

# RAG를 위한 Knowledge Base 설정

KB (Knowledge Base) for Amazon Bedrock 설정 방법 가이드 입니다. 이제 Bedrock 콘솔에서 RAG 작업을 쉽게 진행하세요!

19 July 2024

김제삼 (Jesam Kim)

Solutions Architect AWS

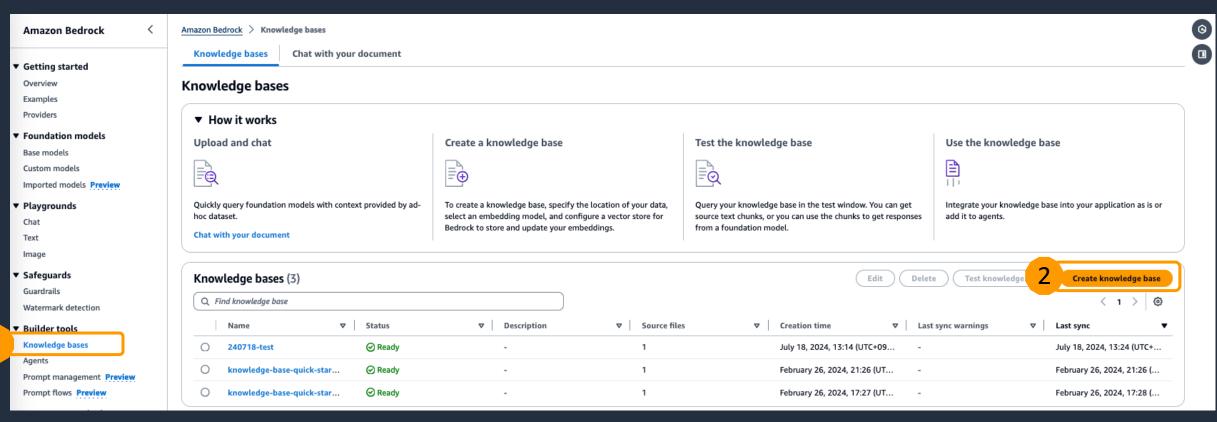
#### 1. KB 생성

Amazon Bedrock 콘솔에 접속 합니다.

- (1) Knowledge bases 메뉴에 접속 합니다.
- (2) Create knowledge base 를 클릭하여 새로운 KB를 생성 합니다.

\* KB를 만드는 것은 OpenSearch Serverless 컬렉션을 하나 만드는 것 입니다. 여기서 인텍스 구성 정보(청킹 등) 설정이 표함 됩니다.

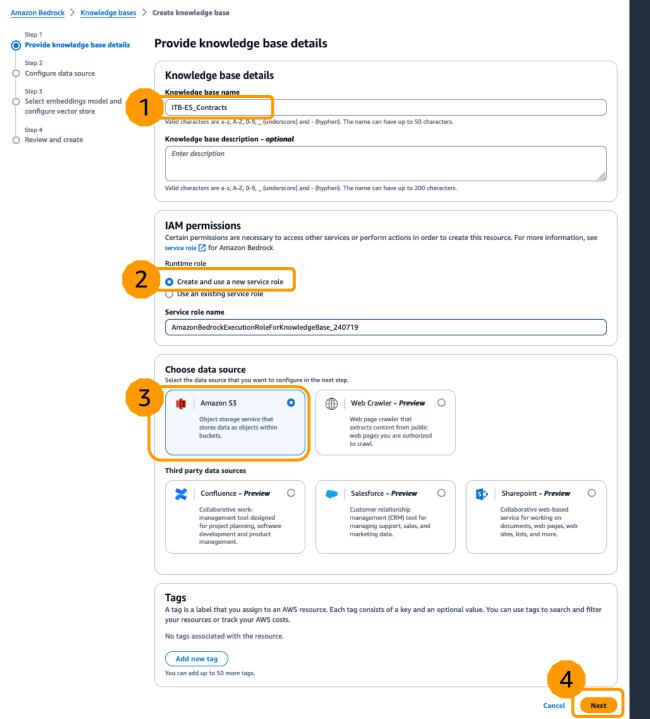
즉, KB*를 만드는 것은 하나의* Index*를 만드는 것과 같습니다*.





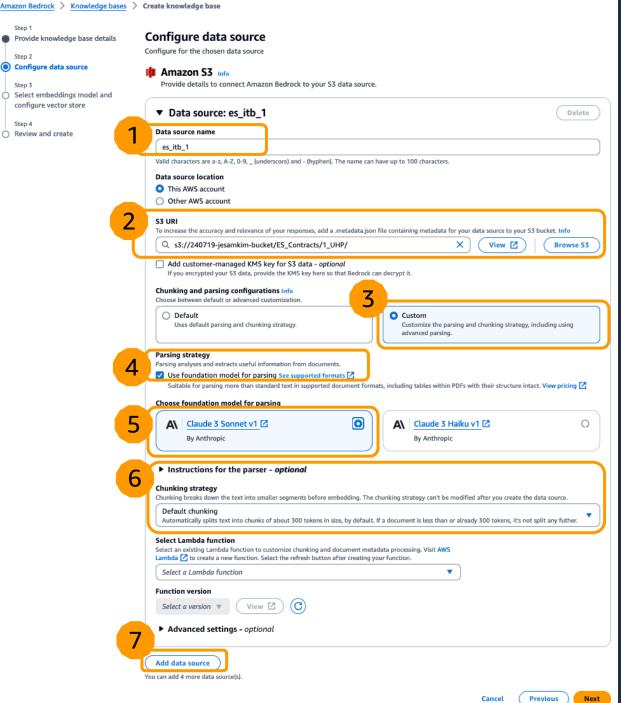
#### 2. KB 세부사항

- (1) Knowledge bases name 지정
- (2) IAM 롤 생성 (이미 만든 롤이 있다면 기존 롤 선택 가능)
- (3) Data source 는 S3 선택
- (4) Next 클릭



#### 3. Data source 구성 (1/2)

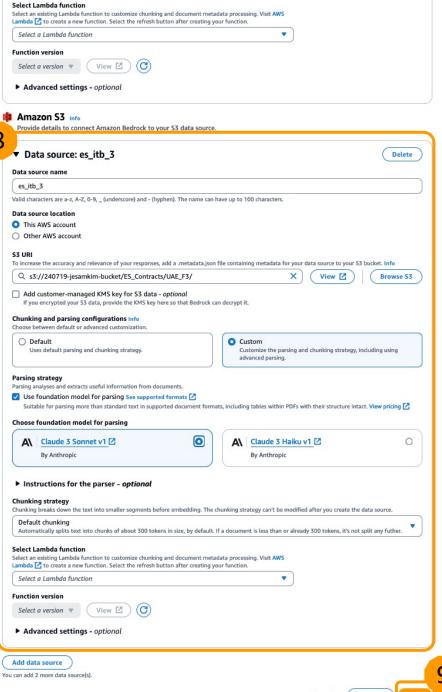
- (1) Data source name 지정
- (2) 문서가 업로드 된 S3 URI 선택 (Parsing Strategy를 사용하는 경우, S3 URI 당 처리 가능 문서 최대 100개)
- (3) 청킹 부분에서 Custom 선택
- (4) Parsing strategy 사용으로 체크
- (5) 문서 Parsing 부분에 사용할 모델 선택 → Claude 3 Sonnet
- (6) 선택사항 : Parsing strategy에 대한 프롬프트 수정 및 Chunking strategy는 바꿔볼 수 있음 (문서 마지막 Appendix 참조)
- (7) 선택사항: 추가 데이터 소스가 있다면 Add data source 선택 © 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved



## 3. Data source 구성 (2/2)

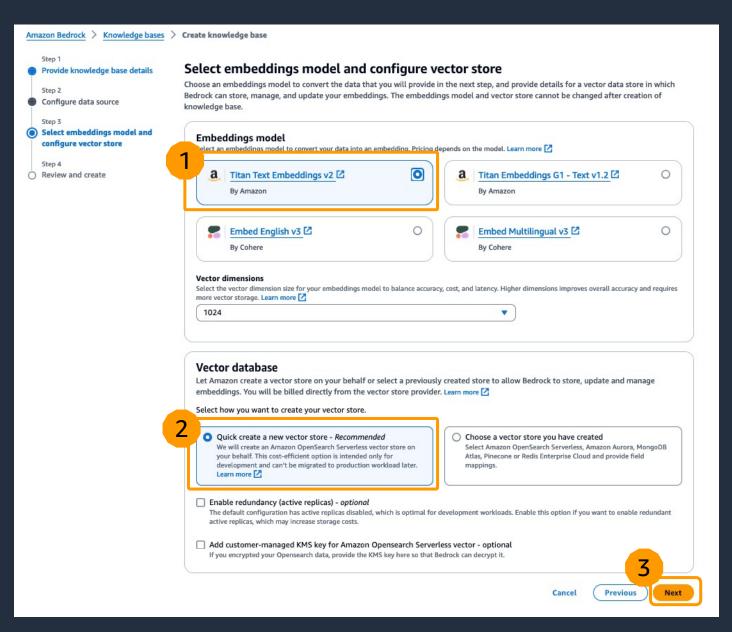
(8) **선택사항 :** 추가 Data source 가 있다면 앞의 과정처럼 S3 URI를 지정하고 Chunking 구성 등은 동일하게 설정 합니다.

(9) Next 버튼 클릭



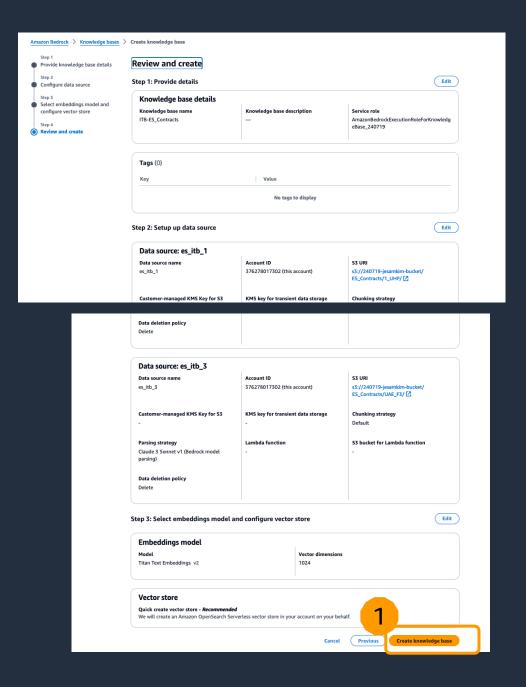
#### 4. 임베딩 모델 선택

- (1) 임베딩 모델 선택
  → 여기서는 Titan Embeddings v2 선택
- (2) Vector database 부분에서는 Quick create ~~ 를 선택 합니다.
- (3) Next 클릭



#### 5. 리뷰 & 생성

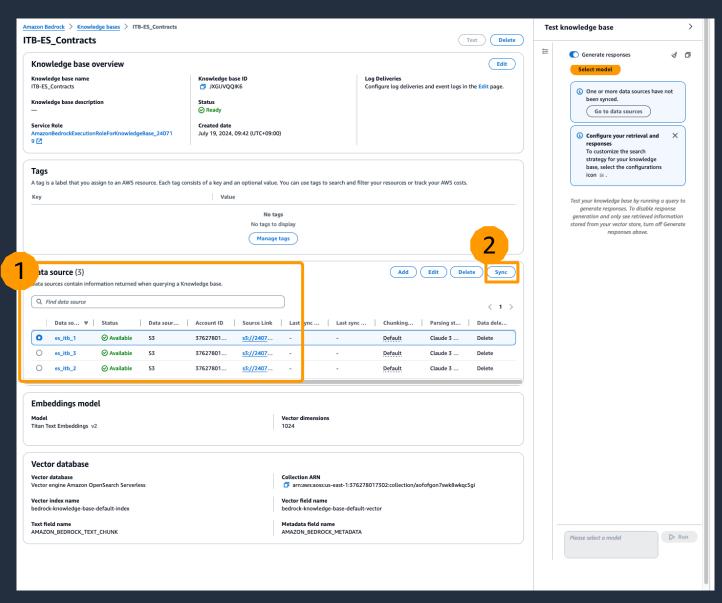
(1) 앞서 선택한 구성이 제대로 되었는지 확인하고 문제가 없으면 Create knowledge base 버튼을 클릭 합니다.





# 6. Data Source Sync (인덱싱) (1/2)

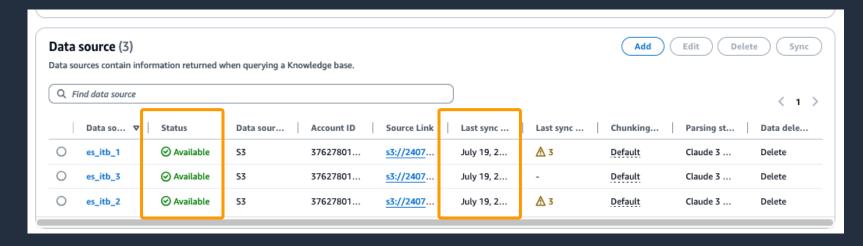
- (1) 앞서 추가한 data source 항목이 보입니다.
  (문서 갯수가 많아서 3개로 나누어서 등록했습니다;
  Parsing Strategy를 사용했기 때문에 Data source 1개 당 문서 최대 100개)
- (2) Data source를 선택하고 Sync 버튼을 누릅니다. (인덱싱 시작) 나머지 문서도 순차적으로 Sync 합니다.





# 6. Data Source Sync (인덱싱) (2/2)

Sync 가 완료되면 Last sync에 시간이 표시 됩니다. Status도 Available 인지 확인 입니다.

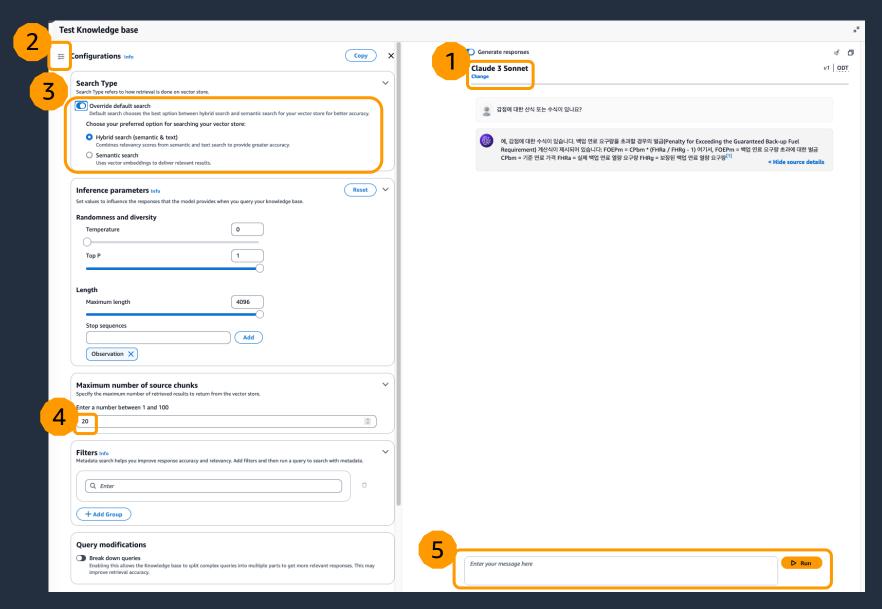




## 7. RAG 테스트 (1/2)

KB는 챗봇 UI가 제공되어, RAG 테스트를 콘솔에서 바로 해볼 수 있습니다.

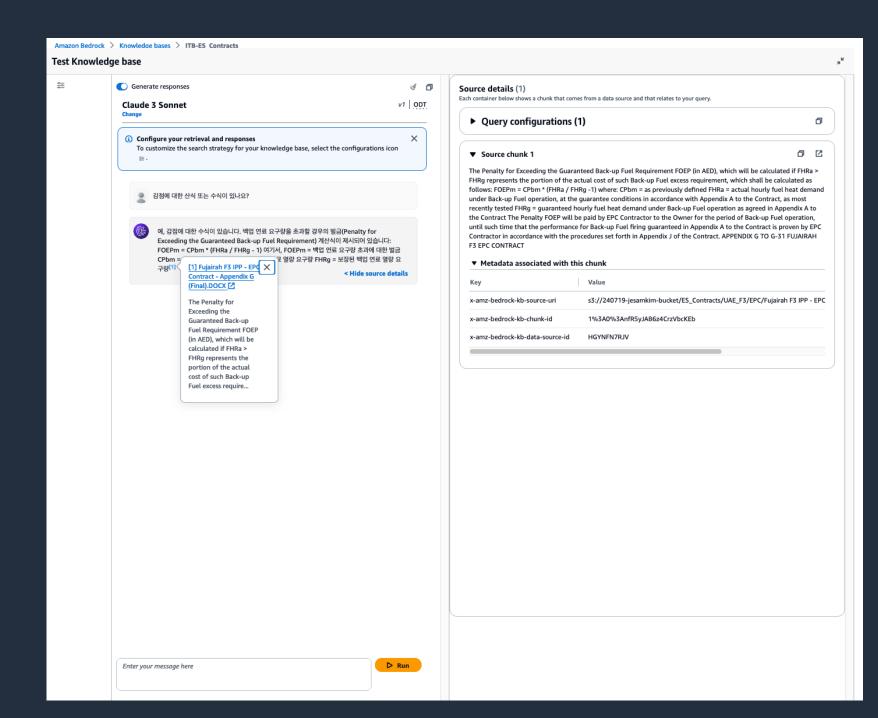
- (1) 챗봇에 사용할 모델 선택
- (2) Configurations 창 오픈
- (3) Search Type 은 하이브리드 방식으로 변경
- (4) 가져올 청크 갯수 조정
- (5) 사용자 쿼리 입력





# 7. RAG 테스트 (2/2)

KB는 챗봇 UI에서는 사용자 쿼리에 대한 응답을 생성할 때 사용한 Context를 함께 확인할 수 있습니다. (Citations)





# Thank you!

#### Instruction for the parser 프롬프트 (default)

Transcribe the text content from an image page and output in Markdown syntax (not code blocks). Follow these steps:

- 1. Examine the provided page carefully.
- 2. Identify all elements present in the page, including headers, body text, footnotes, tables, visulizations, captions, and page numbers, etc.
- 3. Use markdown syntax to format your output:
- Headings: # for main, ## for sections, ### for subsections, etc.
- Lists: \* or for bulleted, 1. 2. 3. for numbered
- Do not repeat yourself
- 4. If the element is a visulization
- Provide a detailed description in natural language
- Do not transcribe text in the visualization after providing the description
- 5. If the element is a table
- Create a markdown table, ensuring every row has the same number of columns
- Maintain cell alignment as closely as possible
- Do not split a table into multiple tables
- If a merged cell spans multiple rows or columns, place the text in the top-left cell and output ' ' for other
- Use | for column separators, |-|-| for header row separators
- If a cell has multiple items, list them in separate rows
- If the table contains sub-headers, separate the sub-headers from the headers in another row
- 6. If the element is a paragraph
- Transcribe each text element precisely as it appears
- 7. If the element is a header, footer, footnote, page number
- Transcribe each text element precisely as it appears

#### Output Example

A bar chart showing annual sales figures, with the y-axis labeled "Sales (\$Million)" and the x-axis labeled "Year". The chart has bars for 2018 (\$12M), 2019 (\$18M), 2020 (\$8M), and 2021 (\$22M).

Figure 3: This chart shows annual sales in millions. The year 2020 was significantly down due to the COVID-19 pandemic.

# Annual Report

## Financial Highlights

- \* Revenue: \$40M
- \* Profit: \$12M
- \* EPS: \$1.25

|| Year Ended December 31, || || 2021 | 2022 | |-l-l-| | Cash provided by (used in): ||| | Operating activities | \$ 46,327 | \$ 46,752 | | Investing activities | (58,154) | (37,601) | | Financing activities | 6,291 | 9,718 |

Here is the image.

#### 선택 가능한 Chunking strategy

# Chunking strategy Chunking breaks down the text into smaller segments before embedding. The chunking strategy can't be modified after you create the data source. Default chunking Automatically splits text into chunks of about 300 tokens in size, by default. If a document is less than or already 300 tokens, it's not split any futher. Default chunking Automatically splits text into chunks of about 300 tokens in size, by default. If a document is less than or already 300 tokens, it's not split any futher. Fixed-size chunking Splits text into your set approximate token size. Hierarchical chunking Organizes text chunks (nodes) into hierarchical structures of parent-child relationships. Each child node includes a reference to its parent node. Semantic chunking Organizes text chunks or groups of sentences by how semantically similar they are to each other. No chunking Suitable for documents that are already pre-processed or text split into separate files without any further chunking necessary.



#### **Chunking Strategy: Semantic Chunking**

- 자연어 처리 기술로 주어진 텍스트를 보다 의미 있고 완전한 청크 단위로 나누는 방식
- 임베딩 모델이 계산한 의미론적인 유사도 (semantic similiary) 기준으로 chunking
- 텍스트의 의미와 문맥을 활용하기 때문에 대부분의 활용 사례에서 보다 성능을 개선
- 지정 파라미터
  - Maximum Tokens: 하나의 chunk가 가질 수 있는 최대 token
  - Buffer Size for grouping surrounding sentences: Semantic 유사도를 평가할 때 함께 포함할 주변 문장의 수. Buffer가 1일 경우, 해당 문장과 앞뒤 문장을 1개씩 포함하여 총 3 문장을 임베딩. 맥락이 보다 풍부해짐
  - Breakpoint percentile threshold: 의미론적 유사도 기준을 설정 (90%일 경우 90% 미만이면 다른 chunk로 처리, 권장 95%)
- 작동 방식:
  - 1. 지정된 Buffer 크기로 먼저 chunking 진행
  - 2. 각 chunk에 대해 임베딩 생성
  - 3. 임베딩 차원에서 유사한 청크들이 buffer size와 similary Threshold 기준으로 합쳐짐



#### **Chunking Strategy: Hierarchical Chunking**

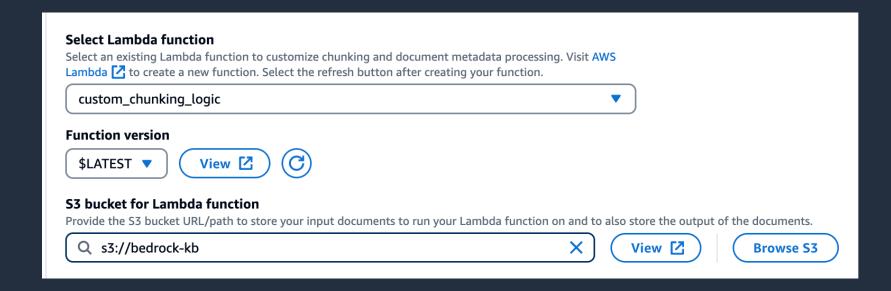
- 문서를 **부모-자식(parent-child)** chunk으로 구조화
- 보다 복잡한, 중첩된 구조의 데이터에서도 RAG를 효과적으로
- Semantic 검색은 child chunk로 이루어지지만 결과는 해당 child의 parent 로 모델에게 입력하여 전체 맥락을 모델에게 제시
- 법률 문서나 기술적인 매뉴얼, 논문 등에서 효과적
- 지정 파라미터
  - Parent: Parent의 최대 토큰 사이즈
  - Child: Child의 최대 토큰 사이즈
  - Overlap Tokens: Parent chunk는 child token 기준으로 중복되며 child는 지정. Child의 최대 token 크기의 20% 권장
- 작동 방식
  - 1. 문서는 먼저 부모와 자식의 chunking size 기준으로 chunking 됨
  - 2. 부모 자식 구조가 유지되며, 검색은 자식을 기준으로 이루어짐
  - 3. 자식 기준으로 검색이 되면 해당 자식이 속한 부모 단위로 모델에게 입력

aws

15

#### **Custom Processing using AWS Lambda**

- Knowledgebase에서 기본 제공하는 chunking 옵션이 아닌, 자체 로직을 활용할 수 있도록 함
- AWS Lambda 함수를 통해 사용되며, 함수가 문서를 chunking 한 후 다시 동일한 S3 버킷에 저장
- LangChain이나 LlamaIndex와 같은 프레임워크의 chunking 방식을 활용 가능
- Chunking뿐만 아니라 각 chunk에 메타데이터 등을 추가하는 데에도 활용할 수 있음





#### **Metadata Selection for CSV**

- CSV 형식의 파일을 RAG로 구성하기 위해 향상된 처리 기능
- **특정 열**들을 콘텐츠 필드 혹은 메타데이터 필드로 **구분**할 수 있도록 지원
- CSV로 처리 시 행을 기준으로 chunking됨
- Chunk(Row)에 대한 메타데이터 추가 가능
- Csv 파일과 함께 동일한 이름으로 <filename>.csv.metadata.json suffix 파일명으로 입력
- 기존 지원되었던 metadata 필터링과 함께 활용 가능

```
"metadataAttributes": {
  "docSpecificMetadata1": "docSpecificMetadataVal1",
  "docSpecificMetadata2": "docSpecificMetadataVal2"
"documentStructureConfiguration": {
  "type": "RECORD_BASED_STRUCTURE_METADATA",
  "recordBasedStructureMetadata": {
    "contentFields": [
         "fieldName": "column name"
    "metadataFieldsSpecification": {
      "fieldsToInclude": [
           "fieldName": "column_name"
      "fieldsToExclude": [
           "fieldName": "column_name"
```

#### **Metadata Customization**

- Chunk 단위의 메타데이터를 Lambda를 통해서 추가 가능
- 기본이나 fixed-size chunking만을 지원
- Knowledgebase가 먼저 chunking 된 파일을 S3에 입력하며, 지정된 Lambda가 chunk 단위 메타데이터를 추가
- 메타데이터가 추가된 파일을 동일 버킷에 저장하며, Knowledgebase가 이후 임베딩 등



#### **Query Reformulation**

- 복잡한 프롬프트(query)에 대해서 LLM을 활용해 여러 작은 sub-queries로 나누어 retrieve 작업 수행
- 작은 단위로 query 하여 단순화 시키고, 이에 대한 각각에 대한 chunk를 가져와서 복잡한 입력에 대해서도 높은 정확도를 가져감
- 이렇게 불러와진 chunk들은 유사도 기준으로 ranking이 되고, FM에게 입력

#### **Query modifications**

Break down queries

Enabling this allows the Knowledge base to split complex queries into multiple parts to get more relevant responses. This may improve retrieval accuracy.



"id": "msg\_bdrk\_01HMuPtzMmv7W6LC7VjUm3VZ",

를 어떻게 살펴보는가?</query>\n<query>\n<query>\n<query>\n<query>\n<query>\n<query>\n<query>\n<query>\n

"model": "claude-3-sonnet-20240229",

"type": "text",

"type": "message",
"role": "assistant",

"content": [

#### Query Reformulation 예시

사용자 프롬프트: 플랫폼 별로 인스타, 틱톡, 유투브 광고를 어떻게 살펴보고, 취미 여가 생활 정보는 보통 무슨 플랫폼을 사용해?

```
"operation": "InvokeModel".
                                                                               Claude 3.0 Sonnet 활용
   "modelId": "arn:aws:bedrock:us-east-1::foundation-model/anthropic.claude-3-sonnet-20240229-v1:0"
   "input": {
      "inputContentType": "application/json",
      "inputBodyJson": {
         "anthropic_version": "bedrock-2023-05-31",
         "messages": [
               "role": "user",
                                                                                                                           Query Generation Agent
               "content": [
                                                                                                                           기존 대화 기록을 고려하고,
                     "text": "플랫폼 별로 인스타, 틱톡, 유투브는 광고를 어떻게 살펴보고, 취미 여가 생활
                                                                                                                                                        여러 세부 쿼리로 나누어라
from the given conversation history.\nStep 2. Decide if the standalone question is a complex question. If so, decompose it into a set of relevant queries, which can be easier to answer.\n\nPlease pay
attention to the conversation history. If the user question is already a standalone question, you can skip Step 1.\nIf the standalone question does not require a further decomposition, you can skip Step
2.\n\nHere's an example for Step 1:\n\n<example>\nInput:\n<conversation_history>\n<conversation>How many vehicles can I include in a quote in Kansas</question>\n<answer>You can include 5 vehicles
in Texas?<standalone_question></qenerated_queries>\n\n\nHere are two examples for Step 2:\n\n<example>\nInput:\n\nUser: Were both of the following rock groups formed in California: Dig and Thinking
Fellers Union Local 282?\n\n0utput:\n<generated_queries><standalone_question>Were both of the following rock groups formed in California: Dig and Thinking Fellers Union Local 282?</standalone_question>
<query>What is the formation place of rock group Dig?</query><query>What is the formation place of the rock group Thinking Fellers Union Local 282?
</generated_queries>\n</example>\n\n<example>\nInput:\n\nUser: in 2018 , what percentage of undeveloped acres were located in the u.s?\n\nOutput:\n<qenerated_queries><standalone_question>in 2018 , what
percentage of undeveloped acres were located in the u.s?</standalone_question><query>the net undeveloped acres in 2018 of u.s.</query>query>the total net undeveloped acres in 2018</query>
</qenerated_queries>\n/n\nPlease output your response between <qenerated_queries>
         "max_tokens": 2048.
         "temperature": 0,
         "top_p": 1,
         "stop_sequences": [
            "\n0bservation'
                                                                                           플랫폼 별로 인스타, 틱톡, 유튜부에서는 광고를 어떻게 살펴보고, 취미 및 여가 생활 정보는 보통 어떤
         "top_k": 50
      "inputTokenCount": 681
                                                                                            플랫폼을 사용하는가?
                                                                                            인스타그램에서 광고를 어떻게 살펴보는가?
     "outputContentType": "application/json",
                                                                                            틱톡에서 광고를 어떻게 살펴보는가?
     "outputBodyJson": {
```

유투브에서 광고를 어떻게 살펴보는가?

"text": "<generated\_queries>\n<standalone\_question>플랫폼 별로 인스타, 틱톡, 유투브에서는 광고를 어떻게 살펴보고, 취미 및 여가 생활 정보는 보통 어떤 플랫폼을 사용하는가?</standalone\_question>\n<query>인스타그램에서 광고

취미 생활에서 주로 살펴보는 플랫폼은 무엇인가?