



aws INNOVATE

AI/ML AND DATA EDITION

FEBRUARY 22, 2024

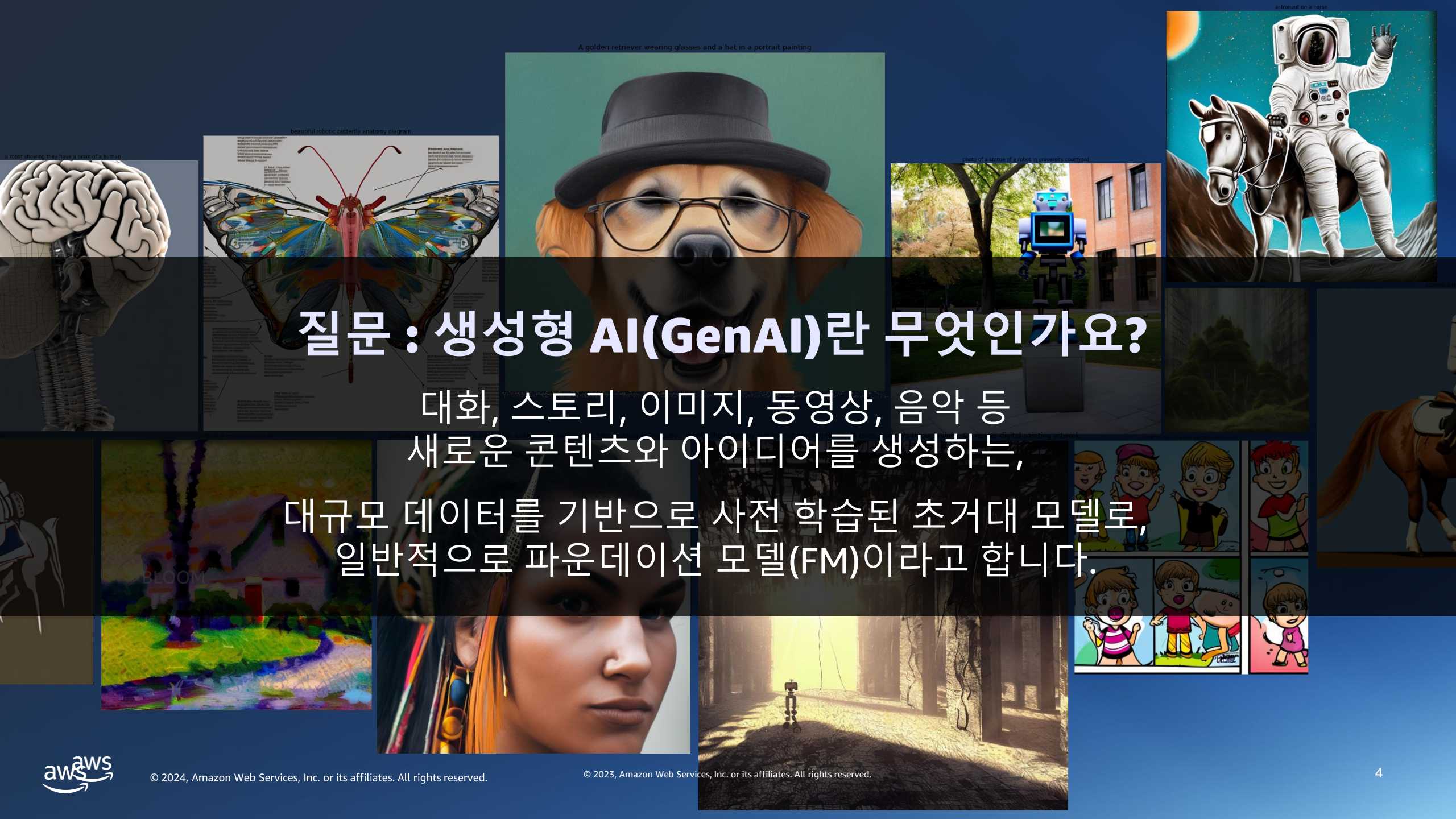
AWS에서 GenAI RAG를 구성하기 위한 다양한 Vector database에 대해 알아보기

전소영

Solutions Architect | Migration and Modernization, AWS

목차

1. 생성형 AI를 활용한 서비스의 도전과제
2. 검색 증강 생성 (RAG)와 기술 요소
3. AWS에서 제공하는 다양한 벡터 데이터베이스
4. AWS의 완전 관리형 End to End RAG 워크플로우
- Amazon Bedrock 지식 베이스 데모



A golden retriever wearing glasses and a hat in a portrait painting

beautiful robotic butterfly anatomy diagram

photo of a statue of a robot in university courtyard

astronaut on a horse

a robot showing how they a brain of a human

질문 : 생성형 AI(GenAI)란 무엇인가요?

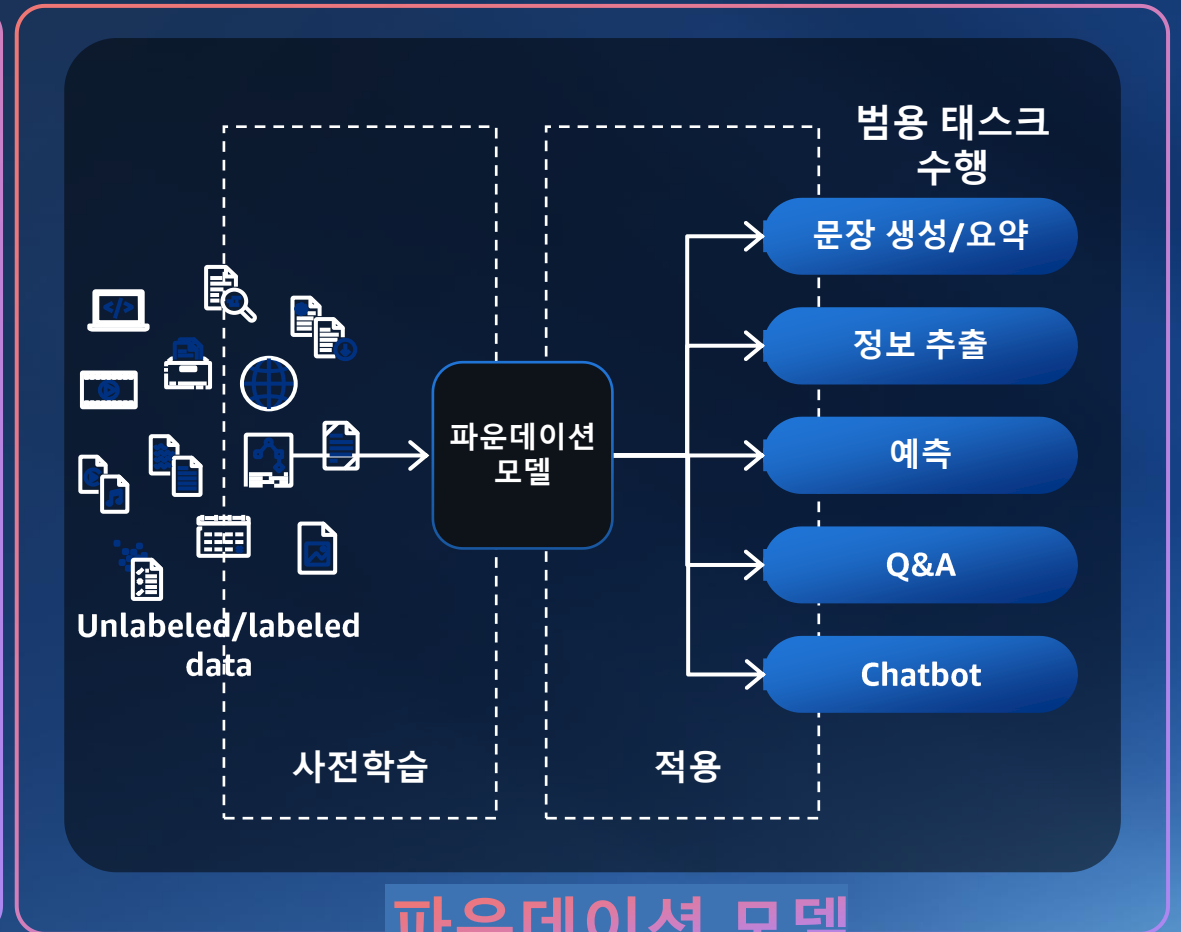
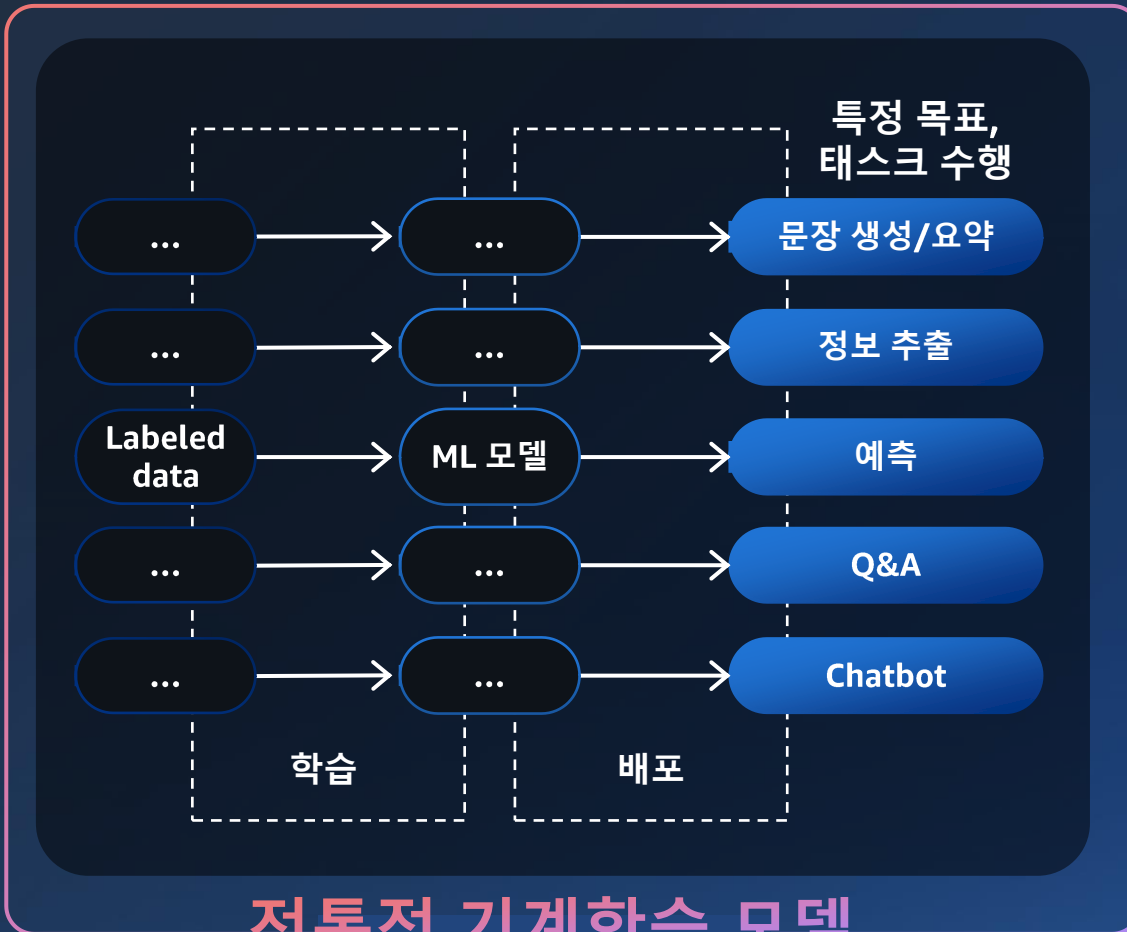
대화, 스토리, 이미지, 동영상, 음악 등
새로운 콘텐츠와 아이디어를 생성하는,

대규모 데이터를 기반으로 사전 학습된 초거대 모델로,
일반적으로 파운데이션 모델(FM)이라고 합니다.

전통적 기계 학습 모델과 파운데이션 모델 비교

정제 된 충분히 많은 데이터로 **특정 목적 포커스**된 학습/배포 과정을 목표 지능 수준까지 반복 수행

초거대 용량 데이터로 초거대 FM을 사전학습 시킨 후 **특정 목적 태스크 범용** 적용



생성형 AI 활용 서비스 개발 시 도전 과제



학습 데이터에 의존적



기한이 지난 정보



부정확한 사실을
제공하는 환각현상



맥락(컨텍스트)
이해 및 추론의 부족

파운데이션 모델의 도메인 적응 방법

1. 컨텍스트 기반 프롬프트 엔지니어링

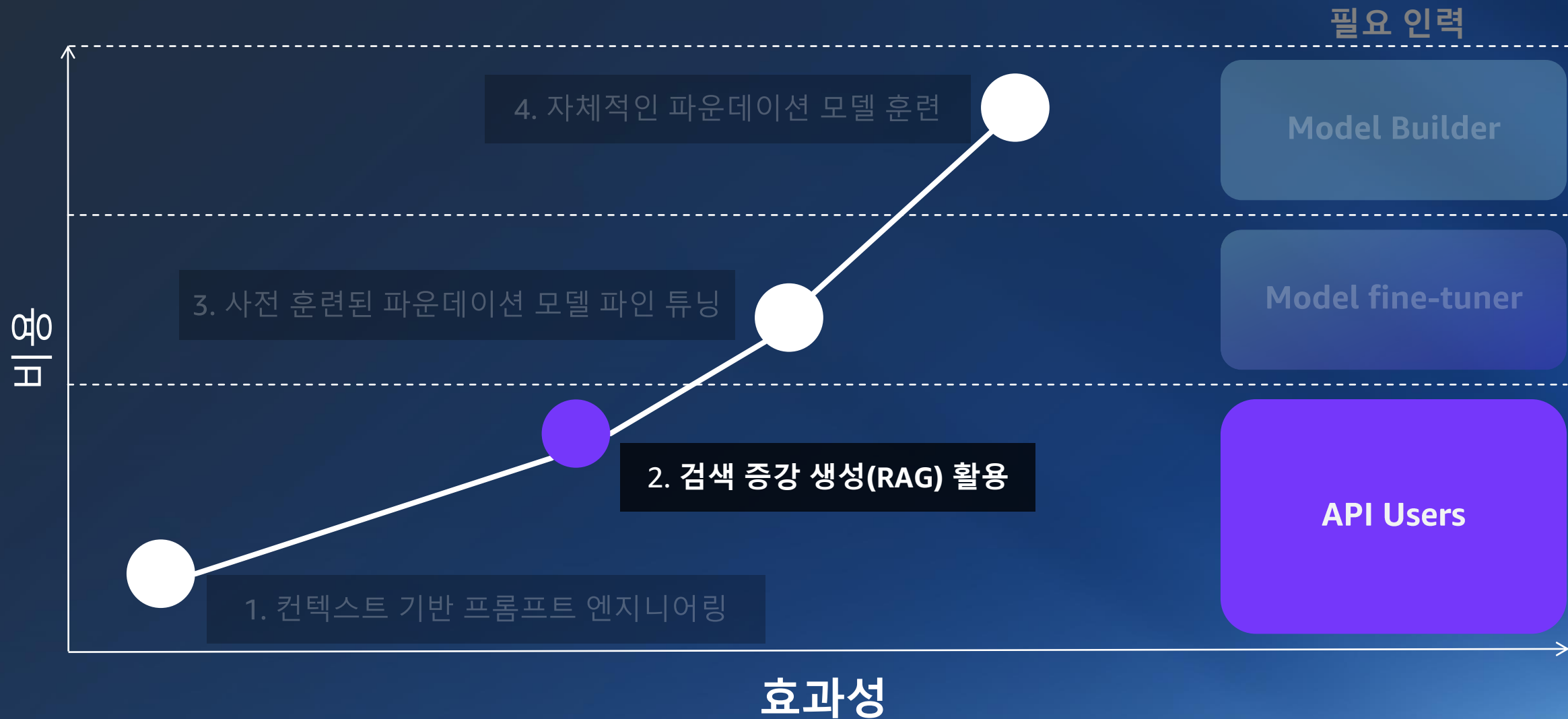
2. 검색 증강 생성(RAG) 활용

3. 사전 훈련된 파운데이션 모델 파인 튜닝

4. 자체적인 파운데이션 모델 훈련

사전 훈련된
파운데이션 모델을
그대로 활용

파운데이션 모델에 도메인 적응 비용과 효과성



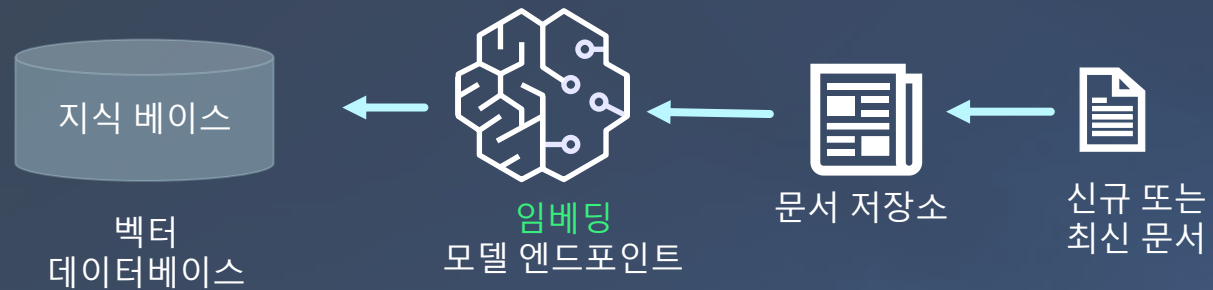
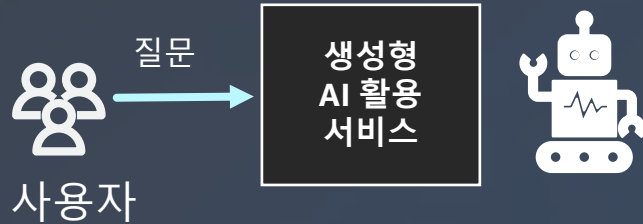
검색 증강 생성 (RAG)

1 최신/특정 도메인 지식 베이스 생성 (배치성 작업)

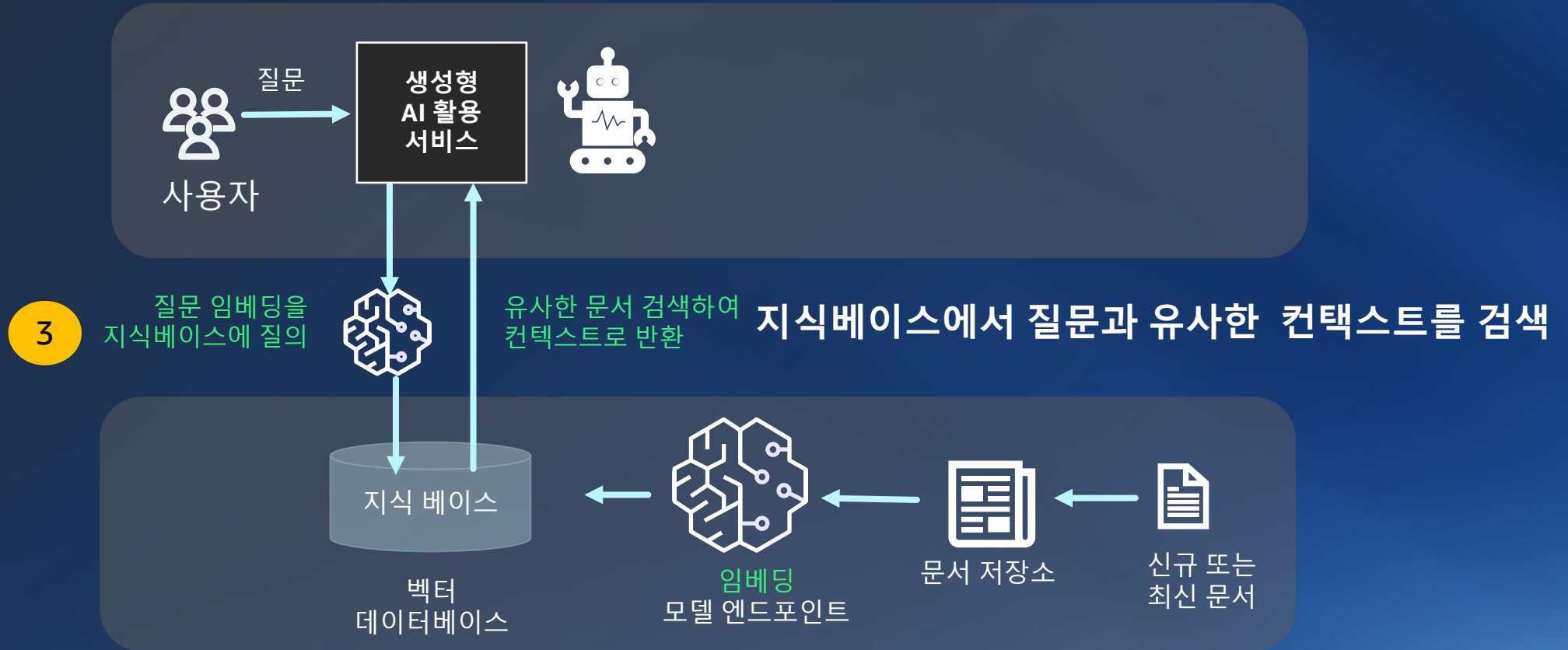


검색 증강 생성 (RAG)

2 사용자 질문 제출 (실시간)



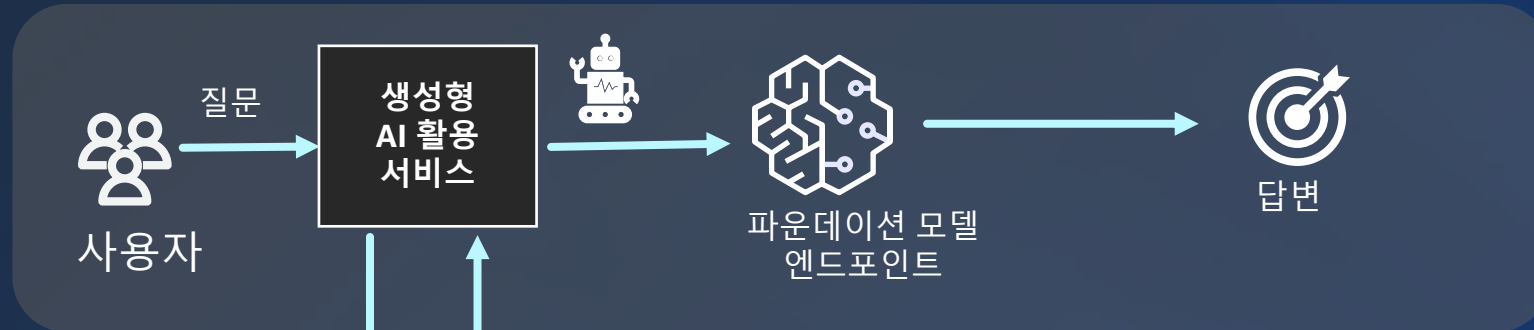
검색 증강 생성 (RAG)



검색 증강 생성 (RAG)

4

질문 + 반환된 컨텍스트 컨텍스트 기반 프롬프트로 확장

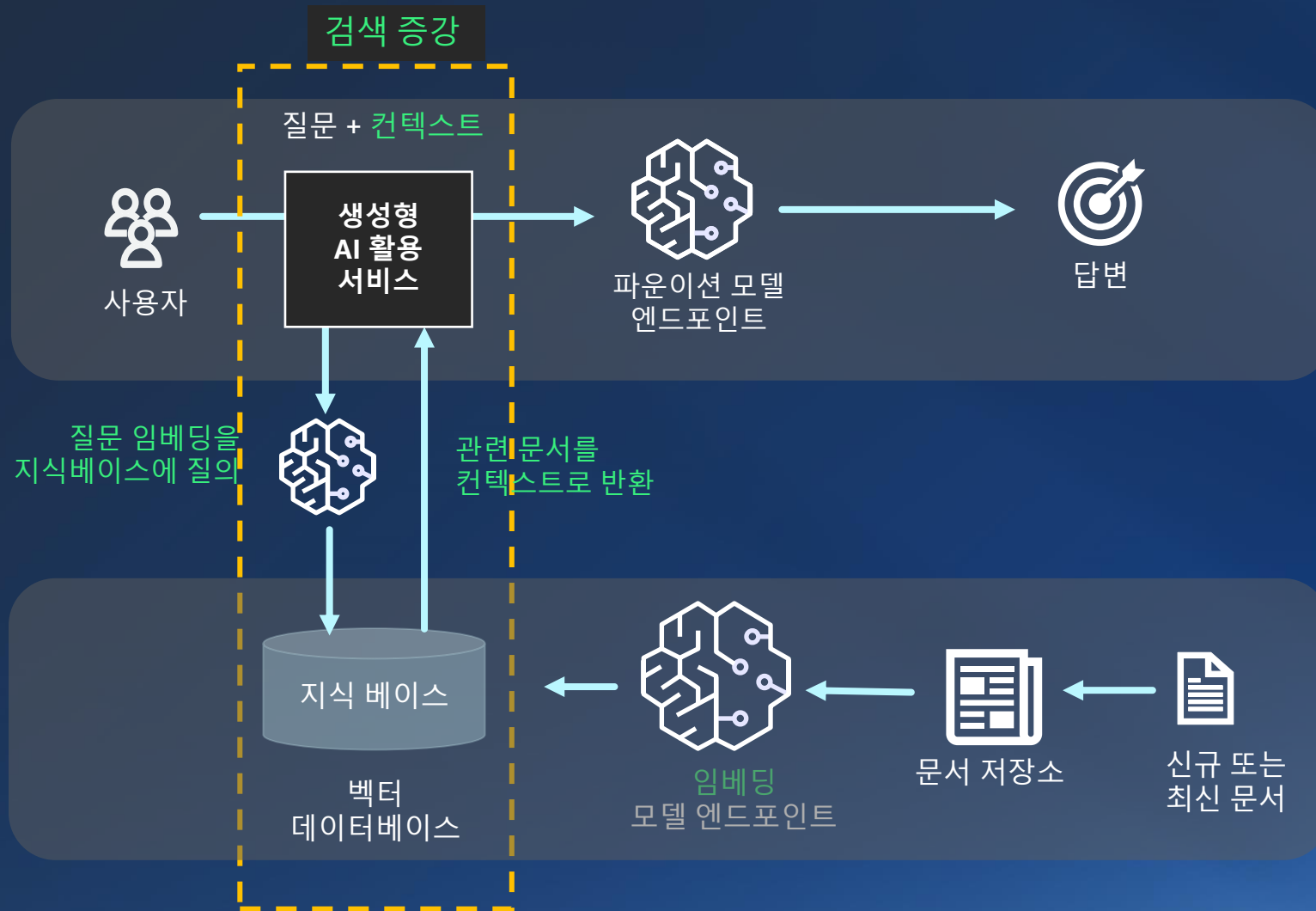


5

생성형 AI 활용 서비스는 신뢰성 높은 정보 생성



검색 증강 생성 (RAG)



RAG의 지식 베이스 구축 프로세스

원시 데이터

벡터 임베딩 생성

데이터 적재

데이터 활용



AWS에서 지원하는 벡터 데이터베이스

실시간 벡터 임베딩을 저장, 변경, 관리 및 고성능 벡터 유사성 검색 알고리즘을 제공하는 벡터 저장소

NEW!!

NEW!!

NEW!!

NEW!!



AMAZON
OPENSEARCH
SERVICE



AMAZON
AURORA
POSTGRESQL



AMAZON RDS
FOR
POSTGRESQL



AMAZON
MEMORYDB
FOR REDIS



AMAZON
DOCUMENTDB



AMAZON
NEPTUNE



AMAZON
DYNAMODB

VIA ZERO-
ETL

벡터 Vector

Magnitude (크기)

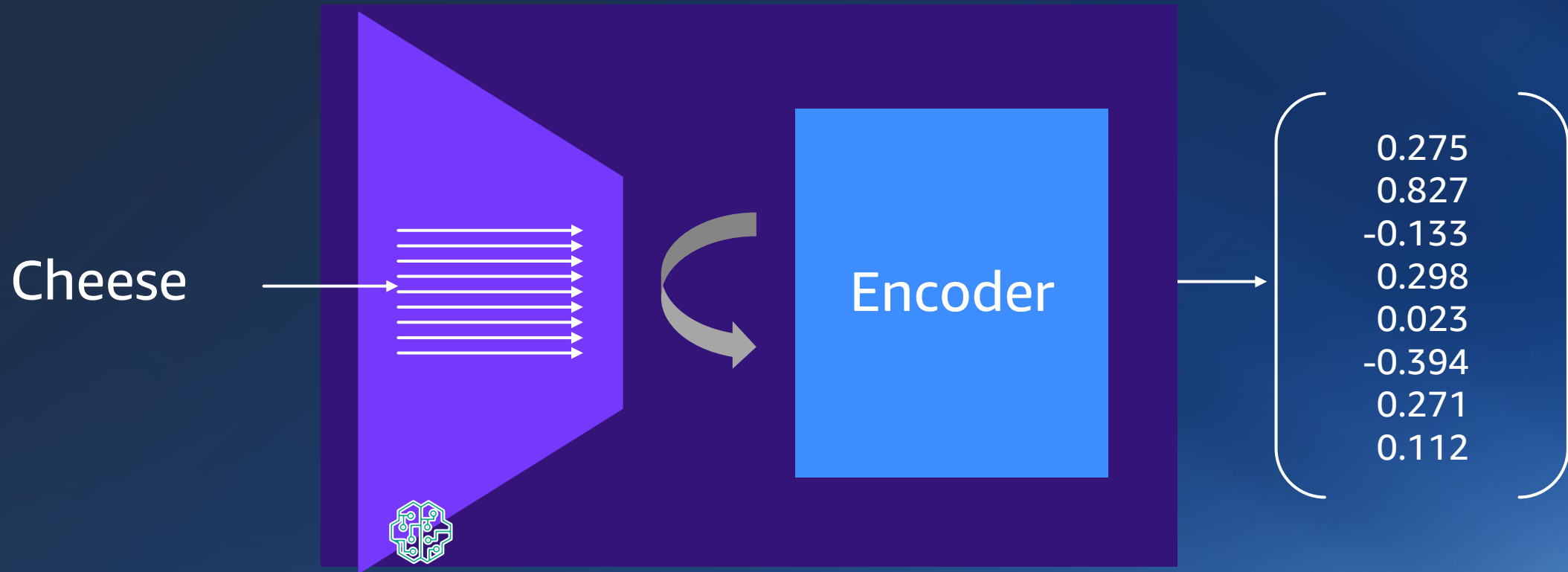


N 차원 의 원소들의 배열, 1xN 행렬, 행 벡터

X	Y	Z	N	
0.275	0.827	-0.133	0.298	0.023	-0.394

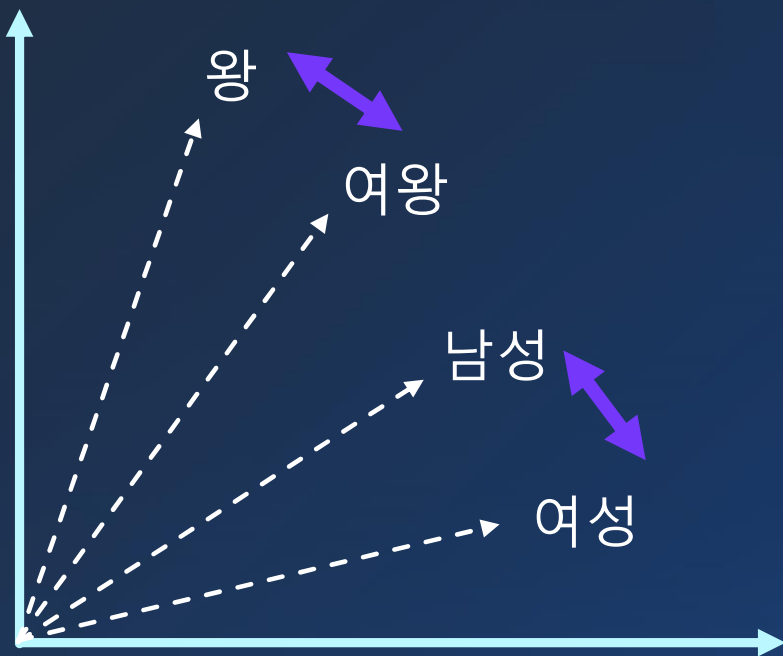
임베딩 Embedding

임베딩은 단어, 문장, 문서, 이미지 등의 데이터를 모델의 인코더를 활용하여 컴퓨터가 이해할 수 있는 형태의 벡터로 변환하는 과정 또는 변환된 벡터



벡터 공간 Vector space

왕 - 남성 = 여왕

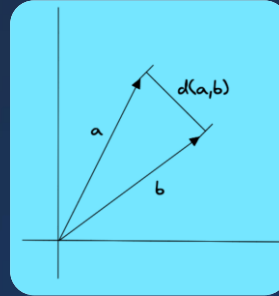


클러스터링



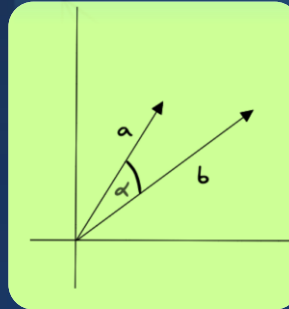
벡터 간의 유사성 측정

유클리드
(Euclidean, L2)



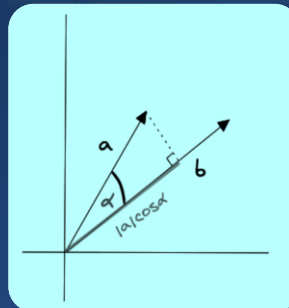
$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$$

코사인 유사성
(Cosine Similarity)



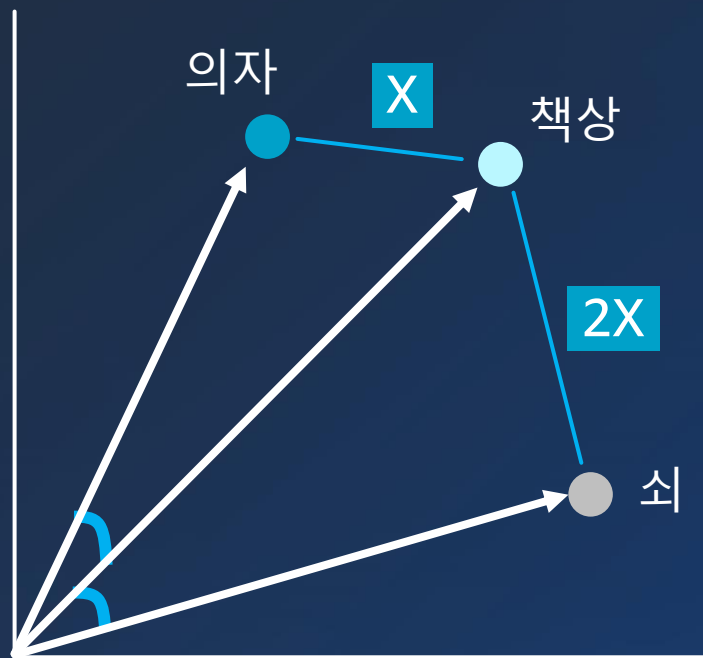
$$\text{sim}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|}$$

내적
(Inner Product)



$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \alpha$$

벡터 검색 알고리즘



k-NN k-최근접 이웃
k-Nearest Neighbors

A-NN 근사 최근접 이웃
Approximate Nearest Neighbors

재현율(Recall)

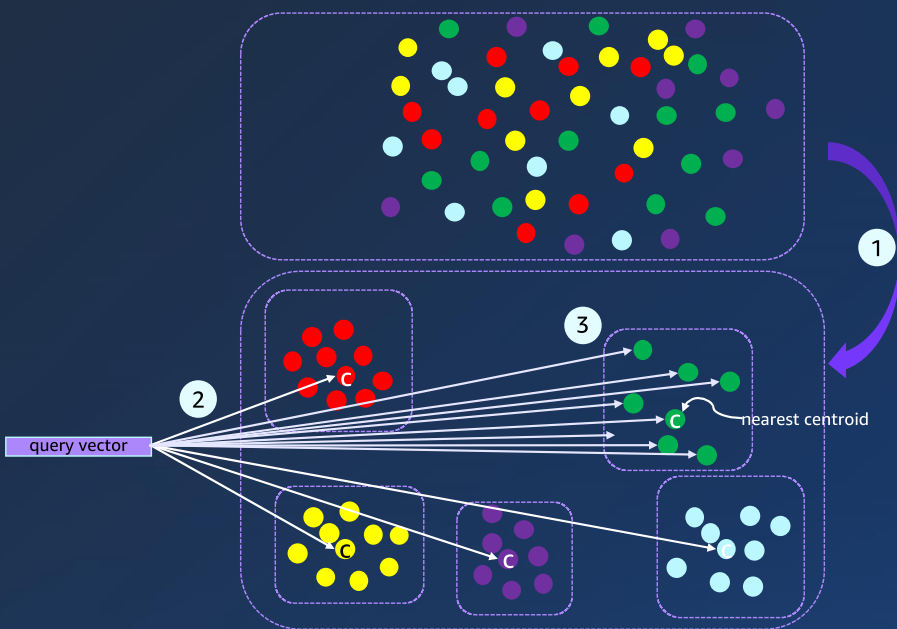


Ex) 재현율 80%

대규모 벡터 검색을 위한 ANN 기반 인덱싱 알고리즘

IVFFlat

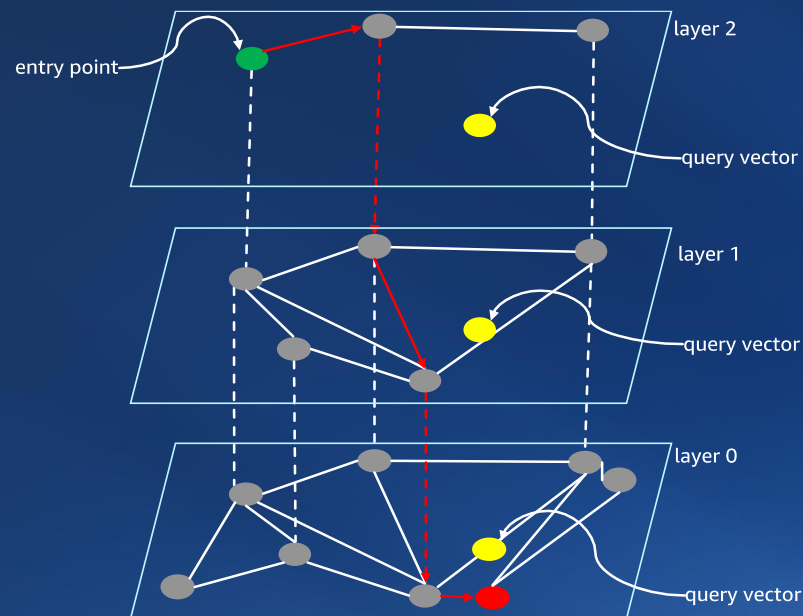
Inverted File with Flat Compression



- K-means 기반 버킷 구성
- 빠른 인덱싱
- List, Probes 매개변수

HNSW

Hierarchical navigable small world



- 그래프 기반, 이웃 벡터 레이어구성
- 대규모 데이터 세트의 고성능, 높은 재현율
- M, ef_construction, ef_search 매개변수



Amazon OpenSearch 벡터 엔진



사용의 용이성

서버리스를 지원하여 간단한 API로 시작가능.
용량을 계획, 규모를 관리할 필요성 제거



고성능 및 대규모 벡터 유사성 검색

수천 차원의 수십억 벡터를 밀리초 단위로 저장,
업데이트, 검색 가능



벡터 검색과 키워드 검색 하이브리드

DSL을 통한 키워드 검색 및 랭킹 기능을 사용하여
키워드 검색과 벡터 검색을 결합한 검색 가능



비용 효율성과 안정성 최적화

오픈서치의 스토리지 수직 및 노드의 수평 확장성
활용하여 프로덕션의 비용 효율성과 안정성 확보



엔터프라이즈급 보안

암호화 및 VPC 엔드포인트를 통해 데이터 저장
및 이동 시 데이터 보안, 인증 및 액세스 제어

OpenSearch의 벡터 데이터베이스

- K-NN 플러그인 벡터 엔진 제공

PUT my-index

```
{
  "settings": { "index.knn": true },
  "mappings":
  {
    "properties": {
      "my_vector1": { "type": "knn_vector", "dimension": 2 },
      "my_vector2": { "type": "knn_vector", "dimension": 4 }
    }
  }
}
```

POST _bulk

```
{ "index": { "_index": "my-index", "_id": "1" } }
{ "my_vector1": [1.5, 2.5], "price": 12.2 }
{ "index": { "_index": "my-index", "_id": "2" } }
{ "my_vector1": [2.5, 3.5], "price": 7.1 }
{ "index": { "_index": "my-index", "_id": "3" } }
{ "my_vector1": [3.5, 4.5], "price": 12.9 }
{ "index": { "_index": "my-index", "_id": "4" } }
{ "my_vector1": [5.5, 6.5], "price": 1.2 }
{ "index": { "_index": "my-index", "_id": "5" } }
{ "my_vector1": [4.5, 5.5], "price": 3.7 }
{ "index": { "_index": "my-index", "_id": "6" } }
{ "my_vector2": [1.5, 5.5, 4.5, 6.4], "price": 10.3 }
{ "index": { "_index": "my-index", "_id": "7" } }
{ "my_vector2": [2.5, 3.5, 5.6, 6.7], "price": 5.5 }
{ "index": { "_index": "my-index", "_id": "8" } }
{ "my_vector2": [4.5, 5.5, 6.7, 3.7], "price": 4.4 }
{ "index": { "_index": "my-index", "_id": "9" } }
{ "my_vector2": [1.5, 5.5, 4.5, 6.4], "price": 8.9 }
```

https://docs.aws.amazon.com/ko_kr/opensearch-service/latest/developerguide/knn.html
<https://opensearch.org/docs/latest/search-plugins/knn/knn-index/>



© 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Vector DB로써의 OpenSearch

- 시맨틱 벡터 검색

```
GET my-index/_search
{
  "size": 2,
  "query": {
    "knn": {
      "my_vector2": {
        "vector": [2, 3, 5, 6],
        "k": 2
      }
    }
  }
}
```

혼합 검색 (Hybrid search)

```
GET my-index/_search
{
  "size": 2,
  "query": {
    "knn": {
      "my_vector2": {
        "vector": [2, 3, 5, 6],
        "k": 2
      }
    }
  },
  "post_filter": {
    "range": {
      "price": {
        "gte": 6,
        "lte": 10
      }
    }
  }
}
```

OpenSearch 지원 벡터 엔진과 인덱싱

	NMSLIB-HNSW	FAISS-HNSW	Lucene-HNSW
Max dimension	16,000	16,000	1024
Filter	Post-filter	Pre-filter and Post-filter	Filter while search
Distance formula	l2, innerproduct, cosinesimil, l1, linf	l2, innerproduct	l2, cosinesimil
Indexing latency	Low	Low	Low
Query latency & quality	Low latency & high quality	Low latency & high quality	High latency & high quality
Vector compression	Flat	Flat	Flat
Memory consumption	High	High	High

OpenSearch 서버리스 벡터 엔진 정식 출시

컬렉션 설정 구성 정보

컬렉션은 워크로드를 지원하기 위해 함께 작동하는 인덱스의 논리적 그룹입니다.

수집 세부 정보

수집 이름

vector-stores

설명 - 선택 사항

데이터셋의 벡터를 저장하기 위한 벡터 저장소

수집 유형

사용 사례 선택

☐ 시계열

머신에서 생성된 대량의 반정형 데이터를 실시간으로 분석하는 데 사용합니다.

☐ 검색

네트워크 내에서 애플리케이션을 구동하는 전체 텍스트 검색에 사용합니다.

☒ 벡터 검색

벡터 엔진에서 벡터 임베딩을 저장하고 검색하는 데 사용합니다. [자세히 알아보기](#)

배포 유형

사용 사례에 맞는 배포 설정 지정

☐ 개발 및 테스트 모드 활성화

기본적으로 프로덕션 워크로드에는 복제본을 사용합니다. 개발 또는 테스트에 사용하려면 확인란을 선택하세요.

벡터 필드 추가

벡터 필드 이름

my_vector2

이름은 3~64자여야 합니다. 유효한 문자는 a~z(소문자 및 대문자), 0~9(숫자), -(하이픈), _(밑줄)입니다.

Engine

nmslib

치수

4700

값은 1~16,000 사이의 숫자여야 합니다.

거리 지표

유클리드

▼ 추가 설정 - 선택 사항

M

플러그인이 각 새 요소에 대해 생성하는 양방향 링크의 수입니다. 이 값을 늘리거나 줄이면 메모리 소비에 큰 영향을 미칠 수 있습니다.

16

값은 2~100 사이의 숫자여야 합니다

ef_construction

k-NN 그래프 생성 중에 사용되는 동적 목록의 크기. 값이 클수록 그래프는 더 정확하지만 인덱싱 속도는 느려집니다.

512

ef_search

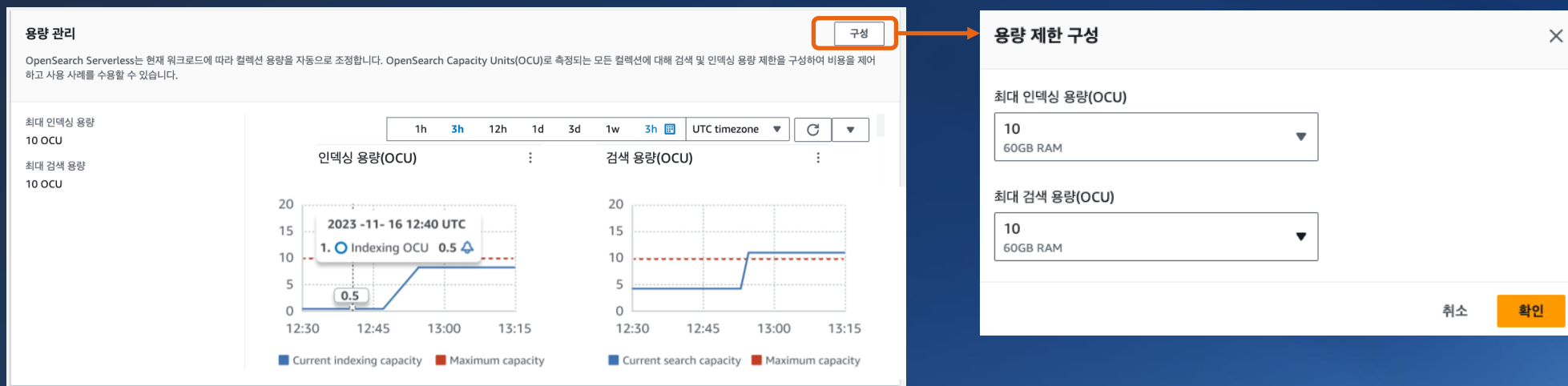
k-NN 검색 중에 사용되는 동적 목록의 크기. 값이 클수록 검색 정확도는 높아지지만 검색 속도는 느려집니다.

512



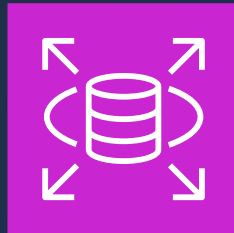
OpenSearch 서버리스 자동 용량 관리

- 1 OCU는 1 vCPU, 6 GB RAM, 120GB of Amazon EBS 볼륨 지원
- OCU 단위로 유사성 검색 쿼리 수집 및 실행
- OCU 단위로 128개 차원의 경우 2백 만개 벡터 처리
- 용량 자동 조정 기능을 제공하여 재색인 없이 10억 벡터 규모로 확장 가능





Amazon Aurora



Amazon RDS



PostgreSQL

pgVector 확장

Aurora PostgreSQL 15.3, 14.8, 13.11, 12.15 and higher
Amazon RDS running PostgreSQL 15.2 and higher



벡터 저장소

오픈 소스 pgvector 확장(extension) 사용 가능



벡터 인덱싱

IVFFlat, HNSW 인덱스 지원



유사성 검색 및 벡터 거리 측정

K-NN, ANN 알고리즘 사용 가능
유클리드 <-> , 코사인 유사성(<=>), 내적(<#>) 지원



메타데이터 저장 및 조인

데이터, 임베딩, 메타데이터를 같은 테이블에 저장 가능
또는 임베딩과 다른 테이블간의 조인 가능

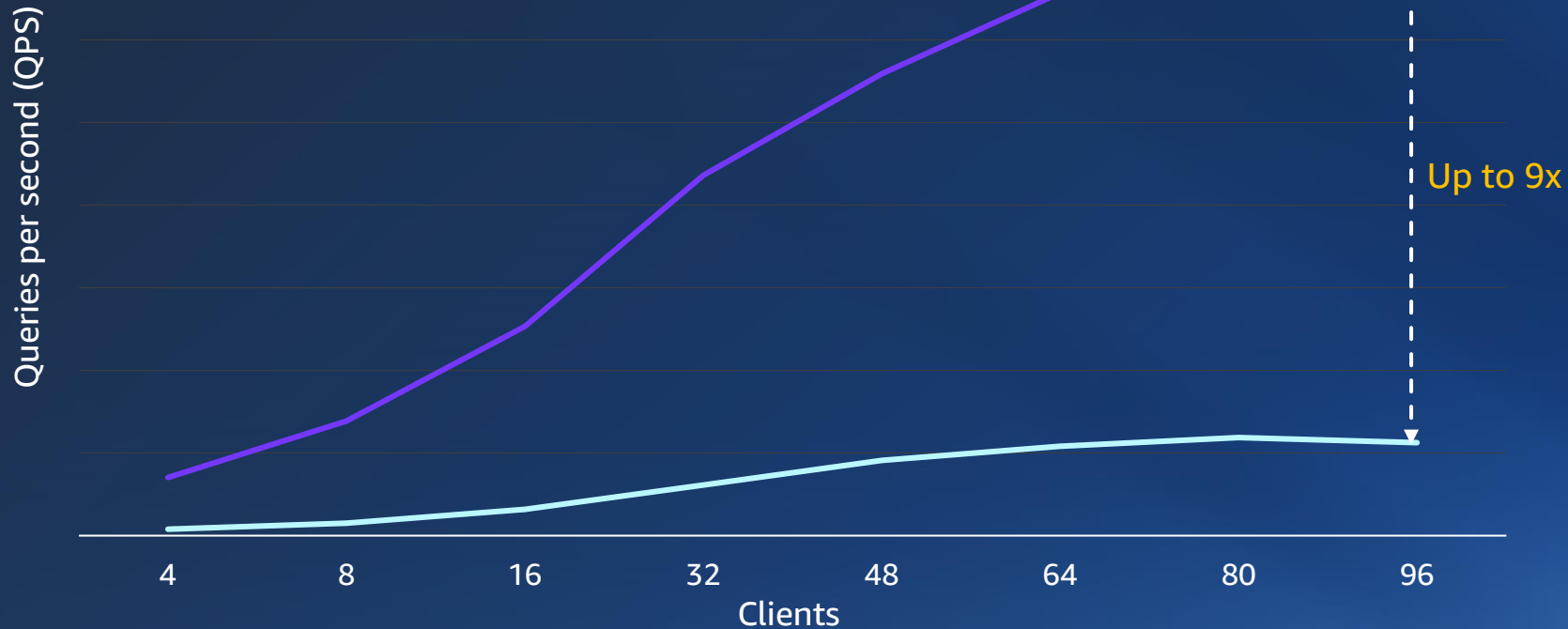


뛰어난 성능·확장성·가용성·비용 효율성

고사양 상용 데이터베이스의 성능, 확장성, 가용성에
오픈소스 데이터베이스의 비용 효율성의 이점 활용 가능

Aurora I/O 최적화에서의 벡터 검색 성능 향상

BigANN 벤치마크 및 재현율 0.9578, 벡터 10억 데이터



— QPS - I/O Optimized - db.r6g.12xlarge

— QPS - I/O Optimized w/Optimized Reads - db.r6gd.12xlarge

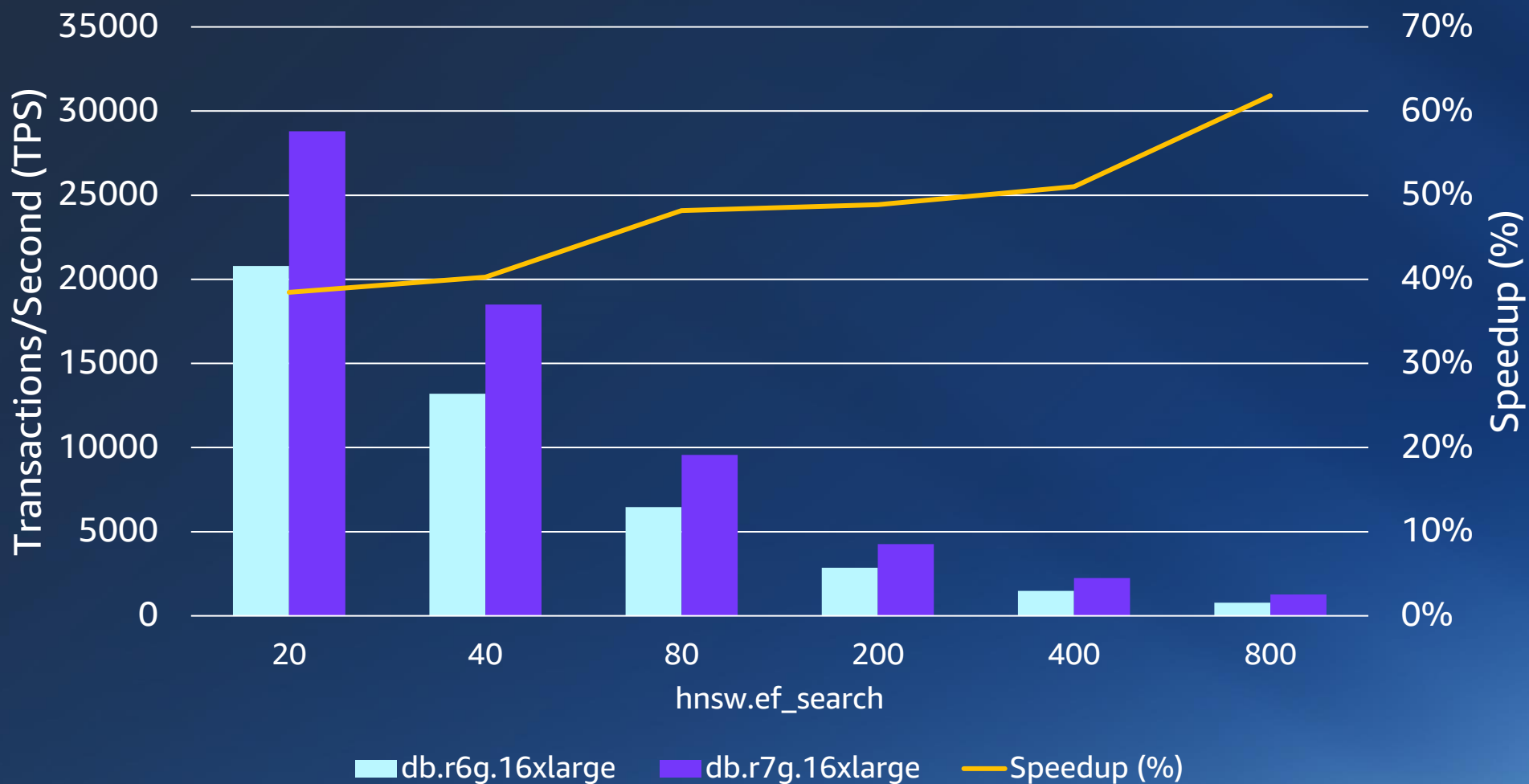
Database size: 1.28TB (Data: 560 GB, Index: 720GB)

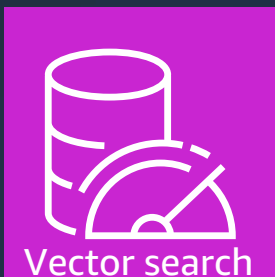
pgvector v0.5: HNSW index



AWS Graviton3 타입에서의 벡터 검색 성능 향상

1536 차원 벡터 HNSW 검색





Amazon MemoryDB for Redis 벡터 검색 (Preview)

Redis v7.1용 MemoryDB와 함께 사용 가능

미국 동부(버지니아 북부), 미국 동부(오하이오), 미국 서부(오레곤),
아시아 태평양(도쿄) 및 유럽(아일랜드) 리전에서 지원



© 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

높은 처리량과 높은 재현율이 필요한 애플리케이션을 위한 AWS에서 가장 빠른 벡터 검색



밀리초 단위의 검색
및 업데이트 소요 시간



최대 419GB 메모리 및
32,768개 차원 지원



수백만 개의 임베딩 저장
Redis JSON 및 해시 key



FLAT 및 HNSW 인덱싱



유클리드, 코사인
및 내적 유사성 검색 지원

MemoryDB 벡터 검색 사용 설정

클러스터 설정 정보

클러스터 생성 방법 선택

☐ 간편 생성

권장 모범 사례 구성을 사용하세요. 클러스터를 생성한 후 옵션을 수정할 수도 있습니다.

☒ 새 클러스터 생성

새 클러스터에 대한 모든 구성 옵션을 설정합니다.

☐ 스냅샷에서 복원

기존 RDB 파일을 사용하여 클러스터를 복원합니다.

클러스터 정보

이름

프라이마리 노드와 읽기 전용 복제본 노드를 포함하는 클러스터의 이름입니다.

이름

이름은 필수이고 최대 40자를 사용할 수 있으며, 문자로 시작해야 합니다. 하이픈으로 끝나거나 2개의 연속적인 하이픈을 포함해서는 안 됩니다. 유효한 문자: A~Z, a~z, 0~9 및 -(하이픈)

설명 - 옵션

이 클러스터에 대한 설명입니다.

설명

서브넷 그룹

서브넷 그룹

☒ 기존 서브넷 그룹 선택

☐ 새 서브넷 그룹 생성

서브넷 그룹

Amazon VPC에서 실행 중인 클러스터에 대해 지정할 수 있는 서브넷 모음입니다.

서브넷 그룹 선택

클러스터 설정

☒ 벡터 검색 활성화 - 공개 평가판 정보

벡터 임베딩을 저장하고 벡터 유사성 검색을 수행할 수 있습니다.

이 벡터 검색을 위한 평가판은 MemoryDB for Redis 버전 7.1 및 단일 샤드 구성과 호환됩니다. 벡터 검색 및 이러한 구성은 생성 후에는 수정할 수 없습니다. 프로덕션 클러스터에서는 이 기능을 활성화하지 않는 것이 좋습니다.

Redis 버전 호환성

노드에서 실행되는 Redis 엔진의 버전 호환성입니다.

7.1

포트

노드가 연결을 허용하는 포트 번호입니다.

6379

파라미터 그룹

파라미터 그룹은 노드와 클러스터의 런타임 속성을 제어합니다.

default.memorydb-redis7.search.preview

노드 유형

배포할 노드의 유형과 연결된 메모리 크기입니다.

db.r7g.large

13.07 GiB memory Up to 12.5 Gigabit network performance

샤드 수

샤드 수를 1~500 범위의 값으로 입력합니다.

1

샤드당 복제본

각 샤드의 복제본 수를 0~5 범위의 값으로 입력합니다.

1



MemoryDB 벡터 인덱스 생성

```
FT.CREATE my_index ON HASH // 또는 JSON
    SCHEMA field_name VECTOR HNSW 6 // 또는 FLAT
    TYPE FLOAT32 DIM 128 DISTANCE_METRIC L2 // 또는 IP , COSINE
```

```
FT.CREATE my_index ON HASH
    SCHEMA field_name VECTOR HNSW 6
    TYPE FLOAT32 DIM 128 DISTANCE_METRIC COSINE
```

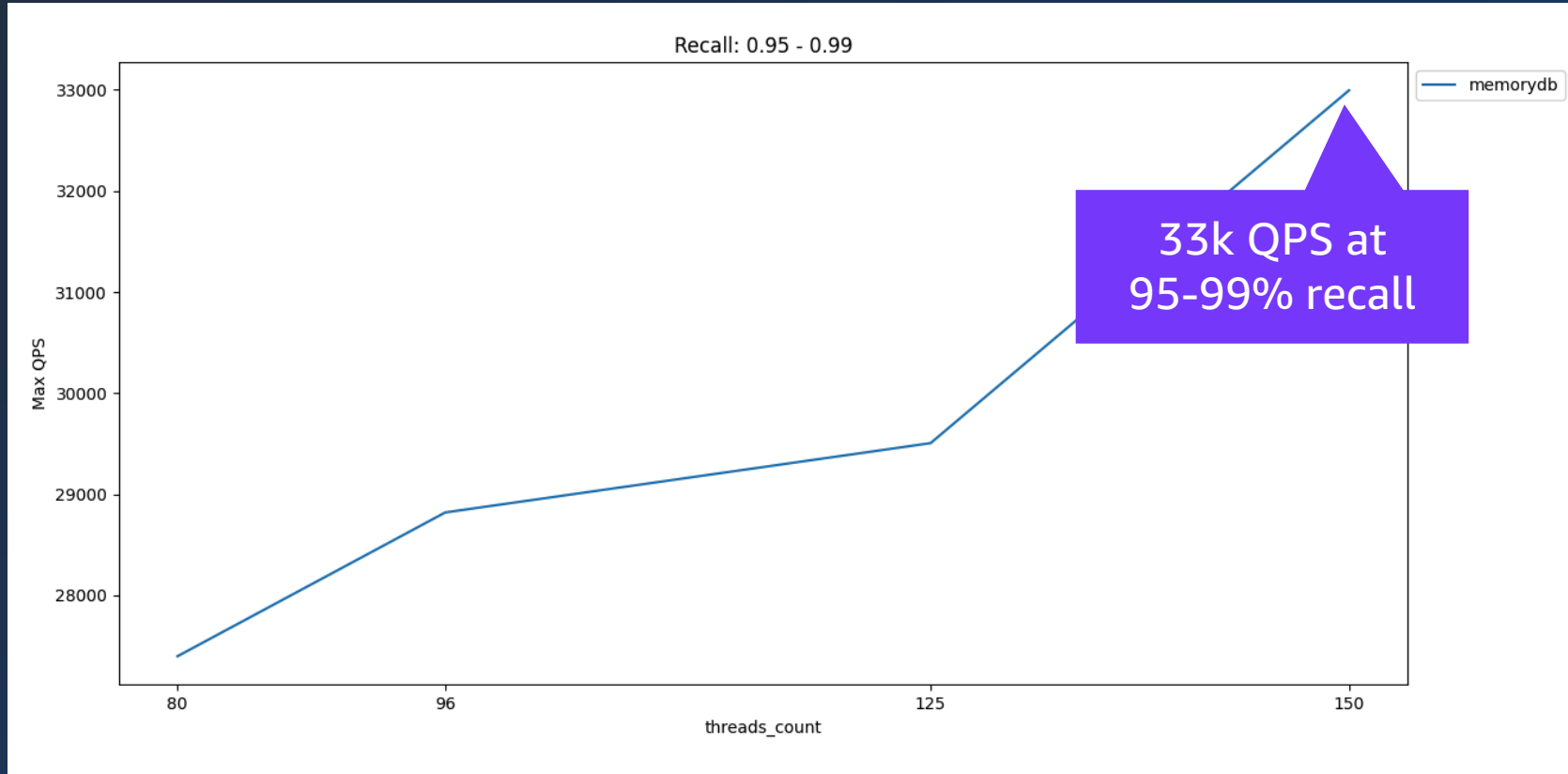
MemoryDB 벡터 검색

```
FT.SEARCH my_index "*=>[KNN 10 @vec $BLOB]"  
PARAMS 2 BLOB "\x12\xa9\xf5\x6c....."
```

```
FT.SEARCH my_index "@Qty:[0 100]=>[KNN 10 @vec $BLOB]"  
PARAMS 2 BLOB "\x12\xa9\xf5\x6c....."
```

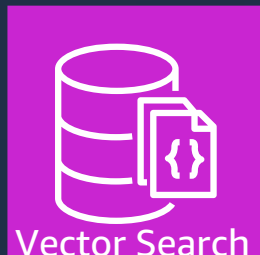
```
FT.SEARCH my_index "@City:{New York | Boston}"  
=>[KNN 10 @vec $BLOB]"  
PARAMS 2 BLOB "\x12\xa9\xf5\x6c....."
```

MemoryDB의 매우 탁월한 처리량과 재현율



MemoryDB 구성

- 읽기 전용 복제본 – 5 개
- r6g.8xlarge
- vCPU 32개, 256GB 메모리
- 128개 차원의 10M 벡터
- HNSW 인덱싱



Amazon DocumentDB (MongoDB 호환) 벡터 검색

Engine version 5.0.0 지원

문서와 벡터의 통합 저장소

문서 저장소와 벡터 데이터베이스를 통합하여 별도의 저장소 관리 필요성 제거

벡터 검색 시 동일한 DB와 문법 사용

데이터 복제, Integration 코드 작성 필요성 제거를 통한 유지 관리 비용 효율성 달성

다차원 벡터와 거리 측정 알고리즘 지원

16,000 차원 지원 (인덱싱할 경우 2,000 차원 지원)
유클리드, 코사인 유사성, 내적 알고리즘 지원

IVFFLAT 벡터 인덱스 지원

IVFFlat 인덱스 지원, List와 Probes 튜닝 파라미터를 통한 향상된 쿼리 성능 제공

차후 HNSW 인덱스도 지원 예정

DocumentDB 벡터 검색 사용

1. 벡터 임베딩 대량 입력

```
db.collection.insertMany([
  {sentence: "나는 고양이와 강이지를 좋아한다." vectorField: [0.82421875, -0.6953125,...]},
  {sentence: " 내 강아지는 매우 귀엽다." vectorField: [0.05883789, -0.020385742,...]},
  {sentence: " 나는 펜으로 글을 쓴다", vectorField: [-0.020385742, 0.32421875,...]}, ... ]);
```

2. 벡터 인덱스 생성

```
db.collection.createIndex (
{ vectorField: "vector" },
{ "name": "index name",
  "vectorOptions": {
    "dimensions": 100, // 벡터 데이터 차원 수
    "similarity": "euclidean", // 또는 cosine, dotProduct 설정 가능
    "lists": 100 } // IVFFlat의 버킷 수
} );
```

DocumentDB 벡터 검색 사용

3. 벡터 검색

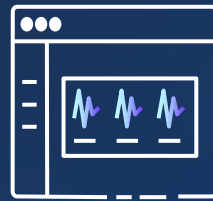
```
db.collection.aggregate ({
$search: {
  "vectorSearch": {
    "vector": [0.82421875, -0.6953125,...], // “나는 동물을 좋아한다”를 검색
    "path": vectorField,
    "k": 5, // 5개의 유사 벡터 검색 결과 반환
    "similarity": "euclidean",
    "probes": 1 // 검색 대상 버킷 수}
  } });
```



Amazon Neptune Analytics 벡터 검색



Neptune Analytics는 대규모 그래프 데이터를 몇 초 만에 분석할 수 있는 메모리 최적화된 그래프 데이터베이스 엔진



Neptune Analytics Notebook에서 Neptune 데이터베이스의 그래프 데이터, Amazon S3 그래프 데이터를 로드하여 분석가능



openCypher 기반 최적화된 분석 그래프 알고리즘을 사용. 그래프 분석과 벡터 검색을 같이 활용하여 데이터의 숨겨진 관계를 더 쉽게 발견

Amazon Neptune Analytics에서의 벡터 검색 설정

Neptune

▼ 데이터베이스

Clusters

스냅샷

서브넷 그룹

파라미터 그룹

이벤트

이벤트 구독

▼ Analytics

Graphs

Snapshots

Import tasks

노트북

Neptune > Graphs > Create Graph

Create graph

You can create a graph by specifying the graph name, the source of the data, network configuration, and the IAM roles to load the data into.

Settings

Graph name

Specify a unique name for the graph.

VectorGraph

The name must be unique cross all graphs owned by your AWS account in the current AWS Region, case-insensitive, 1 to 60 alphanumeric characters or hyphens, first character must be a letter, it can't contain two consecutive hyphens, it can't end with a hyphen.

Data source

Graph import task

☒ Create empty graph

☐ Create Graph from existing source

You can create a new graph by directly importing data into it from Neptune, an S3 bucket or a Neptune Database snapshot.

Memory-Optimized Neptune Credit Units (m-NCU) [정보](#)

Number of m-NCUs to be allocated to your graph

128 m-NCU

Vector search settings

Vector search dimension configuration

☐ No vector dimension

☒ Use vector dimension

You can select a vector dimension for the vector search configuration

Number of dimensions in each vector [정보](#)

The number of dimensions in the vector.

2000

Must be between 0 and 65535



* m-NCU : 메모리 최적화 Neptune 컴퓨팅 단위, 1024 m-NCU까지 지원

Neptune Analytics의 그래프 검색

```
// Algorithms
MATCH (n:airport {country: 'US'})
WITH collect(n) as airports, n.region as region
CALL neptune.algos.bfs.levels(n)
YIELD node, level
RETURN node, level
```

Neptune Analytics에서의 그래프 검색과 벡터 검색

```
MATCH (n:Book {name: 'Travel: Portugal'})
```

```
// 1 //
```

```
CALL neptune.vectors.topKByNode(n, { topK: 10 } )
```

```
YIELD node, score, rank
```

```
// 2 //
```

```
MATCH p=(node)-[*1..3]->(suspicious)
```

```
WHERE (suspicious: seller OR suspicious: lister OR suspicious: buyer)
```

```
// 3 //
```

```
RETURN n, collect(p), score, rank
```

```
ORDER BY rank DESC
```

Knowledge bases for Amazon Bedrock



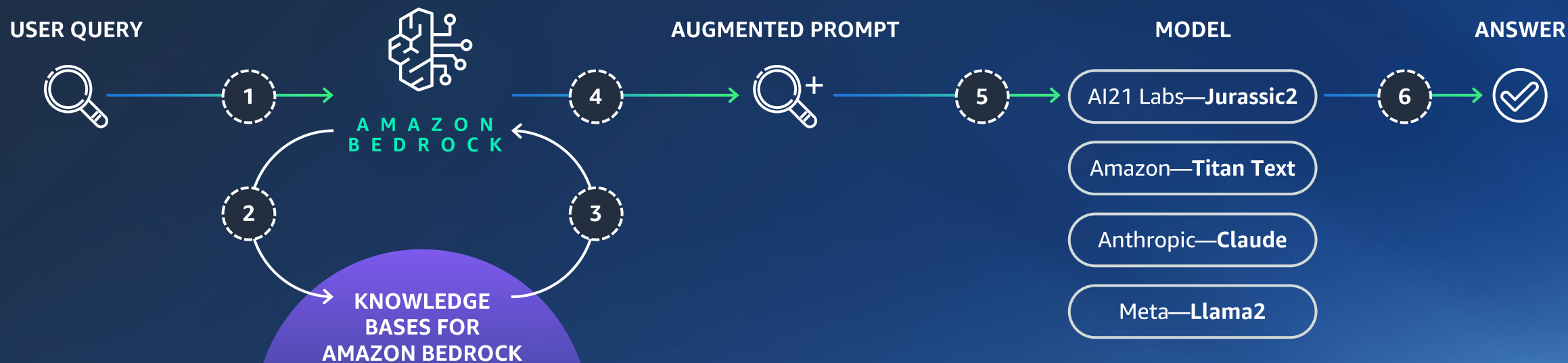
- 검색 증강 생성(RAG)에 대한 기본 지원

FM을 데이터 소스에
안전하게 연결하여
보다 관련성 높은 응답
결과 제공

수집^{ingestion}, 검색^{retrieval},
증강^{augmentation}을
포함한 완전 관리형
RAG 워크플로

멀티 턴 대화를 위한
빌트인 세션 컨텍스트
관리 기능

검색을 통한 자동
인용으로 투명성 향상



Amazon Bedrock 지원 벡터 데이터 베이스



Vector Engine For
Amazon OpenSearch
Serverless



Redis Enterprise
Cloud



Pinecone



Amazon
Aurora

COMING SOON



MongoDB

AWS의 완전 관리형 End to End RAG 워크플로우

아마존 베드락 지식 베이스 데모

AWS vector data store options for RAG



Amazon
Aurora/RDS
PostgreSQL

기존 PostgreSQL 사용, 또는 관계형 DB 전문성이 있을 경우 적합

- Aurora and RDS PostgreSQL에서 모두 사용 가능한 Open-source pgvector 확장 기능
- 벡터간 정확하고 근사값 거리 조회 가능, 3 종류 거리 검색 알고리즘 제공
- 기존 도메인 특화 데이터와 로컬에서 조인



Amazon
OpenSearch
Serverless

NoSQL에 익숙한 경우, 검색 알고리즘 기반으로 한 하이브리드 검색 장점

- k-nn plugin을 통해 시맨틱 검색 지원
- 시맨틱 검색과 키워드 검색 (Lexical 검색) 결합한 하이브리드 검색 가능
- 수평적 확장 가능 (Horizontally scalable / shading)



Amazon
MemoryDB
for Redis

초저지연 벡터 검색 가능

- 시맨틱 검색과 키워드 검색 결합한 하이브리드 검색 가능
- 벡터간 정확하고 근사값 거리 조회 가능, 3 종류 거리 검색 알고리즘 제공



Amazon
DocumentDB

문서 기반 데이터를 활용한 벡터 검색

- 별도의 인프라 관리 필요성 제거
- 2,000 차원, 3 거리 검색, 2 튜닝 파라미터(Lists and Probes)
- 쿼리 성능과 비용 효율성 개선



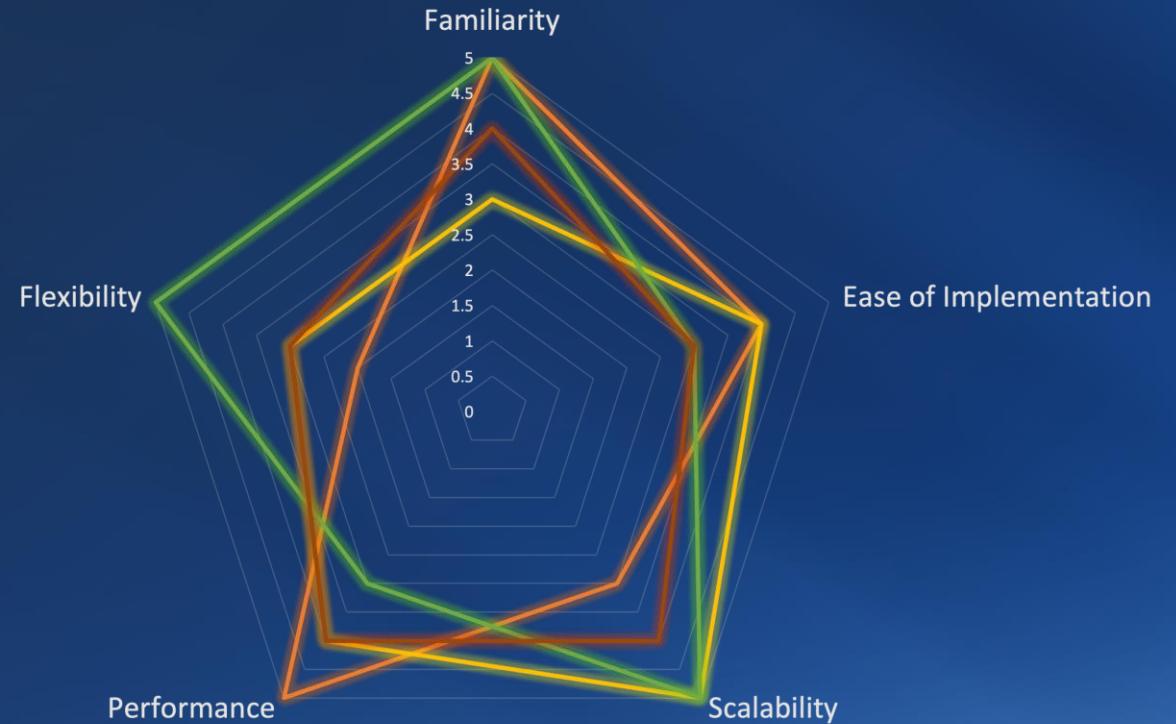
Amazon
Neptune
Analytic

그래프 검색과 벡터 검색을 결합한 데이터의 숨은 패턴 용이

- LLM에서 훈련된 임베딩을 저장하고 그래프 쿼리에서 검색
- • 넵툰의 LangChain 통합을 통한 자연어 그래프 쿼리

벡터 데이터베이스 선택 시 고려사항

- ▶ 친숙함 Familiarity
- ▶ 구현의 용이성 Ease of implementation
 - Abstractions
 - Integrations
- ▶ 확장성 Scalability
 - Vector dimensions supported
 - Number of embeddings
- ▶ 성능 Performance
 - Queries per second (QPS)
 - Recall rate
- ▶ 유연성



AWS 벡터 데이터베이스 비교

	친숙성	구현 용이성	확장성	성능	알고리즘	거리 측정
Aurora/RDS PostgreSQL with pgvector	Technology: PostgreSQL Query language: SQL	Abstraction: AuroraML, Amazon Bedrock integration Integration: Zero-ETL*, federated queries	# of vectors: > 1 billion Dimensions: 16,000 (max. 2000 indexed)	Recall: variable (HNSW ef_search, IVFFLAT probes) Throughput: thousands of QPS and more	Indexing: ivfflat, HNSW Enrichment: Joins	Euclidean Cosine similarity Inner product
Vector engine for OpenSearch Serverless (with k-NN plugin)	Technology: OpenSearch, Elasticsearch Query language: REST API	Abstraction: Neural Search plugin*, Amazon Bedrock integration Integration: OpenSearch Ingestion	# of vectors: > 1 billion Dimensions: 16,000 (max. 1024 for Lucene engine)	Recall: variable (segments, NMSLIB ef_search) Throughput: horizontally scalable	Indexing: NMSLIB and FAISS (HNSW, ivfflat) Enrichment: Keyword search+ semantic search	Euclidean Cosine similarity Dot-product
MemoryDB*	Technology: Redis Query language: API	Abstraction: Neural Search plugin Integration: MemoryDB Ingestion	Dimensions: 32,768	Recall: variable (Flat & HNSW use parameter INITIAL_CAP) Throughput: Single Shard, no horizontally scalable for now	Indexing: HNSW, FLAT Enrichment: Hash search+ semantic search	Euclidean Cosine similarity Dot-product
DocumentDB	Technology: MongoDB, DocumentDB Query language: API	Abstraction: Neural Search plugin Integration: DocumentDB Ingestion	Dimensions: 16,000 (max. 2000 indexed)	Recall: variable [IVFFLAT list sqrt(# of documents)) IVFFLAT probes sqrt (# of lists, HNSW] Throughput: thousands of QPS and more	Indexing: ivfflat, HNSW Enrichment: Keyword search+ semantic search	Euclidean Cosine similarity Dot-product
Neptune	Technology: PostgreSQL Query language: openCypher, Garmin	Abstraction: Neural Search plugin Integration: Neptune Ingestion	Dimensions: 65,535	Recall: variable [IVFFLAT list sqrt(# of documents)) IVFFLAT probes sqrt (# of lists, HNSW] Throughput: thousands of QPS and more	Indexing: Graph indexing	Vector Distance =Euclidean (L2 Distance)



[ivfflat: Inverted File Index - flat](#)

For more efficient and cost effective workloads

[HNSW: Hierarchical Navigable Small World Graphs](#)

For high accuracy, low latency at higher cost (memory requirement)

[FAISS: Facebook AI Similarity Search](#)

[NMSLIB: Non-Metric Space Library](#)

* in preview

AWS 리소스 허브

AWS가 제공하는 AI/ML 및 생성형 AI에 관한 다양한 자료들을 통해 더욱 심층적으로 학습해보세요!

- 기계 학습 여정 가이드
- 기계 학습의 7가지 주요 사용 사례
- 대규모 기계 학습 개발 현대화 전략 가이드
- 기계 학습으로 혁신적인 비즈니스 성과 달성 가이드
- 생성형 AI를 통해 비즈니스 혁신을 가속화 시키는 6단계 가이드
- 생성형 AI 보안에 대한 4가지 핵심 질문과 가이드
- + 외의 다양한 동영상 학습 자료 및 기술 학습 자료!



<https://shorturl.at/wzL13>

[리소스 허브 방문하기](#)



AWS 교육 및 자격증

AWS Skill Builder에서 AWS 전문가들이 개발한 AI & ML 학습 과정들을 무료로 만나보세요!

여러분의 진도에 맞춰 온디맨드 교육 과정 및 Ramp-Up 가이드 등 다양한 학습 리소스를 이용해 여러분의 기술 역량을 향상시키세요.

업계에서 떠오르는 인공 지능과 기계 학습 및 딥 러닝을 어떻게 클라우드에 적용시킬 수 있는지 배울 수 있습니다.

AWS Certified Machine Learning – Specialty 자격증을 통해 여러분의 전문성을 증명하십시오. AWS Skill Builder 에서 학습하실 수 있습니다.



<https://bit.ly/3FnxDH7>

Learn your way [explore.skillbuilder.aws](https://skillbuilder.aws) »



AWS Innovate – AI/ML & Data 특집에 참석해 주셔서 감사합니다.

저희가 준비한 강연, 어떻게 보셨나요?
더 나은 세미나를 위하여 **설문을 꼭 작성해 주세요!**



aws-korea-marketing@amazon.com



twitter.com/AWSKorea



facebook.com/amazonwebservices.ko



youtube.com/AWSKorea



linkedin.com/company/amazon-web-services



twitch.tv/aws

Thank you!

