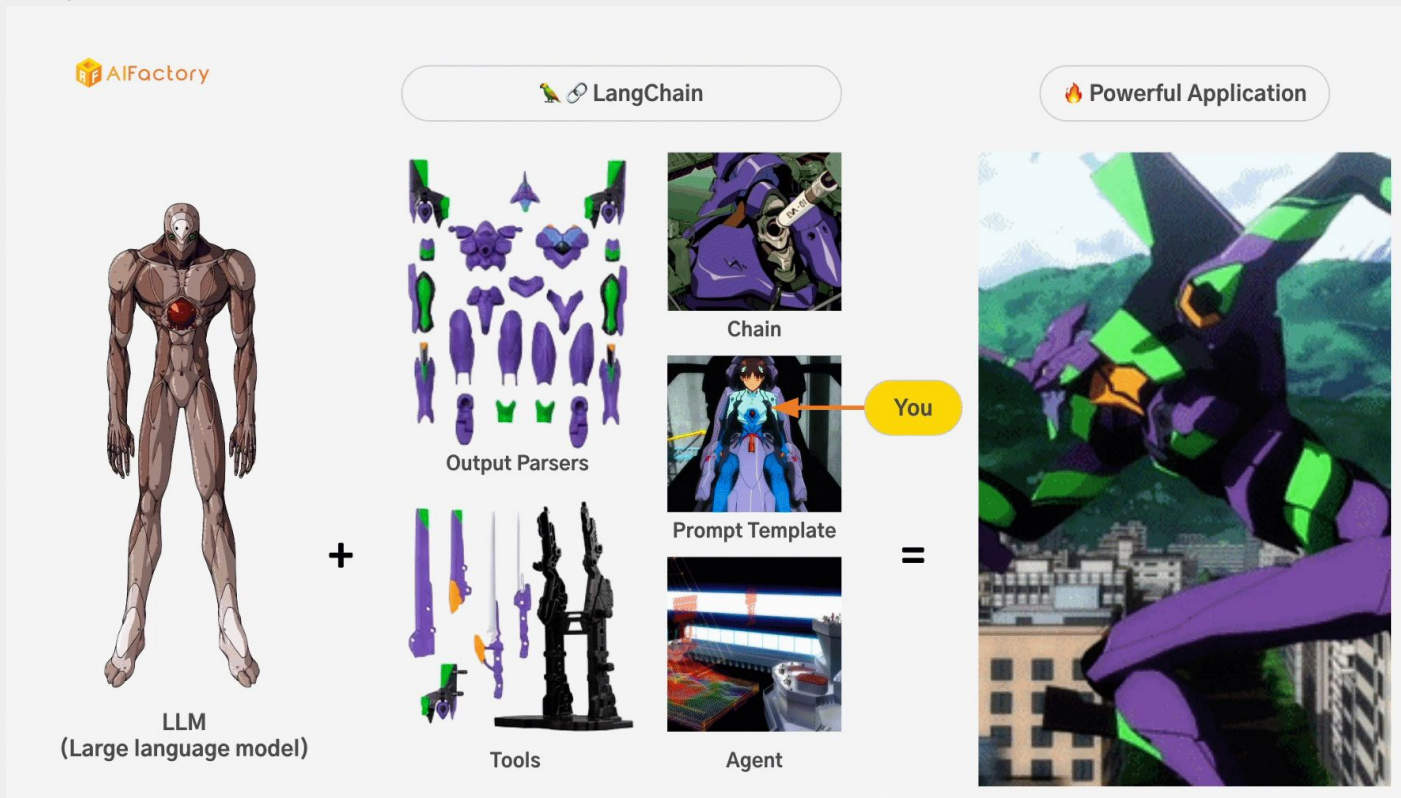
Python

```
>>> from langchain_community.vectorstores import (
    Chroma
)
>>> from langchain_core.output_parsers import StrOutputParser
>>> from langchain_core.prompts import ChatPromptTemplate
>>> from langchain_core.runnables import RunnablePassthrough
>>> from langchain_openai import ChatOpenAI

chain = (
    RunnablePassthrough.assign(
        context=Chroma(...).as_retriever()
    ) | ChatPromptTemplate(...)
    | ChatOpenAI(...)
    | StrOutputParser()
)
```

# Agent를 위한 기술들

- LangChain



# Agent를 위한 기술들

- **LangGraph**

복잡한 **AI Agent** 흐름을 제어하는 상태 기반 프레임워크

**LangGraph**는 **LangChain** 생태계에서 복잡한 멀티스텝 **AI Agent**의 흐름을 안정적으로 제어하기 위한 상태 기반 워크플로우 오케스트레이션 도구입니다.

각 상태(**state**)를 노드로 정의하고, 조건(**condition**)에 따라 다음 상태로 분기, 반복, 종료할 수 있음

**LangChain**과 통합되어, 체인/**Agent**를 조립해 복잡한 시나리오 구현 가능



# Agent를 위한 기술들

- LangGraph의 핵심 3요소
  - 상태 (State)
  - 노드 (Node)
  - 엣지 (Edge)

# Agent를 위한 기술들

- **상태 (State)**

- 에이전트가 현재 어떤 단계에 있는지를 나타내는 정보입니다.
- 예: 사용자의 입력, 지금까지의 도구 실행 결과, 다음 작업 지시 등
- 상태는 계속 업데이트되며, 전체 흐름의 컨텍스트 역할을 함

```
class State(TypedDict):  
    messages : Annotated[list, add_messages]  
    document : Annotated[Document, "Retrieve Response"]
```

# Agent를 위한 기술들

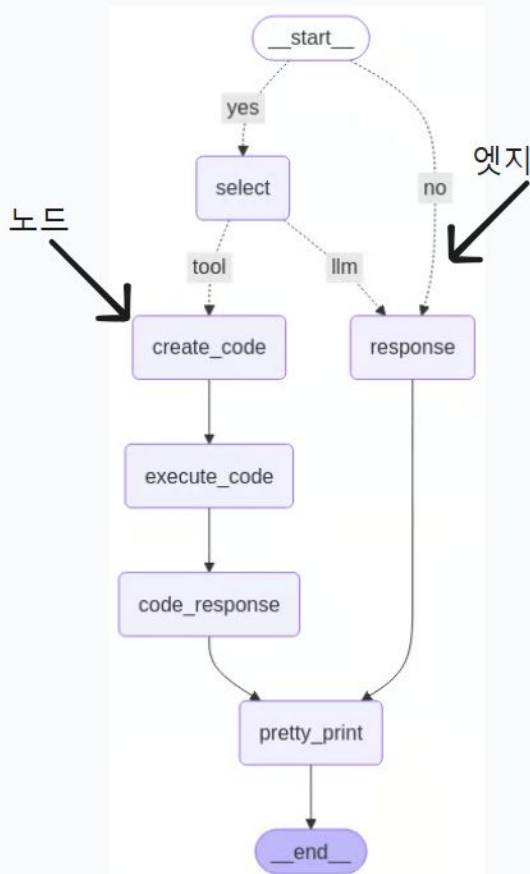
- **노드 (Node)**

- 하나의 독립적인 작업 단위입니다.
- 상태를 입력받아, 작업을 수행한 후 변경된 상태를 반환

- **엣지 (Edge)**

- 노드 간의 흐름을 정의하는 연결선
- 작업 결과나 상태 조건에 따라 다음 노드를 선택
- 분기, 반복, 종료 같은 흐름 제어가 가능

# Agent를 위한 기술들



# Agent를 위한 기술들

- 상태 (State)의 변화 예시

# Agent를 위한 기술들

- 상태 (State)의 변화 예시

입력

```
{'query': '삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.',  
  "answer", "",  
  'messages': [HumanMessage(content='삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.')]}
```

# Agent를 위한 기술들

- 상태 (State)의 변화 예시

입력

```
{'query': '삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.',  
  'answer': '',  
  'messages': [HumanMessage(content='삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.')]}
```



도구 호출

```
{'query': '삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.',  
  'answer': '',  
  'messages': [HumanMessage(content='삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.')] ,  
  'tool_call': 'Tavily_Search'}
```

# Agent를 위한 기술들

- 상태 (State)의 변화 예시

입력

```
{'query': '삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.!',  
  'answer': '',  
  'messages': [HumanMessage(content='삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.')]}
```



도구 호출

```
{'query': '삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.!',  
  'answer': '',  
  'messages': [HumanMessage(content='삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.'],  
  'tool_call': 'Tavily_Search'}
```



도구 사용 결과

```
{'query': '삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.!',  
  'answer': '['삼성전자는 ...', '삼성전자의 현재 주가는...', '삼성전자 회장인 이재용은...']',  
  'messages': [HumanMessage(content='삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.',  
    ToolMessage('삼성전자는 ...', '삼성전자의 현재 주가는...', '삼성전자 회장인 이재용은...'))],  
  'tool_call': 'Tavily_Search'}
```



# Agent를 위한 기술들

- 상태 (State)의 변화 예시

## 도구 사용 결과

```
{'query': '삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.!',  
'answer': ['삼성전자는 ...', '삼성전자의 현재 주가는...!', '삼성전자 회장인 이재용은...!'],  
'messages': [HumanMessage(content='삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.!',  
                           ToolMessage('삼성전자는 ...', '삼성전자의 현재 주가는...!', '삼성전자 회장인 이재용은...!)),  
'tool_call': 'Tavily_Search']
```

# Agent를 위한 기술들

- 상태 (State)의 변화 예시

## 도구 사용 결과

```
{'query': '삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.',  
'answer': ['삼성전자는 ...', '삼성전자의 현재 주가는...', '삼성전자 회장인 이재용은...'],  
'messages': [HumanMessage(content='삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.',  
                        ToolMessage('삼성전자는 ...', '삼성전자의 현재 주가는...', '삼성전자 회장인 이재용은...'))],  
'tool_call': 'Tavily_Search'}
```



## 응답 확인 후 도구 재호출

```
{'query': '삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.',  
'answer': ['삼성전자는 ...', '삼성전자의 현재 주가는...', '삼성전자 회장인 이재용은...'],  
'messages': [HumanMessage(content='삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.',  
                        ToolMessage('삼성전자는 ...', '삼성전자의 현재 주가는...', '삼성전자 회장인 이재용은...'))],  
'tool_call': 'write pdf'}
```

# Agent를 위한 기술들

- 상태 (State)의 변화 예시

응답 확인 후 도구 재호출

```
{'query': '삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.',  
'answer': ['삼성전자는 ...', '삼성전자의 현재 주가는...!', '삼성전자 회장인 이재용은...!'],  
'messages': [HumanMessage(content='삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.',  
                        ToolMessage('삼성전자는 ...', '삼성전자의 현재 주가는...!', '삼성전자 회장인 이재용은...!)),  
'tool_call': 'write pdf'
```

# Agent를 위한 기술들

- 상태 (State)의 변화 예시

응답 확인 후 도구 재호출

```
{'query': '삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.!',  
'answer': ['삼성전자는 ...', '삼성전자의 현재 주가는...!', '삼성전자 회장인 이재용은...!'],  
'messages': [HumanMessage(content='삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.!',  
                        ToolMessage('삼성전자는 ...', '삼성전자의 현재 주가는...!', '삼성전자 회장인 이재용은...!)),  
'tool_call': 'write pdf'
```

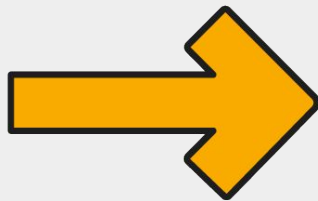


결과

```
{'query': '삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.!',  
'answer': '삼성전자 요약 리포트.pdf파일 생성을 완료하였습니다.!',  
'messages': [HumanMessage(content='삼성전자에 대해 조사해서 pdf 파일로 만들어주세요.!',  
                        ToolMessage('삼성전자는 ...', '삼성전자의 현재 주가는...!', '삼성전자 회장인 이재용은...!'),  
                        ToolMessage('삼성전자 요약 리포트.pdf파일 생성을 완료하였습니다.!)),  
'tool_call': 'write pdf'
```

## Chapter Four

# Agent 실습

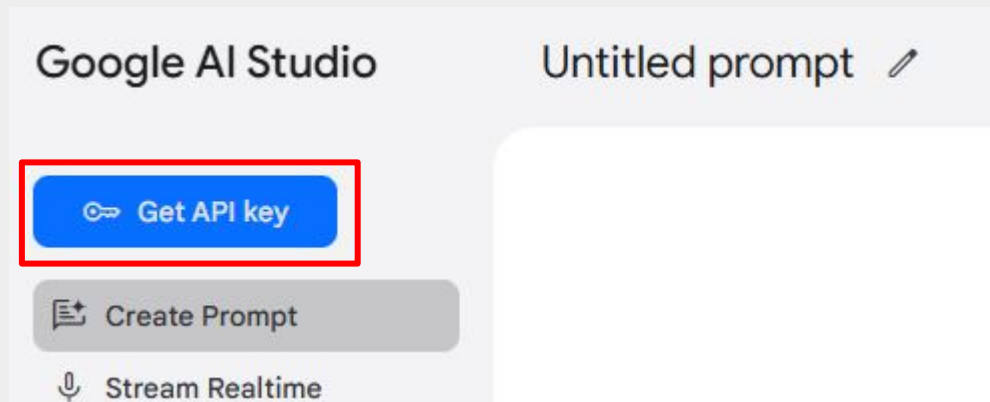
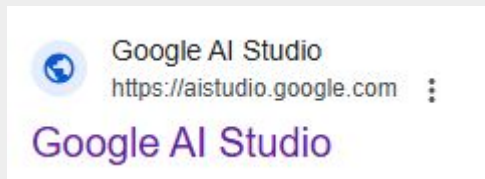


# Agent 실습

- 사용할 API Key
  - Google API Key
    - Gemini 이용을 위한 API Key
  - Tavily API Key
    - 웹 검색을 위한 API Key

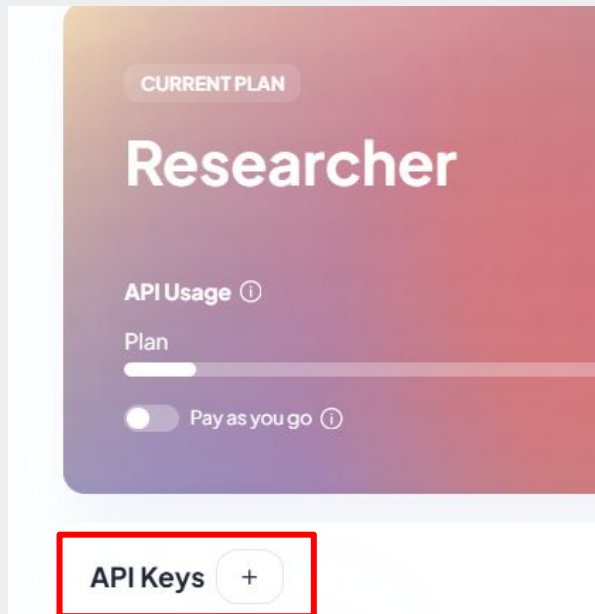
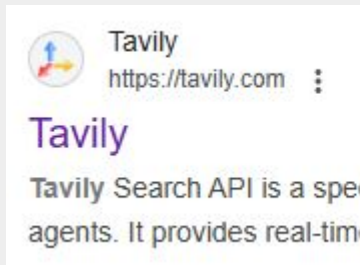
# Agent 실습

- 사용할 API Key
  - Google API Key
    - Gemini 이용을 위한 API Key



# Agent 실습

- 사용할 API Key
  - Tavily API Key
    - 웹 검색을 위한 API Key





## Agent 실습

- 데이터 분석 에이전트
- 레포트 에이전트

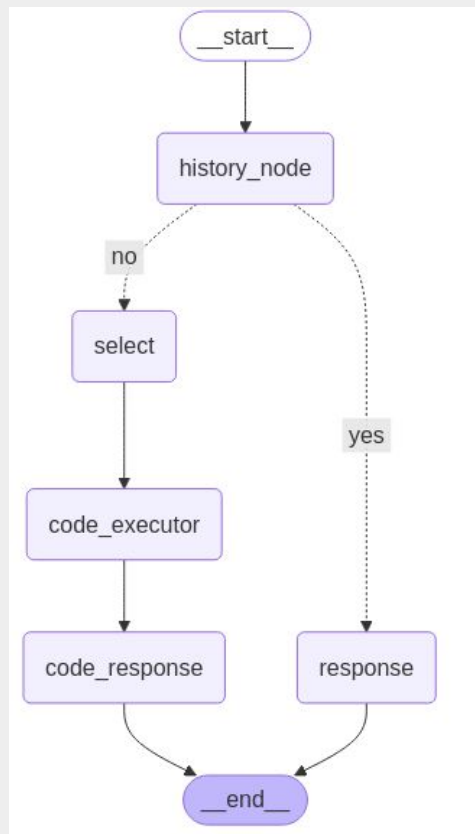
## Agent 실습

- 데이터 분석 에이전트 만들기

# Agent 실습

- 데이터 분석 에이전트 만들기

```
class State(TypedDict):  
    messages : Annotated[list, add_messages] # History  
    code : Annotated[str, "Python code"]  
    code_result : Annotated[str, "Python code Result"]
```



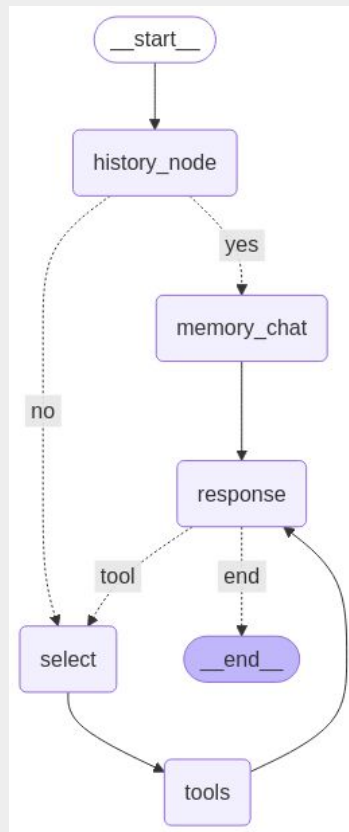
## Agent 실습

- 레포트 에이전트 만들기

# Agent 실습

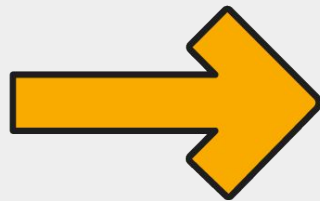
- 레포트 에이전트 만들기

```
class State(TypedDict):  
    query : Annotated[str, "User Question"]  
    answer : Annotated[str, "LLM response"]  
    messages : Annotated[list, add_messages]  
    tool_call : Annotated[dict, "Tool Call Result"]
```



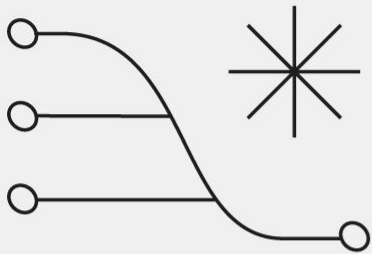
## Chapter Five

# 아이디어 공유 및 회고

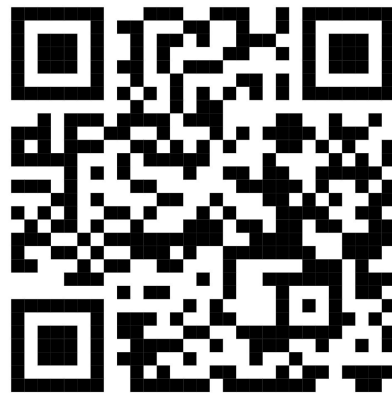




Google Developer Group  
Incheon




장시간 들어주셔서  
감사합니다.



링크드인



{ Build  with AI }