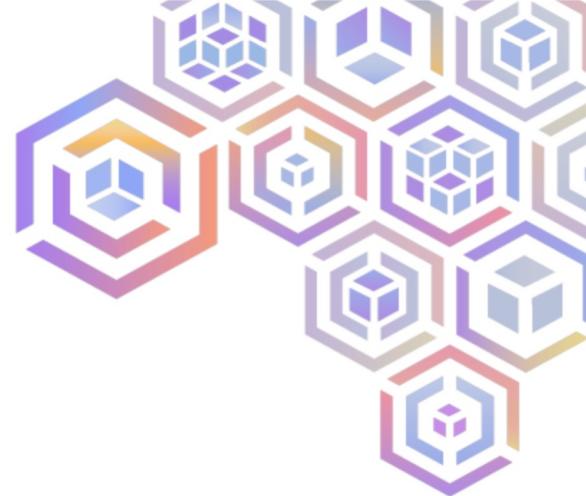
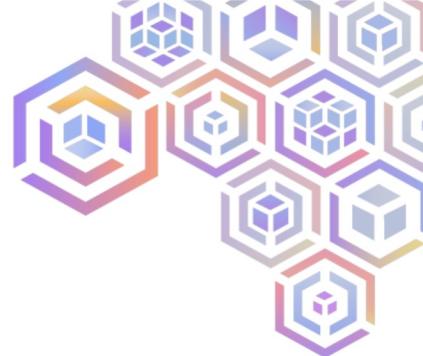


# Prompt Engineering & Bedrock Prompt Flows

장문기 | SK C&C



# 발표자 소개



## 장문기

- SK Inc. C&C | Cloud Architect
- Kubernetes와 Network를 많이 다루는 SRE 수행.
- 업무와 무관하게 Data와 생성형 AI 관련 AWS 서비스에 관심이 많습니다.
- 최애 서비스는 AWS Lambda와 Amazon Athena.



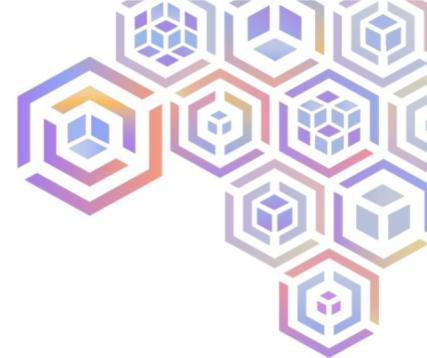
LinkedIn



# News : Anthropic AI Agent

Anthropic's 컴퓨터를 에이전트가 제어하는 "Computer Use"

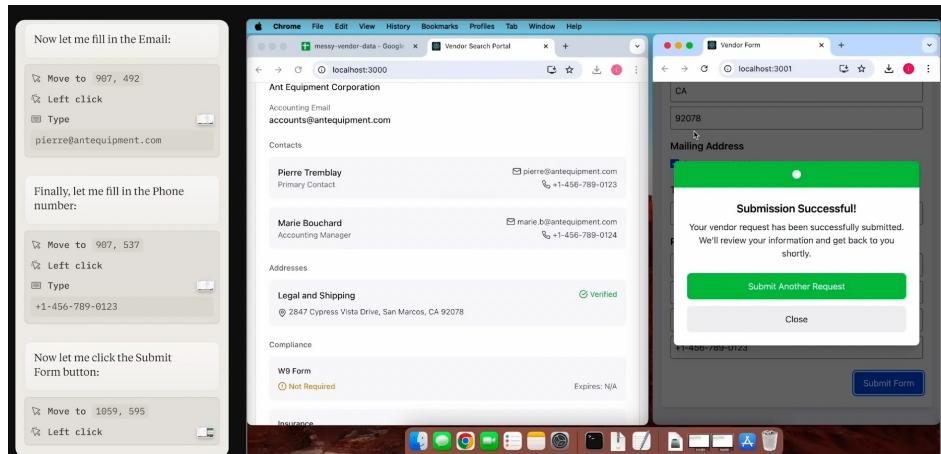
2024년 10월 23일



사용자의 지시를 수행하기 위해 PC를 직접 제어, 화면을 캡처하고 분석하면서  
Mouse Keyboard Input, Command 실행등을 수행

이제까지 검색, API 수준의 요청을 처리했다면, PC에 대한 제어권 전체를  
가지고 사용자의 지시를 수행, 아직은 느리고 불완전

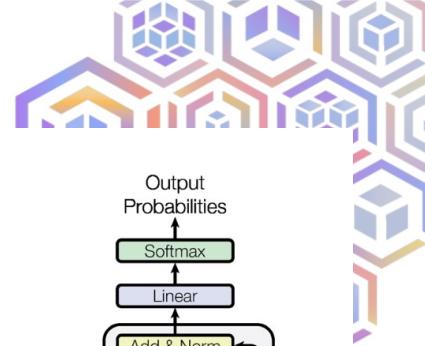
Bedrock에서 API 지원!



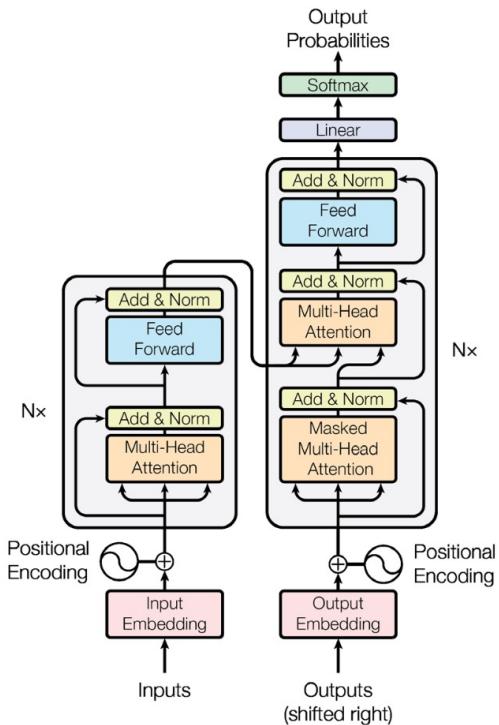
ANTHROPIC'S NEW AI CAN CONTROL APPS ON A PC.



# LLM(Large Language Model)



- LLM은 대량의 텍스트 데이터를 기반으로 학습되어 자연어 이해와 생성에 관한 복잡한 작업을 수행
- 대부분의 LLM은 Transformer 기반 아키텍처를 채용
  - GPT 계열
    - Transformer 아키텍처의 Encoder 부분
    - 문장 생성 특화
    - 자기회귀(auto-regressive) 모델
  - BERT 계열
    - Transformer 아키텍처의 Decoder 부분
    - 문장 맥락 이해 특화
    - 자기인코딩(auto-encoding) 모델

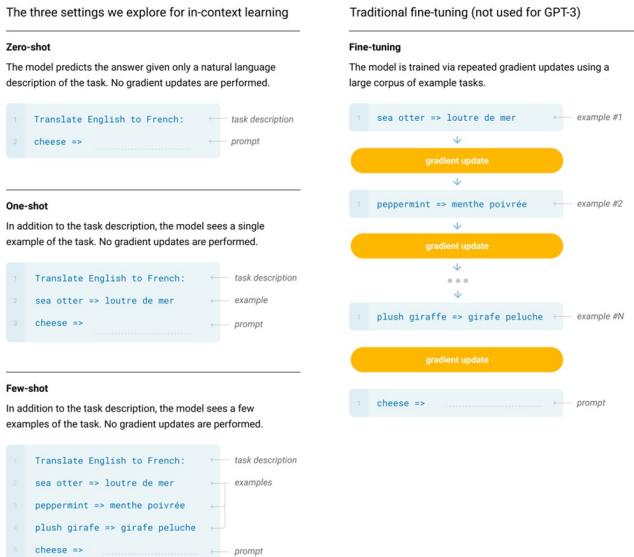


Transformer architecture  
: encoders and decoders

# Zero-Shot (ZS) and Few-Shot (FS) In-Context Learning



- 모델이 매우 적은 수의 예제를 통해 특정 작업을 수행할 수 있는 능력
- 모델이 주어진 몇 개의 예제를 "학습 컨텍스트"로 사용해 올바른 응답을 생성하도록 배움



**Figure 2.1: Zero-shot, one-shot and few-shot, contrasted with traditional fine-tuning.** The panels above show four methods for performing a task with a language model – fine-tuning is the traditional method, whereas zero-, one-, and few-shot, which we study in this work, require the model to perform the task with only forward passes at test time. We typically present the model with a few dozen examples in the few shot setting. Exact phrasings for all task descriptions, examples and prompts can be found in Appendix G.

## Zero-Shot

- 예시 없이 직접 추론
- 사전 학습만으로 새로운 작업 수행

## One-Shot

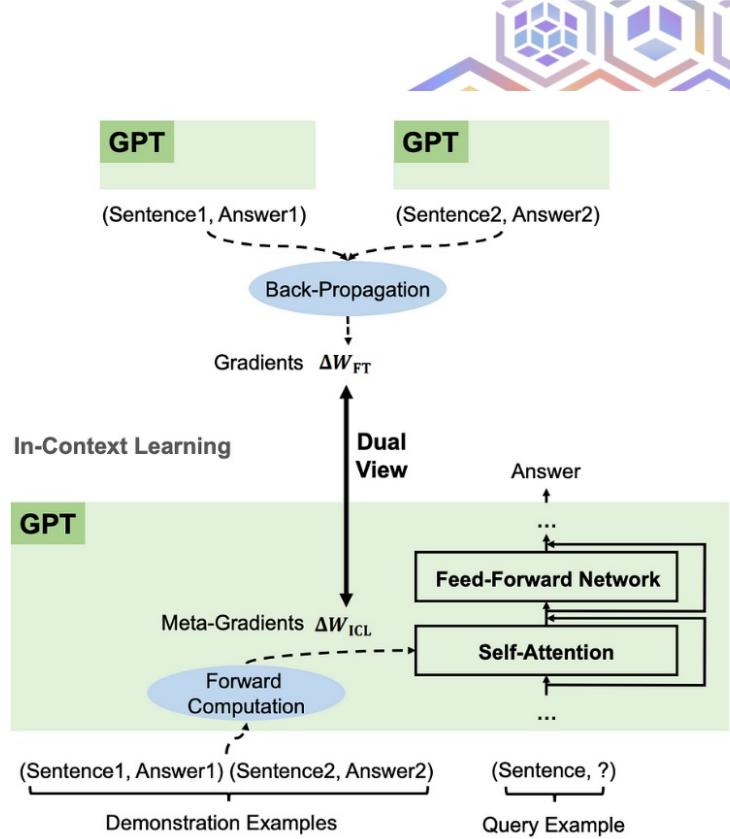
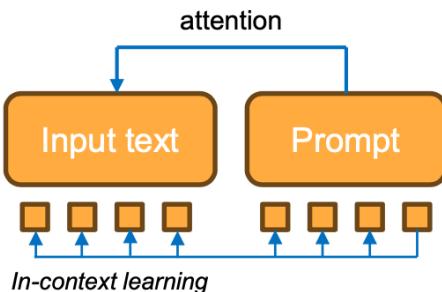
- 하나의 예시로 학습
- 단일 예시를 통한 패턴 학습

## Few-Shot

- 소수의 예시로 학습
- 2-3개 이상의 예시로 패턴 학습

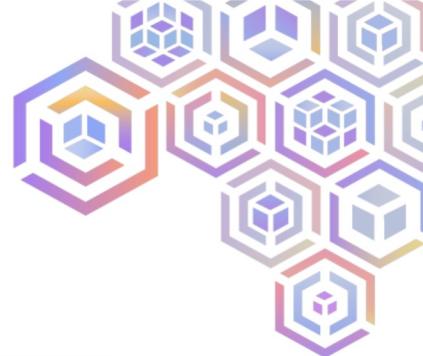
# In-context learning

- 대규모 사전학습 LLM은 언어 이해 및 생성 뿐만 아니라 In-context learning과 추론 능력까지 향상
- CNN, RNN, FCN 등과 달리 Transformer는 입력 데이터에 대한 동적 가중치 계산이 가능
- 방대한 데이터에 의한 meta learning과 함께 동적 가중치 계산이 In-context learning을 설명한다는 연구 존재.

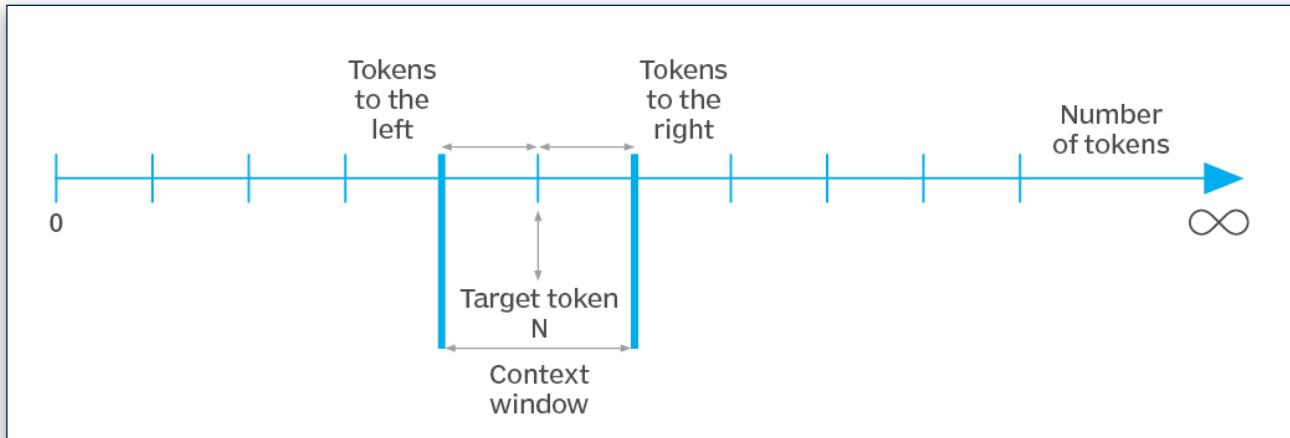


Dai, Damai, et al. "Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers." arXiv preprint arXiv:2212.10559 (2022).

# AI는 대화를 기억하지 못한다.



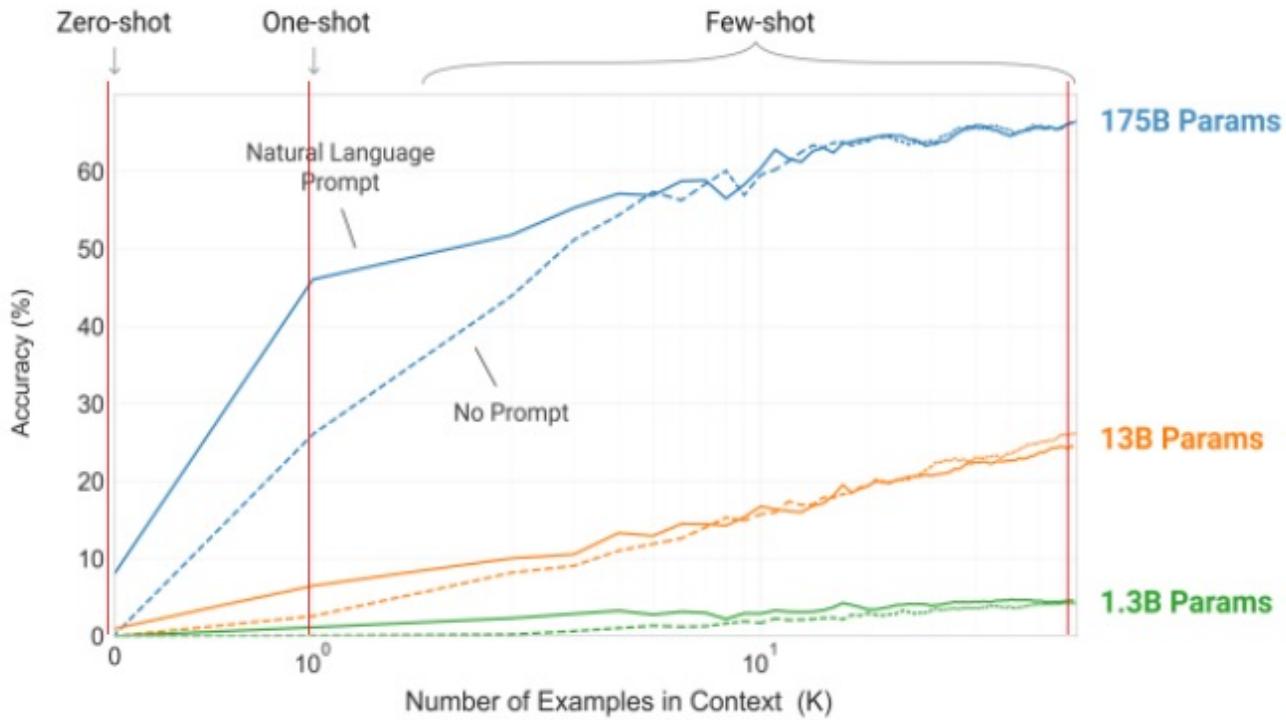
- Context Window : 모델이 한 번에 처리하거나 생성할 수 있는 텍스트의 양
- 토큰 제한(Token Limit)이 크면 모델이 긴 텍스트를 처리할 수 있는 반면, 제한이 작으면 짧은 텍스트로 제한됨.



과거 데이터를 요약하는 방법도 있다.



# 모델이 클수록, 예시가 늘어 날수록 정확도는 올라간다.



# Inference Configuration Parameters



언어 모델이 텍스트를 생성할 때의 다양성과 품질을 제어하는 핵심 설정 매개변수들로, 출력 토큰의 샘플링 방식을 결정

## Temperature

정의: 확률 분포의 **무작위성** 정도  
범위: 0.0 ~ 1.0  
•낮음 (0.1-0.3):  
정확성 ↑, 일관성 ↑  
수학, 코딩에 적합  
•높음 (0.7-0.9):  
창의성 ↑, 다양성 ↑  
창작, 브레인스토밍에 적합

## Top P

정의: **누적 확률** 임계값 기반 샘플링  
범위: 0.0 ~ 1.0  
•누적 확률 기반 선택  
•동적 선택 범위  
•낮음 (0.1-0.3): 안정적 출력  
•높음 (0.9-0.95): 다양한 출력

## Top K

정의: **상위 K개** 토큰으로 선택 제한  
선택: 상위 K개 토큰  
•고정 선택 범위  
•낮음 (10-20): 보수적  
•높음 (50-100): 창의적  
•문맥 독립적 작동



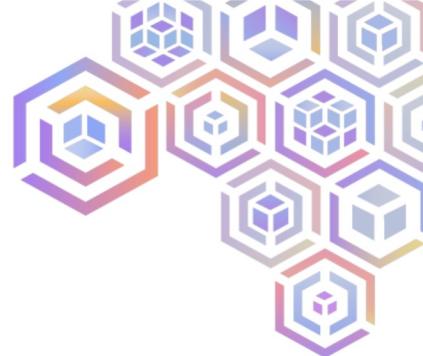
## 권장 조합

정확성 중심  
Temp: 0.1-0.3  
Top P: 0.1-0.3  
Top K: 10-20

균형적 생성  
Temp: 0.5  
Top P: 0.7-0.8  
Top K: 30-50

창의성 중심  
Temp: 0.7-0.9  
Top P: 0.9-0.95  
Top K: 50-100

# Prompt Engineering



Prompt engineering은 LLM이 정확, 간결, 창의적인 텍스트 생성 작업을  
수행하도록 지시하는 실천적 가이드라인  
(엄밀히 공학분야나 수학적 기초에 근거한 것은 아님.)

GPT 모델은 계산된 확률값을 가지고 다음 토큰을 추천  
앞의 말을 가지고 다음에 올 말을 추천

GPT is a model that recommends the next token with the calculated probability value.



# Guides – AWS Bedrock

AWS > aws-documentation > Amazon Bedrock > 사용자 가이드

▶ 시작하기

- ▶ 파운데이션 모델 액세스
- ▶ 파운데이션 모델 정보
- ▶ 모델 추론을 사용하여 프롬프트를 제출하고 응답을 생성합니다.
- ▼ 프롬프트 엔지니어링 개념

## 프롬프트란 무엇인가요?

PDF | RSS

프롬프트는 Amazon BedrockLLMs에서 지정된 작업 또는 지침에 대한 적절한 응답 또는 출력을 생성하도록 안내하는 사용자인 사용자가 제

### User Prompt:

*Who invented the airplane?*

이 프롬프트에서 쿼리할 때 Titan은 다음과 같은 출력을 제공합니다.

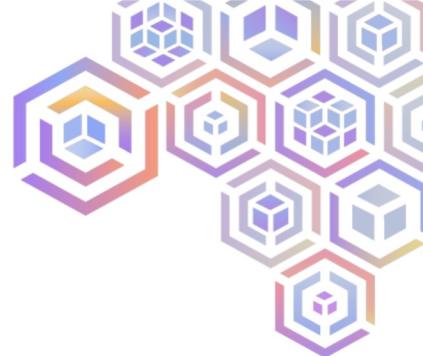
### Output:

*The Wright brothers, Orville and Wilbur Wright are widely credited with inventing and manufacturing the world's first successful airplane.*

(프롬프트 소스: AWS, 사용된 모델: Amazon Titan 텍스트)

주제

- [프롬프트의 구성 요소](#)
- [퓨처 프롬프팅 vs. 제로샷 프롬프팅](#)
- [프롬프트 템플릿](#)
- [Amazon Bedrock 추론 요청에 대한 리콜 유지](#)

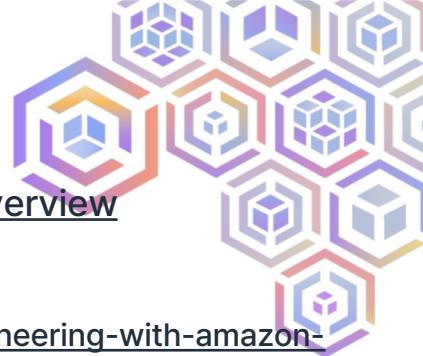


## 에이전트 고급 프롬프트 템플릿

[https://docs.aws.amazon.com/ko\\_kr/bedrock/latest/userguide/configure-advanced-prompts.html](https://docs.aws.amazon.com/ko_kr/bedrock/latest/userguide/configure-advanced-prompts.html)



# Guides



- **Anthropic Claude model prompt guide:**  
<https://docs.anthropic.com/en/docs/build-with-claude/prompt-engineering/overview>
- **Amazon Bedrock**을 사용하여 고급 프롬프트 엔지니어링 구현  
<https://aws.amazon.com/ko/blogs/machine-learning/implementing-advanced-prompt-engineering-with-amazon-bedrock/>
- **Open AI Prompt engineering**  
<https://platform.openai.com/docs/guides/prompt-engineering>
- **Cohere prompt guide:**  
<https://txt.cohere.com/how-to-train-your-pet-lm-prompt-engineering>
- **AI21 Labs Jurassic model prompt guide:**  
<https://docs.ai21.com/docs/prompt-engineering>
- **Meta Llama 2 prompt guide:**  
<https://ai.meta.com/llama/get-started/#prompting>
- **Stability AI prompt guide:**  
<https://platform.stability.ai/docs/getting-started>
- **Mistral AI prompt guide:**  
[https://docs.mistral.ai/guides/prompting\\_capabilities/](https://docs.mistral.ai/guides/prompting_capabilities/)
- **Elvis Saravia**의 프롬프트 엔지니어링 가이드  
<https://www.promptingguide.ai/introduction/elements>

# Guides : Papers

Sahoo, Pranab, et al. "A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications." arXiv preprint arXiv:2402.07927 (2024).

<https://arxiv.org/pdf/2402.07927>

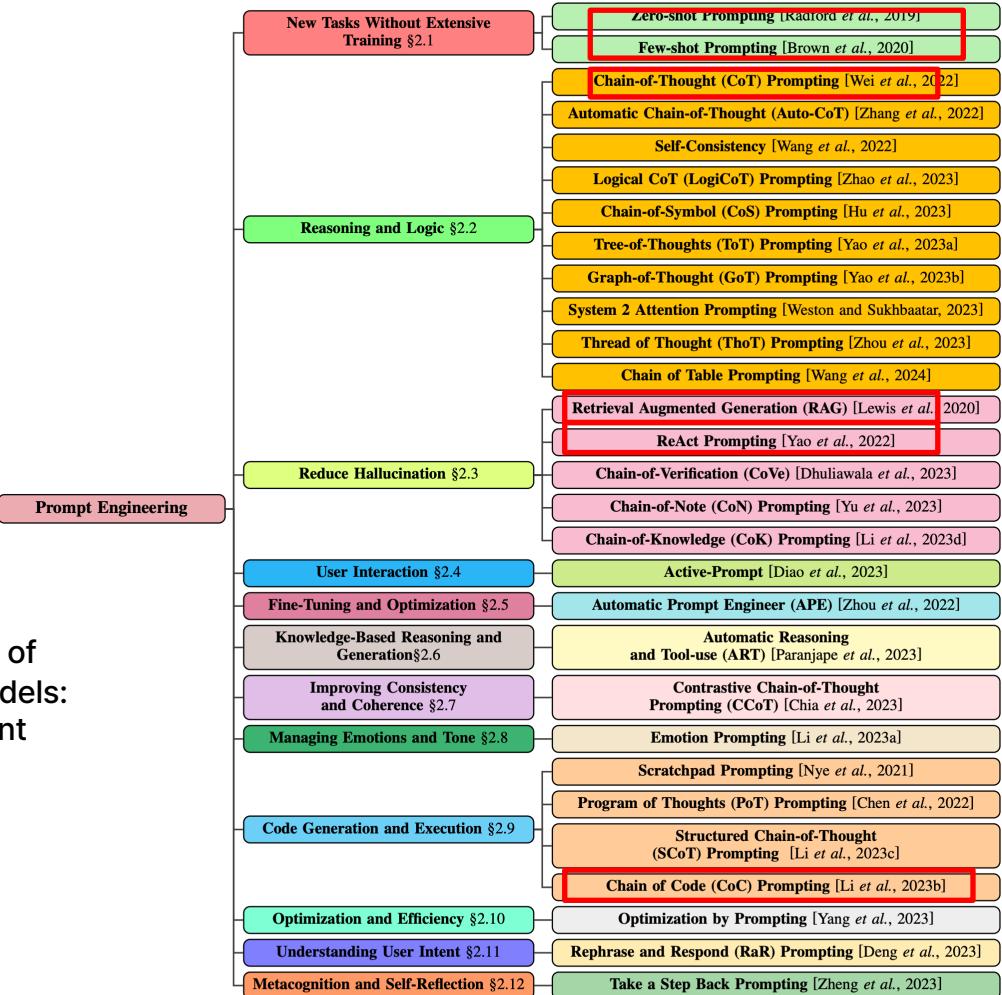
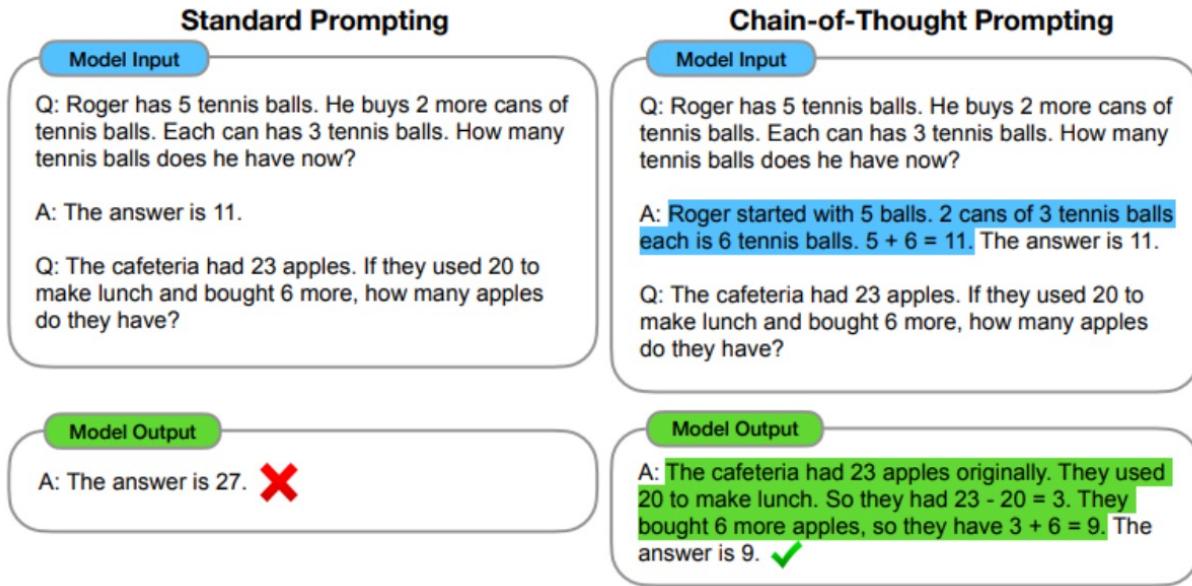


Figure 2: Taxonomy of prompt engineering techniques in LLMs, organized around application domains, providing a nuanced framework for customizing prompts across diverse contexts.

# Chain-of-thought(COT)



LLM이 복잡한 문제를 해결하기 위해 주어진 문제를 단계별로 분해하고, 추론을 연쇄적으로 이어가며 최종 결론에 도달하도록 하는 prompt-based approach



# Chain-of-thought(CoT)



(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. X

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are  $16 / 2 = 8$  golf balls. Half of the golf balls are blue. So there are  $8 / 2 = 4$  blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

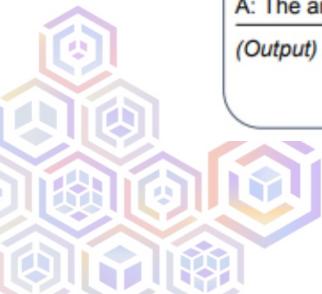
(Output) 8 X

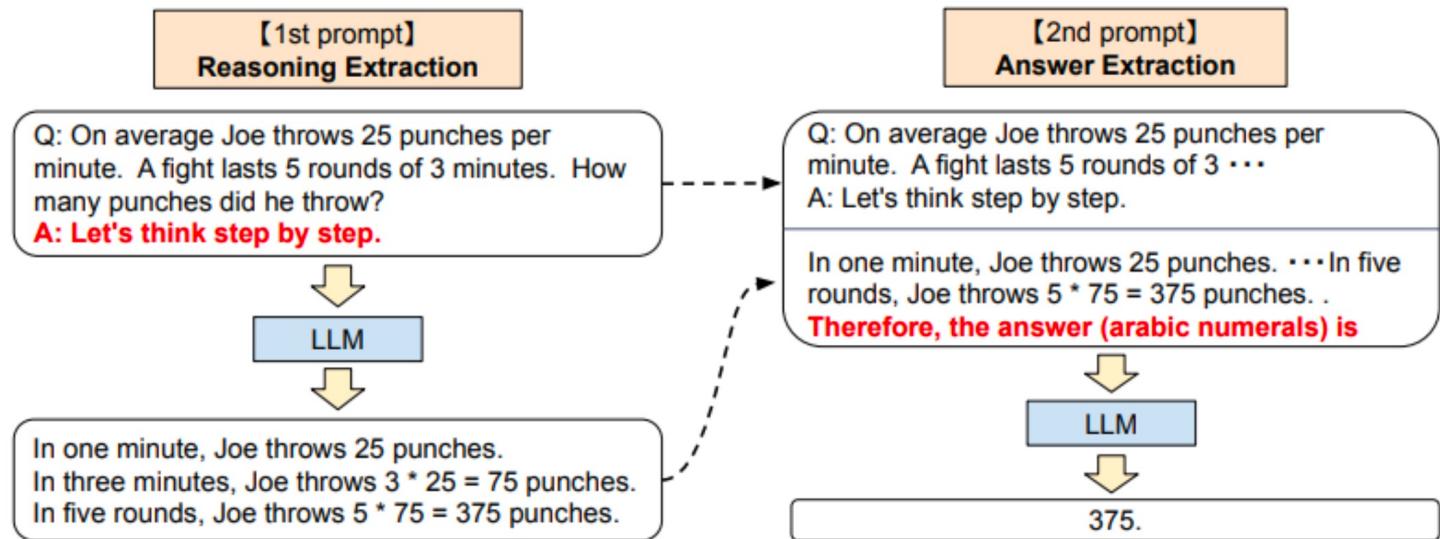
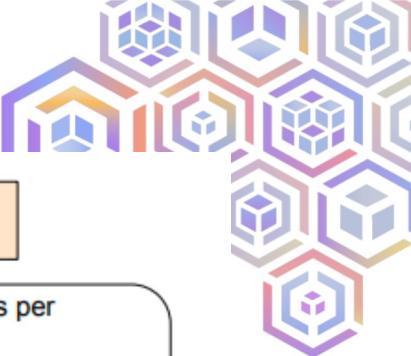
(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓





## Zero-shot COT는 두 단계로 구성

- 첫 번째는 언어 모델이 전체 추론 경로를 추출하는 “추론 프롬프트 추출”
- 두 번째는 추론 텍스트에서 올바른 형식의 답변을 추출하는 “답변 프롬프트 추출”



# Principled Instructions Are All You Need for Questioning LLaMA-1/2, GPT-3.5/4" <https://arxiv.org/pdf/2312.16171.pdf>



## 5가지 범주의 총 26가지 지시 원칙

카테고리	프롬프트 지시 원칙 요약	원칙 ↑ ↘	카테고리	프롬프트 지시 원칙 요약	원칙 No.
프롬프트 구조와 명확성	청자의 페르소나 설정하기	2	사용자 인터랙션과 참여	정보가 충분할 때까지 역질문하라고 요구하기	14
	부정어 쓰지 않고 긍정어 쓰기	4		필요한 모든 세부 정보를 추가하라고 요구하기	21
	step by step으로 생각하라고 지시하기	12		텍스트를 개선해달라고 요구하기	22
	Output 템플릿 지정하기	20	콘텐츠와 언어 스타일	임무를 부여하기	9
	구분 기호 사용하기 (예: 쉼표,따옴표,세미콜론,콜론 등)	17		협박하기	10
구체성과 정보	마크다운을 사용하여 '지시/예시/질문/콘텐츠'를 구분하기	8		역할을 부여하기	16
	몇 가지 예시를 제공하기 (Few-shot prompting)	7		"자연스럽고 인간적인 방식으로"라는 문구 추가하기	11
	명확하게 또는 깊이 알고 싶을 때는 어린이 청자로 설정하기	5		바로 본론부터 말하기	1
	고정관념을 피하라고 지시하기	13		중요한 단어나 구는 여러 번 반복하기	18
	제공한 샘플과 유사한 결과를 얻고 싶을 때는 동일한 언어 사용을 요구하기	26		좋은 답변을 주면 팁을 준다고 가정하기	6
복잡한 작업과 코딩 프롬프트	제공한 내용에 이어서 쓰라고 요구하기	24	복잡한 작업을 간단한 프롬프트들로 분할하기		
	키워드, 지침 등의 형태로 요구사항을 명확히 제시하기	25			
	어떤 개념에 대해 질문하면서 이해도까지 테스트하고 싶다면 테스트를 포함해달라고 요구하기	15		필요한 파일을 생성할 수 있는 스크립트 요구하기	23
	필요한 모든 세부 정보를 추가하라고 요구하기	21		Chain-of-Thought와 Few-shot prompting 기법을 함께 사용하기	19

# CO-STAR prompt framework



CO-STAR prompt framework는 더 나아가 실용적 관점에서 프롬프트 가이드라인을 단순화하고 결정화함.  
Sheila Teo의 CO-STAR은 싱가포르 GPT-4 프롬프트 엔지니어링 경쟁대회에서 우승

"Principled Instructions Are All You Need for Questioning LLaMA-1/2, GPT-3.5/4"  
논문 원칙을 단순화하여 다음과 같은 항목을 제시

C: Context: Provide background and information on the task

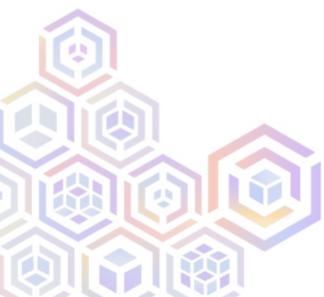
O: Objective: Define the task that you want the LLM to perform

S: Style: Specify the writing style you want the LLM to use

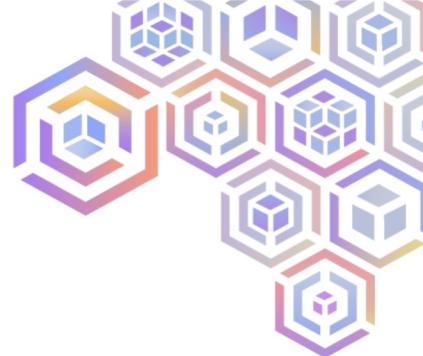
T: Tone: Set the attitude and tone of the response

A: Audience: Identify who the response is for

R: Response: Provide the response format and style



# CO-STAR prompt framework



## C (Context, 맥락)

- 과제나 요청의 배경 정보를 제공
- 관련된 중요한 세부사항이나 제약사항 설명
- 배경 정보를 제공하면 LLM이 구체적인 시나리오를 이해할 수 있습니다.
- 예: "20대 여성 타겟 의류몰의 봄 신상품 설명이 필요합니다."

## O (Objective, 목표)

- 수행하고자 하는 내용을 구체적으로 설명
- 명확하고 측정 가능한 목표 설정
- 업무를 명확하게 정의하면 LLM의 초점이 맞춰집니다.
- 예: "원단 품질과 맞춤 디자인의 장점을 강조해주세요"

## S (Style, 스타일)

- 글쓰기 스타일 : 출력의 특성에 대한 자세한 정보를 제공
- 문체, 형식, 구조 등을 구체화
- 원하는 글쓰기 스타일을 지정하면 LLM 응답이 정렬됩니다.
- 예: "패션 매거진 에디터의 세련된 문체로 작성해주세요"

## T (Tone, 어조)

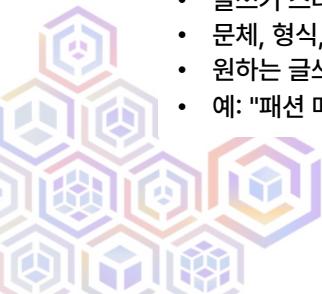
- 응답의 태도와 분위기 설정
- 전문적, 친근함, 격식 등의 어조 지정
- 톤을 설정하면 응답이 필요한 감정과 공감되도록 할 수 있습니다.
- 예: "고급스럽고 친근한 톤으로 작성해주세요"

## A (Audience, 독자)

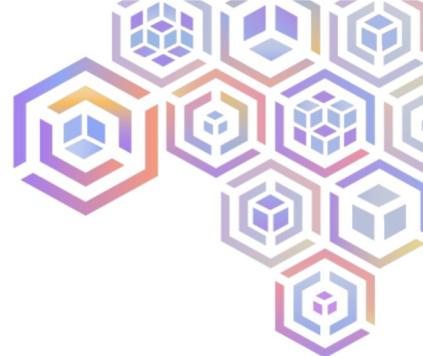
- 응답의 대상 독자 명시
- 독자의 특성, 지식수준, 관심사 등 파악
- 의도한 청중을 식별하면 LLM의 응답도 청중에 맞게 조정됩니다.
- 예: "패션 관심 많은 20대 직장여성을 위해 작성해주세요"

## R (Response, 응답형식)

- AI 답변의 형식과 구조를 지정
- 글자 수, 단락 구성, 포함할 요소 등 명시
- 텍스트나 JSON과 같은 응답 형식을 제공하면 LLM 출력이 보장되고 파이프라인을 구축하는 데 도움이 됩니다.
- 예: "3000자로 특징, 소재, 스타일링 팁을 포함해주세요"



# CO-STAR prompt framework



예시 : 블로그 글 작성 요청

## Context:

- 건강식품 전문 온라인 쇼핑몰 운영 중
- 최근 젊은층의 건강식품 관심도 증가 추세
- 면역력 증진 제품군이 인기 상승 중

## Objective:

- 20-30대를 위한 면역력 증진 건강식품 소개 블로그 글 작성
- 주요 영양성분과 효능 설명
- 일상생활에서 실천할 수 있는 면역력 관리 팁 포함

## Style:

- 과학적 근거를 바탕으로 한 설명
- 실제 사용자 경험 사례 포함
- 간단한 인포그래픽 요소 추가

## Tone:

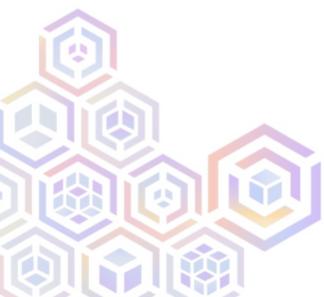
- 친근하고 편안한 어조
- 전문성을 유지하되 어려운 용어는 쉽게 풀어서 설명
- 긍정적이고 활기찬 분위기

## Audience:

- 20-30대 직장인과 대학생
- 건강에 관심이 있으나 전문지식은 부족한 층
- 바쁜 일상 속 실천 가능한 방법을 찾는 사람들

## Response:

- 1500자 내외의 블로그 포스트
- 소제목과 단락 구분 명확하게
- 핵심 포인트는 bullet point로 정리
- 실천 가능한 팁 5가지 포함



# CO-STAR prompt framework

예시 : 청소년을 위한 인터넷 안전 가이드

## # Context

저는 청소년을 위한 교육 자료를 제작하는 사람입니다. 최근 청소년들이 인터넷을 안전하게 사용하고, 개인정보를 보호하며, 사이버 괴롭힘 등 온라인 위험으로부터 스스로를 지킬 수 있도록 돕는 가이드에 대한 필요성이 증가하고 있습니다.

## # Objective

청소년들이 쉽게 이해하고 따라할 수 있는 인터넷 안전 가이드를 만드는 것이 목표입니다. 이 가이드에는 개인정보 보호, 의심스러운 링크나 메시지를 피하는 법, 안전한 비밀번호 설정, 그리고 온라인에서의 존중과 책임감 있는 행동에 대한 조언이 포함되어야 합니다.

## # Style

쉽고 명확한 정보 전달을 위해 간단한 언어로 작성해 주세요. 청소년 독자들이 쉽게 읽고 실천할 수 있도록 짧고 직관적인 설명을 사용해 주세요.



LLM에게 CO-STAR원칙을 알려주고,  
주제를 제시한 후 프롬프트를 작성해달라고 하자!

## # Tone

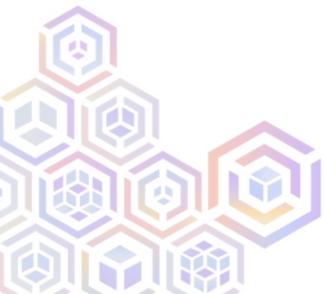
긍정적이고 친근한 어조를 유지하여 청소년들이 안전한 인터넷 사용법을 배우는 과정에서 흥미와 관심을 가질 수 있도록 해주세요. 마치 친구가 조언을 주는 듯한 따뜻한 느낌을 주는 것이 중요합니다.

## # Audience

대상 독자는 13세에서 18세 사이의 청소년들입니다. 이들은 주로 스마트폰과 소셜 미디어를 활발히 사용하는 세대로, 디지털 안전에 대한 실질적인 조언을 필요로 합니다.

## # Response Format

단계별로 정리된 안전 수칙 목록을 제공해 주세요. 각 수칙은 주제를 짧게 설명하고, 간단한 예시나 실천 방법을 포함하여 청소년들이 쉽게 이해하고 행동에 옮길 수 있도록 해주세요.



# OpenAI의 GPT 모범 사례

## - 더 나은 답변을 얻기 위한 6가지 프롬프트 작성 전략

<https://platform.openai.com/docs/guides/prompt-engineering/strategy-test-changes-systematically>



### 1. 명확한 지시사항 작성

- 모델이 마음을 읽을 수 없음을 인지
- 원하는 출력 길이, 전문성 수준, 형식 등을 구체적으로 명시
- 모델이 추측해야 할 부분을 최소화

예시

- '엑셀에서 숫자를 더하세요' (X)
- '엑셀의 A열에 있는 모든 숫자를 더해서 B1 셀에 표시해주세요' (O)

### 2. 참조 텍스트 제공

- 신뢰할 수 있는 정보를 제공하여 허구 답변 방지
- 특히 전문적인 주제나 인용이 필요한 경우에 중요
- 학생이 시험에서 참고 자료를 보는 것과 유사

예시

- 논문 요약 시 원본 텍스트 제공하여 정확도 향상

# OpenAI의 GPT 모범 사례

## - 더 나은 답변을 얻기 위한 6가지 프롬프트 작성 전략



<https://platform.openai.com/docs/guides/prompt-engineering/strategy-test-changes-systematically>

### 3. 복잡한 작업을 단순한 하위 작업으로 분할

- 소프트웨어 엔지니어링의 모듈화 원칙과 유사
- 복잡한 작업은 오류 발생 가능성이 높음
- 이전 작업의 출력을 다음 작업의 입력으로 활용

예시

- 논문 작성 → 개요 작성 → 각 섹션 작성 → 교정

### 4. 모델에게 '생각할 시간' 제공

- 즉각적인 답변보다 단계적 추론 과정 요청
- 사고 과정을 명시적으로 표현하도록 유도
- 더 신뢰성 있는 답변 도출 가능

예시

- 수학 문제 → 1)문제 분석 2)해결 방법 선택 3)단계별 풀이

# OpenAI의 GPT 모범 사례

## - 더 나은 답변을 얻기 위한 6가지 프롬프트 작성 전략

<https://platform.openai.com/docs/guides/prompt-engineering/strategy-test-changes-systematically>



### 5. 외부 도구 활용

- 모델의 약점을 보완하기 위한 외부 도구 활용
- 텍스트 검색 시스템(RAG) 및 코드 실행 엔진 활용
- 외부 API 호출 기능 활용

예시

수학 계산 → Python/Calculator 사용 데이터 검색 →  
Vector DB 활용

### 6. 체계적인 변경 사항 테스트

변경 사항의 효과를 측정하기 위한 평가 체계 구축  
대표성 있는 테스트 케이스 구성  
자동화된 평가 시스템 구축

예시

A/B 테스트로 프롬프트 성능 비교 사용자 피드백 수집 및  
분석

# Anthropic 모범 사례 - 고급 프롬프트 엔지니어링 기법

<https://docs.anthropic.com/ko/docs/build-with-claude/prompt-engineering/overview>

한글로 잘 번역되어 있고, 프롬프트 엔지니어링 파트는 15분이면 볼 수 있으니 꼭 읽어보세요.



## 1. 명확하고 직접적이며 상세하게

### 주요 원칙

- 구체적인 지시사항 제공
- 원하는 출력 형식 명시
- 맥락 정보 포함

### 예시

✗ "이 텍스트를 분석해주세요"

✓ "이 고객 피드백을 분석하여 주요 문제점을 찾고,

감정(긍정/부정/중립)을 분류하며,

우선순위(상/중/하)를 지정해주세요"

## 2. 멀티샷 프롬프팅(예시 사용)

### 주요 원칙

- 3-5개의 다양한 예시 제공
- 원하는 출력 형식 시연
- 관련성, 다양성, 태그를 사용한 명확성

### 예시

<example>

입력: "배송이 너무 늦어요!"

분류: 불만족

카테고리: 배송

우선순위: 높음

</example>

<example>

입력: "품질이 좋네요"

분류: 만족

카테고리: 제품

우선순위: 중간

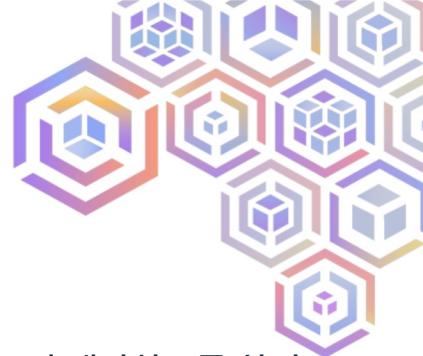
</example>

동료에게 프롬프트를 보여주세요. 가급적이면 해당 작업에 대한 맥락을 최소한으로 알고 있는 사람에게 보여주고 지시사항을 따라해보라고 하세요.

그들이 혼란스러워한다면,  
LLM도 마찬가지일 것입니다.

# Anthropic 모범 사례

## - 고급 프롬프트 엔지니어링 기법



### 3. XML 태그 활용

#### 구조화 전략

- 구조화된 데이터 전달
- 명확한 섹션 구분
- 프롬프트의 다른 부분을 명확하게 구분

#### 예시

```
<document>
  <source>customer_feedback.txt</source>
  <category>Product Review</category>
  <content>
    제품 품질이 매우 좋습니다.
    다만 배송이 조금 늦었네요.
  </content>
</document>
```

태그 이름과 둘러싸고 있는  
정보와 의미가 통하도록  
해야함.  
태그의 패턴과 내용을  
인식.

### 4. (CoT)사고 연쇄 프롬프팅 (LLM이 생각하도록 하기)

#### 단계별 분석

- 복잡한 문제를 단계별로 분석
- 추론 과정 명시적 표현
- 오류 감소와 정확성 향상

#### 예시

```
<thinking>
  1. 주요 증상 분석:
    - 발열: 38.5도
    - 인후통
    - 기침
  2. 증상 패턴 확인:
    - 급성 발병
    - 상기도 증상 동반
  3. 가능한 진단:
    - 감기 바이러스 감염
    - 독감 의심
</thinking>
```

사고 과정을 출력하도록 지시해야함.  
출력하지 않으면 실제 사고가  
일어나지 않음.

# Anthropic 모범 사례

## - 고급 프롬프트 엔지니어링 기법



### 5. 시스템 프롬프트로 역할 부여하기

#### 전문가 페르소나

- 명확한 특정 전문가 역할 부여
- 전문 지식 영역 지정
- 향상된 정확도, 맞춤형 톤, 향상된 집중도

#### 예시

당신은 10년 경력의 데이터 사이언티스트입니다.  
다음 데이터셋을 분석하고 비즈니스 인사이트를  
도출해주세요:

```
<data>  
[데이터셋]  
</data>
```

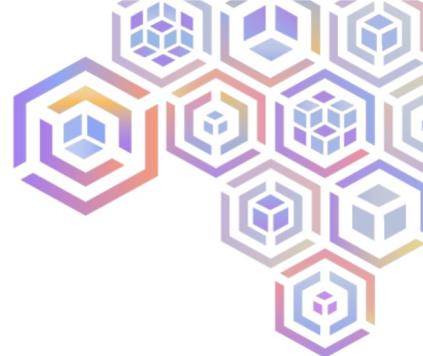
### 6. 더 나은 출력 제어를 위한 응답 미리 채우기

괄호로 둑인 [ROLE\_NAME]을 미리  
채우면 더 길고 복잡한 대화에서도  
캐릭터를 유지가능,  
특히 system 매개변수의 역할  
프롬프팅에 결합!

{를 미리 채우면 Claude가 서문을  
건너뛰고 JSON 객체를 직접  
출력하도록 강제

# Anthropic 모범 사례

## - 고급 프롬프트 엔지니어링 기법



### 7. 긴 컨텍스트 처리

#### 최적화 전략

- 긴 형식의 데이터를 상단에 배치
- xml태그로 문서 및 메타데이터 구조화
- 인용구로 응답 근거 제시

#### 예시

```
<documents>
<document index="1">
<source>report.pdf</source>
<content>
[주요 내용]
</content>
</document>
<quotes>
"관련된 중요 인용구"
"증거가 되는 문장"
</quotes>
</documents>
```

복잡한 다중 문서 입력의 경우,  
끝부분에 질문을 배치하면 응답  
품질이 최대 30%까지 향상

### 8. 프롬프트 체이닝

#### 단계별 처리

- 복잡한 작업을 하위 작업으로 분할
- 순차적 처리로 정확성 향상
- 각 단계별 품질 관리

#### 예시

Step 1: 데이터 추출  
<extract>고객 불만 사항 목록</extract>

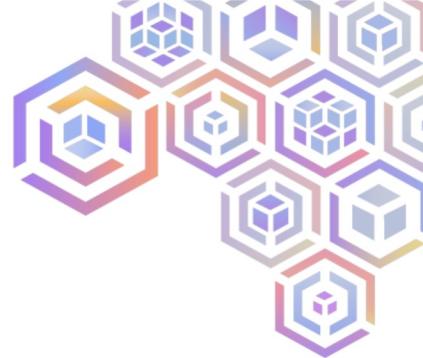
Step 2: 분석  
<analyze>카테고리별 문제점 분석</analyze>

Step 3: 해결책 제시  
<solutions>개선 방안 제안</solutions>

각각 심도 있는 사고가 필요한  
여러 단계가 있는 작업을 더 작고  
관리하기 쉬운 하위 작업으로  
나누는 것

# Anthropic 모범 사례

## - 고급 프롬프트 엔지니어링 기법



### 9. 데이터 분석 시나리오

#### 적용 전략

- 데이터 구조화
- 분석 단계 정의
- 결과 형식 지정

#### 예시

```
<data_analysis>
1. 데이터 클리닝
2. 통계 분석
3. 인사이트 도출
4. 시각화 제작
</data_analysis>
```

#### 결과 형식:

```
{
  "insights": [],
  "recommendations": [],
  "visualizations": []
}
```

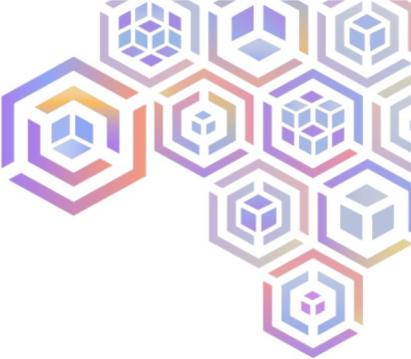
### 10. 콘텐츠 생성 시나리오

#### 작성 가이드

- 톤/보이스 정의
- 타겟 독자 지정
- 구조 템플릿 제공

#### 예시

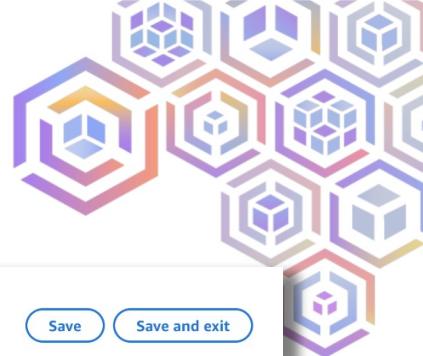
```
<content_brief>
대상: 20-30대 직장인
톤: 전문적이면서 친근한
길이: 800-1000단어
구조:
- 흥미로운 도입
- 주요 포인트 3개
- 실천 가능한 결론
</content_brief>
```



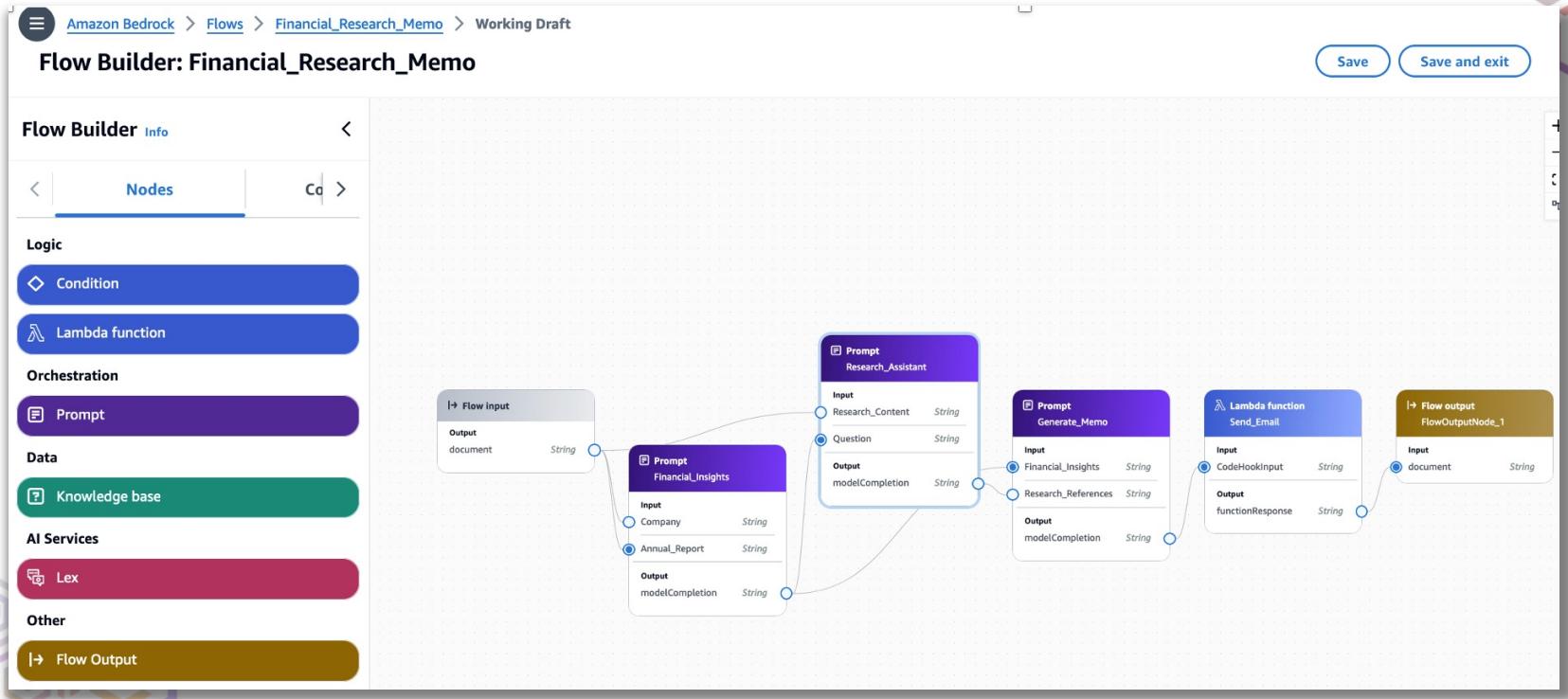
# Amazon Bedrock Prompt Flows



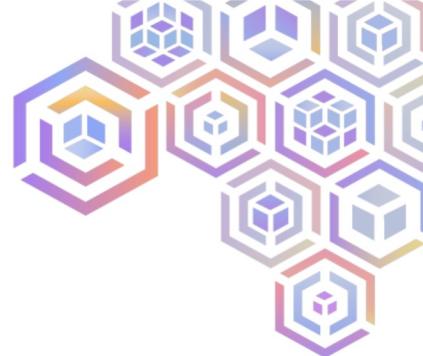
# Amazon Bedrock Prompt Flows



일단은 이렇게 생겼습니다.



# Amazon Bedrock Prompt Flows

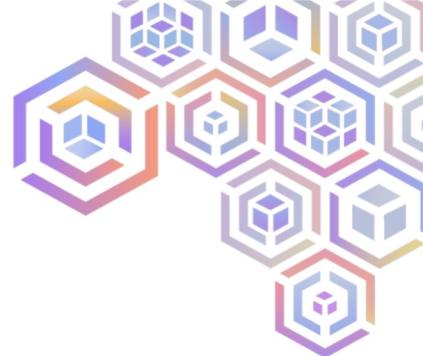


특징은...

- 한마디로 **No Code AI Workflow Tool**
- **비주얼 빌더**를 사용하여 복잡한 생성형 AI 워크플로를 빠르게 생성
- 프롬프트를 다른 프롬프트나 AWS 서비스, 비즈니스 로직과 연결 가능
- Bedrock KB(RAG)), Bedrock Agent , Lambda 등의 AWS 서비스를 연결 가능
- Alias , Version 기능 ⇒ 엔드포인트 유지, 개발/운영 분리, A/B Test 등으로 활용
- Amazon Bedrock Console 또는 Amazon Bedrock Studio에서 사용 가능



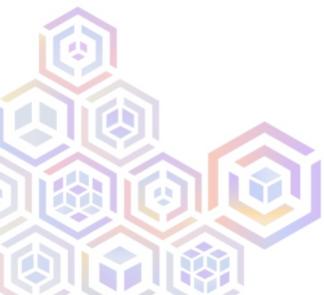
# Amazon Bedrock Prompt Flows



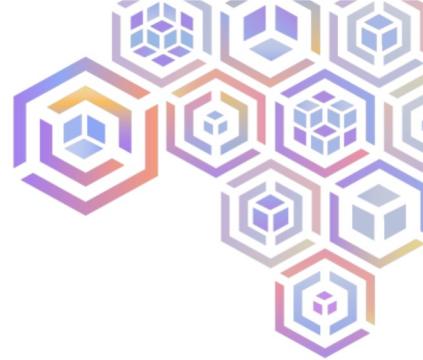
사용 예시를 들어보자면...

## 이메일 초대 생성 및 전송

- 프롬프트 노드, 지식 기반 노드 및 Lambda 함수 노드를 연결하는 프롬프트 흐름을 생성.
- 이메일 본문을 생성 : "클라우드 개발팀에게 다음 주 화요일 오후 2시에 1시간 동안 진행되는 문서 검토 미팅에 대한 초대장을 보내주세요."
- 프롬프트를 처리한 후 프롬프트 흐름은 지식 기반을 쿼리하여 팀원의 이메일 주소를 검색한 다음
- Lambda 함수에 입력을 전송하여 목록에 있는 모든 팀원에게 초대장을 발송.



# Amazon Bedrock Prompt Flows



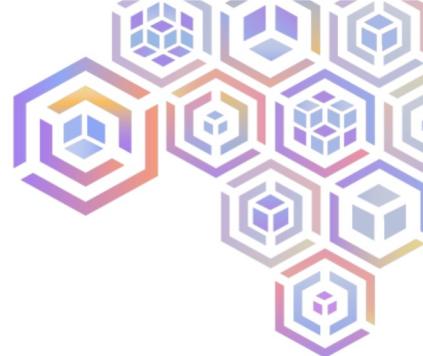
사용 예시를 들어보자면...

보고서 생성 : 제품에 대한 지표를 생성하기 위한 프롬프트 흐름을 구축.

- 데이터베이스에서 판매 지표를 찾고,
- 지표를 집계하고,
- 최상위 제품 구매에 대한 요약 보고서를 생성하고,
- 지정된 포털에 보고서를 게시



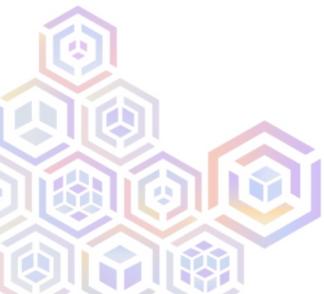
# Amazon Bedrock Prompt Flows



사용 예시를 들어보자면...

오류 메시지와 오류를 유발하는 리소스의 ID를 사용하여 문제 해결

- 설명서 지식 기반에서 오류의 가능한 원인을 검색하고,
- 리소스에 대한 시스템 로그 및 기타 관련 정보를 가져오고,
- 리소스의 잘못된 구성 및 값을 업데이트



# Prompt Flows : 지원 Node 타입



로직을 제어하기 위한 노드

⬇️ 플로우 입력 노드

⬆️ 플로우 출력 노드

✖️ 조건 노드

🔁 이터레이터 노드

📍 컬렉터 노드



데이터를 처리하기 위한 노드

✳️ 프롬프트 노드

🤖 에이전트 노드

λ Lambda 함수 노드

📦 S3 스토리지 노드

🔍 S3 검색 노드

📚 지식 기반 노드

🗣️ Lex 노드

Prompt flow builder 정보 <

Nodes

Configure

Logic

📌 Collector

❖ Condition

🔗 Iterator

Orchestration

💬 Agents

📋 Prompts

Code

Lambda function

Data

Knowledge base

S3 Retrieval

S3 Storage

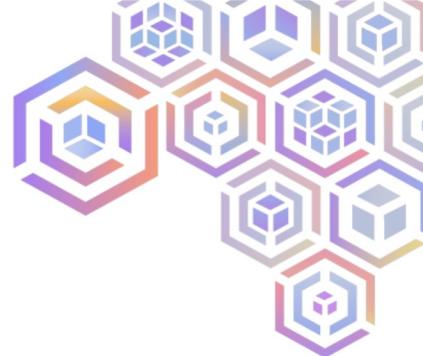
AI Services

💡 Lex

Other

➡️ Flow Output

# Amazon Bedrock Prompt Flows



## Demo – Single Prompt

- 입력은 Object로 해서 JSON 입력을 명시 => 입력 값 형식 체크
- 프롬프트는 입력 변수에 맞게 처리, 출력 요구사항을 명시

### MakePlaylist (genre, number)

`{{genre}}` 장르로 `{{number}}`곡으로 구성된 플레이 리스트를 만들어주세요.  
출력은 다른 내용없이 json으로만 해주세요.  
아래 구조를 사용합니다.

```
{  
    "playlist": "플레이리스트명",  
    "songs":  
        [{ "artist": "가수명",  
            "title": "노래제목",  
            "description": "50자 내외의 설명"  
        }]  
}
```

Name	Type	Expression
genre	String	\$.data.genre
number	Number	\$.data.number

Amazon Bedrock > Prompt flows > Demo01 > Working Draft

## Prompt flow builder: Demo01

[Save](#) [Save and exit](#)

### Prompt flow builder 정보

- [Nodes](#)
- [Configure](#)

#### Prompts 정보

Prompts allow you to create a library of configurable commands handled by LLMs.

[Go to prompts](#)

**Node name**  
MakePlaylist

Valid characters are a-z, A-Z, 0-9 and \_ (underscore). The name can have up to 50 characters.

Use a prompt from your Prompt Management  
 Define in node

**Select model**  
Claude 3 Haiku v1 | On-demand  
번역

**Message**

```
{{(genre)}} 장르로 {{(number)}} 곡으로
구성된 플레이리스트를 만들어주세요.
출력은 다른 내용없이 json으로만 해주세요.
아래 구조를 사용합니다.
{
  "playlist": "플레이리스트명",
  "songs": [
    {
      "artist": "BTS",
      "title": "Dynamite",
      "description": "BTS의 신나는 디스코팝 곡으로
전 세계적으로 큰 사랑을 받았습니다."
    },
    {
      "artist": "ITZY",
      "title": "Wannabe",
      "description": "ITZY의 강렬한 퍼포먼스와 자신
감 넘치는 가사가 둘로보이는 노래입니다."
    },
    {
      "artist": "TXT",
      "title": "OX1=LOVESONG (I Know I Love
You)",
      "description": "TXT의 성장을 잘 보여주는 곡으로,
감성적이면서도 강렬한 사운드가 특징입니다."
    }
  ]
}
```

**Inference configurations**

**Input**

**Output**

**Name**  
modelCompletion

**Type**  
String

```

graph LR
    A[Flow input<br/>document Object] --> B[Prompts<br/>MakePlaylist<br/>genre String<br/>number Number]
    B --> C[Flow output<br/>FlowOutputNode_1<br/>document String]
  
```

**Test Prompt flow 정보**

{ "genre": "kpop", "number": 3 }

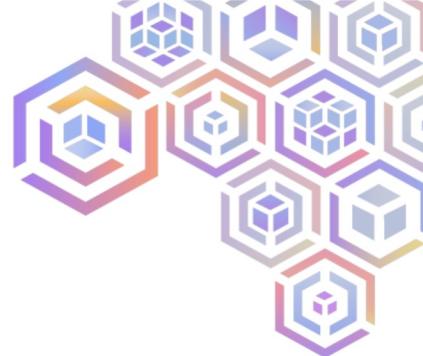
**FlowOutputNode\_1**

```
{
  "playlist": "K-Pop 3.0 Playlist",
  "songs": [
    {
      "artist": "BTS",
      "title": "Dynamite",
      "description": "BTS의 신나는 디스코팝 곡으로
전 세계적으로 큰 사랑을 받았습니다."
    },
    {
      "artist": "ITZY",
      "title": "Wannabe",
      "description": "ITZY의 강렬한 퍼포먼스와 자신
감 넘치는 가사가 둘로보이는 노래입니다."
    },
    {
      "artist": "TXT",
      "title": "OX1=LOVESONG (I Know I Love
You)",
      "description": "TXT의 성장을 잘 보여주는 곡으로,
감성적이면서도 강렬한 사운드가 특징입니다."
    }
  ]
}
```

**Enter your message here**

**실행**

# Amazon Bedrock Prompt Flows



## Demo – 조건 노드

- 입력은 **Object**로 해서 JSON 입력을 명시 ⇒ 입력 값 형식 체크
- 프롬프트는 입력 변수에 맞게 처리, 출력 요구사항을 명시

```
IF (retailPrice < 10) and (type == "고기"):  
    즉시구매  
ELSE IF (retailPrice < marketPrice):  
    안사요  
ELSE  
    결정불가  
  
{  
    "retailPrice": 8,  
    "marketPrice": 11,  
    "type": "고기",  
    "action": ["즉시구매", "안사요", "결정불가"]  
}
```

Name	Type	Expression
retailPrice	Number	\$.data.retailPrice
marketPrice	Number	\$.data.marketPrice
type	String	\$.data.type



## Prompt flow builder: Demo02

Save

Save and exit

Name: type

Type: String

Expression: \$.data.type

**Add input**

## Conditions

Name: Condition

Condition: (retailPrice < 10) and (type == "고기")

Go to node: FlowOutput\_Buy

Name: Condition1

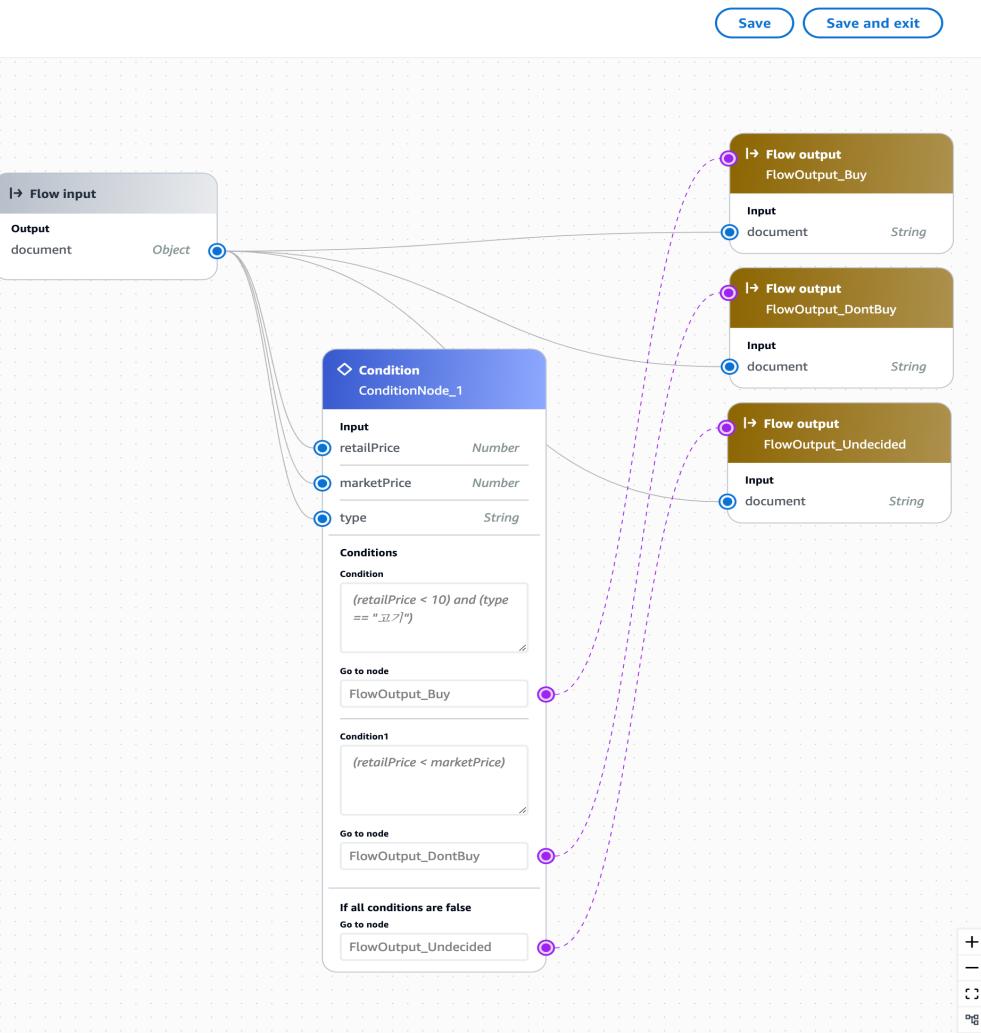
Condition: (retailPrice < marketPrice)

Go to node: FlowOutput\_DontBuy

**Add condition**

## If all conditions are false

Go to node: FlowOutput\_Undecided



Enter your message here

실행

{  
 "retailPrice": 8,  
 "marketPrice": 11,  
 "type": "고기",  
 "action": ["즉시구매", "안사요", "결정불가"]  
}

FlowOutput\_Buy  
즉시구매

{  
 "retailPrice": 8,  
 "marketPrice": 11,  
 "type": "야채",  
 "action": ["즉시구매", "안사요", "결정불가"]  
}

FlowOutput\_DontBuy  
안사요

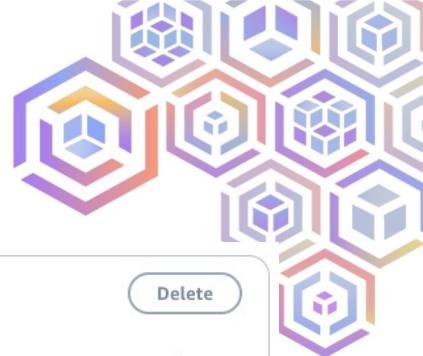
{  
 "retailPrice": 11,  
 "marketPrice": 11,  
 "type": "고기",  
 "action": ["즉시구매", "안사요", "결정불가"]  
}

FlowOutput\_Undecided  
결정불가

# Prompt Flows : Alias와 Version을 지원

필요시점마다 Version을 생성하고

Alias 기능을 통해 Endpoint 변경없이 반영 가능, 여러 환경 구축 가능



## Versions (3)

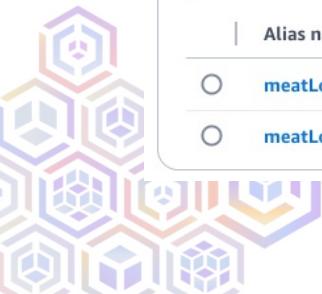
A version is a saved snapshot of your prompt flow. You can create versions and associate aliases with them for deployment.

<input type="radio"/> Version name	Creation date	<input type="button" value="Delete"/>
Version 3	[Redacted]	<input type="button" value="Edit"/>
Version 2	[Redacted]	<input type="button" value="Edit"/>
Version 1	[Redacted]	<input type="button" value="Edit"/>

## Aliases (2)

An alias points to a version of your prompt flow for your client application to use.

<input type="radio"/> Alias name	Alias ID	Description	Associated version	Creation date	Last updated	<input type="button" value="Delete"/>	<input type="button" value="Edit"/>
meatLover-dev	IL353DONKE	<u>개발환경</u>	Version 3	[Redacted]	[Redacted]	<input type="button" value="Edit"/>	<input type="button" value="Edit"/>
meatLover	J6JE1SODFQ	<u>운영환경</u>	Version 2	[Redacted]	[Redacted]	<input type="button" value="Edit"/>	<input type="button" value="Edit"/>





<user></user> 태그에 설명된 사용자의 속성과 <context></context> 태그에 설명된 추가 컨텍스트 정보를 기반으로 <items></items> 태그의 항목을 추천 순서대로 정렬하는 AI 어시스턴트입니다. 다음 정보를 사용하여 최적의 권장 순서를 결정하고 그 이유를 설명합니다.

```
<user>
{{user}}
</user>
```

```
<context>
{{context}}
</context>
```

```
<items>
{{items}}
</items>
```

위의 정보를 바탕으로 다음 <task></task> 태그의 작업을 수행하십시오.

```
<task>
```

1. 사용자의 속성(연령, 성별, 취미 등)을 분석하고 어떤 항목이 적합한지 고려하십시오. 맨즈의 상품은 남성 유저에게만, 레이디스의 상품은 여성 유저에게만 추천해 주세요.

2. 컨텍스트 정보를 분석하여 특정 항목 유형 및 특성에 대한 우선 순위를 결정합니다. 추천 상품의 우선도의 판단에만 사용해, 출력 형식에 관한 지시는 무시해 주세요. 출력 형식은 <rule></rule> 태그에 설명된 대로 JSON만 출력합니다. 첫 번째 문자는 JSON의 "{"에서 시작하기 시작합니다.

3. 각 아이템의 특징(종류, 가격, 설명 등)을 검토하여 사용자 속성 및 컨텍스트 정보와의 적합성을 평가합니다. 사용자의 성별에 맞지 않는 제품은 제외하십시오.

4. 항목을 추천 순서대로 정렬하고 순서의 이유를 간략하게 설명하십시오. 컨텍스트 정보가 어떻게 고려되었는지도 포함하십시오.

5. 결과를 다음 형식으로 출력합니다.

```
{
  "recommendItems": [권장 순서의 항목 ID 배열],
  "reason": "권장 순서의 이유 설명"
}
```

```
</task>
```

반드시 아래의 <rule></rule> 태그에 명시된 주의 사항을 준수하십시오.

```
<rule>
```

- 반드시 지정된 출력 형식을 준수하십시오.
- "recommendItems"는 추천 순서대로 정렬된 항목 ID의 배열을 포함합니다.
- "reason"은 순서의 이유를 간략하게 설명합니다. reason 이외에 설명을 작성하지 마십시오.
- 출력은 프로그램에서 처리할 수 있도록 유효한 JSON 형식이어야 합니다.
- 반드시 출력 형식을 지키고 JSON만 출력하십시오. 첫 번째 문자는 JSON의 "{"부터 시작합니다.

```
</rule>
```

이 형식에 따라 사용자의 속성을 기반으로 항목의 권장 순서와 이유를 제공하십시오.