Choosing an AWS vector database for RAG use cases

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Choosing an AWS vector database for RAG use cases

Table of Contents

Choosing an AWS vector database for RAG use cases

Mayuri Shinde, Ivan Cui, and Anand Bukkapatnam Tirumala, Amazon Web Services

Vector databases are becoming increasingly important for organizations that implement generative AI applications. These databases store and manage *vectors*, which are numerical representations of data that enable processing of text, images, and other content in ways that capture their meaning and relationships.

As organizations explore vector database options on AWS they need to understand the capabilities, trade-offs, and best practices for different solutions. This guide helps you compare commonly used vector stores on AWS and make informed decisions about which options best suit your specific needs or use case. Whether you're implementing retrieval augmented generation (RAG), building recommendation systems, or developing other AI applications, this guide provides a framework to help you evaluate and choose a vector database solution.

Intended audience

This guide is intended for people in these roles:

- Data scientists and machine learning (ML) engineers who use vector databases to store and retrieve high-dimensional data for ML models.
- Data engineers who design and implement data pipelines that include vector databases for storing and processing high-dimensional data.
- MLOps engineers who use vector databases as part of the ML pipeline to store and serve model outputs or intermediate representations.
- Software engineers who integrate vector databases into applications that require similarity search or recommendation systems.
- DevOps engineers who are responsible for deploying and maintaining vector databases in production environments, ensuring scalability and reliability.
- AI researchers who use vector databases to store and analyze large datasets of embeddings or feature vectors.
- AI product managers who need to understand the capabilities and limitations of vector databases to make informed decisions about product features and architecture.

Overview of vectors

Vectors are numerical representations that help machines understand and process data. In generative AI, they serve two key purposes:

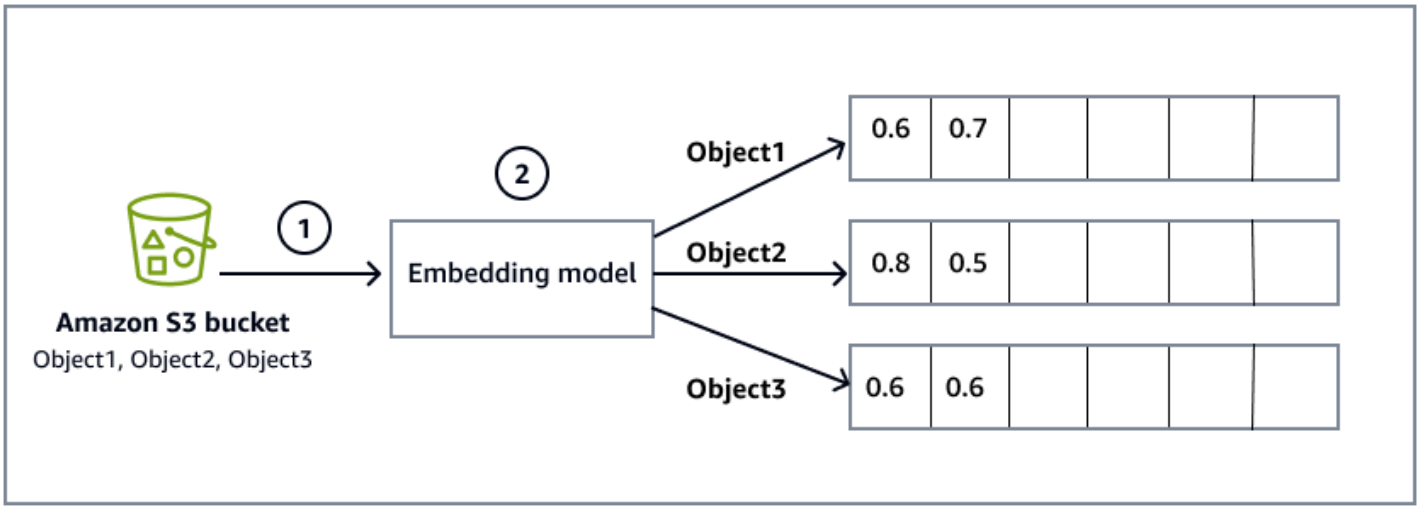
- Representing latent spaces that capture data structure in compressed form
- Creating embeddings for data, such as words, sentences, and images

Embedding models like Word2Vec, GloVe, and Amazon Titan Text Embeddings convert data into vectors through a process called *embedding*. These embedding models can do the following:

- Learn from context to represent words as vectors
- Place similar words closer together in vector space
- Enable machines to process data in a continuous space

The following diagram provides a high-level overview of the embedding process:

1. An Amazon Simple Storage Service (Amazon S3) bucket contains files that are the data sources from which the system will read and process information. The Amazon S3 bucket is specified during the Amazon Bedrock knowledge base configuration, which also includes syncing data with the knowledge base.
2. The embedding model converts the raw data from the object files in the Amazon S3 bucket into vector embeddings. For example, `Object1` is converted into a vector `[0.6, 0.7, ...]` that represents its content in a multi-dimensional space.



Word embeddings are crucial for natural language processing (NLP) because they do the following:

- Capture semantic relationships between words
- Enable generation of contextually relevant text
- Power large language models (LLMs) to produce human-like responses

Overview of vector databases

A vector database is a specialized system that stores and queries high-dimensional vectors efficiently. These databases are fundamental for Retrieval Augmented Generation (RAG) applications.

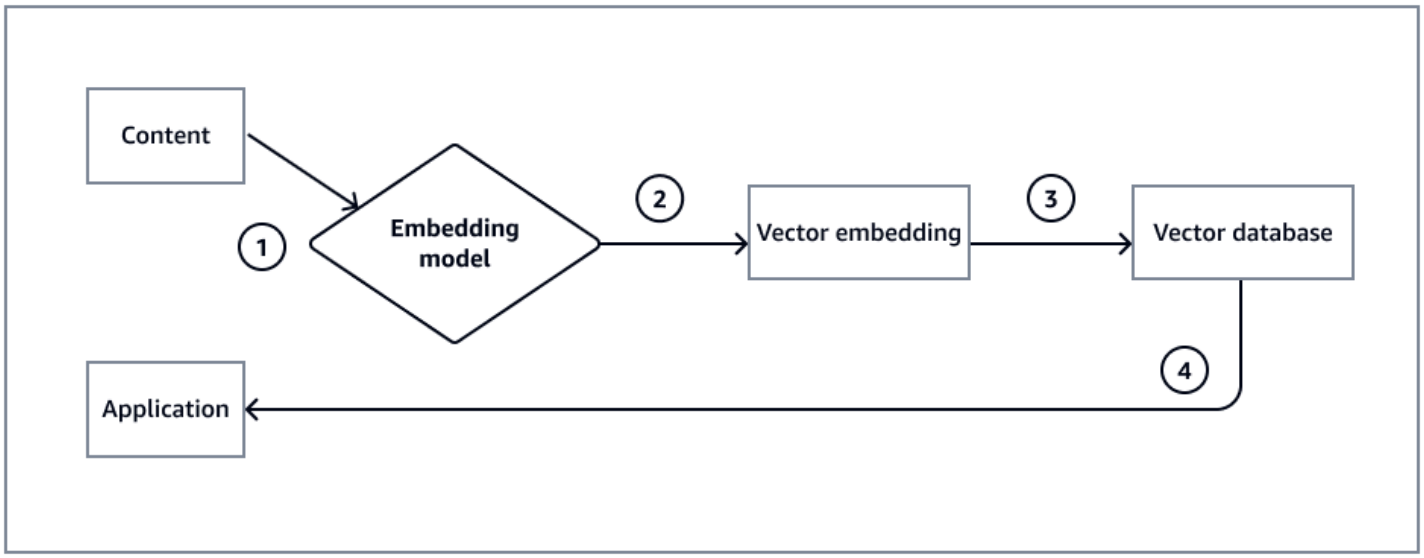
Vector databases handle data conversion and storage in the following ways:

- Objects (such as audio, images, and text files) are converted to vectors by using embedding models.
- Vectors are stored in specialized data formats.
- Vector databases enable rapid similarity searches.

Vector databases offer several key advantages over traditional databases, making them particularly well-suited for modern data challenges. They are specifically optimized for vector operations and handle high-dimensional data efficiently. They also specialize in similarity searches that traditional databases struggle with. Beyond these core capabilities, vector databases are built to meet the evolving demands of ML and generative AI applications. They excel at large-scale vector storage and use distributed computing to balance workloads across multiple nodes. This provides scalability and performance as data volumes grow.

The following diagram shows a RAG implementation:

1. Content, such as documents, PDFs, or text files, is fed into the embedding model as raw data for processing.
2. The embedding model transforms the raw data into numerical vectors, which represent the semantic meaning of the content.
3. The generated vector embeddings are stored in a vector database that is optimized for the storage and retrieval of high-dimensional vectors.
4. Applications can now query the vector database in response to use cases such as semantic search and content recommendation.



Choosing an inappropriate vector database for a RAG solution can lead to significant struggles and limitations including the following:

- Poor query performance
- Scalability bottlenecks
- Data ingestion challenges
- Lack of advanced features, such as filtering and ranking
- Integration difficulties with other systems
- Persistence and durability concerns
- Concurrency and consistency issues in environments with multiple users
- Higher licensing costs or vendor lock-in
- Limited community support and resources
- Potential security and compliance risks

Vector database options

AWS offers a diverse range of vector database solutions to support different use cases and requirements in generative AI applications. These options can be broadly categorized into individual database services and managed service offerings, each with distinct characteristics and advantages. Understanding these options is crucial for organizations looking to implement vector search capabilities effectively while maintaining optimal performance, scalability, and cost efficiency.

For more information about vector database solutions, see the following sections:

- Individual vector database options
- Managed service option
- Choosing the right vector database

Individual vector database options

The individual vector database options on AWS include Amazon Kendra, Amazon OpenSearch Service, Amazon RDS for PostgreSQL with pgvector, Amazon MemoryDB, Amazon DocumentDB, Amazon Neptune Analytics, and Amazon S3 Vector. (An open-source extension, pgvector adds the ability to store and search ML-generated vector embeddings.) These solutions offer different approaches to vector search, allowing organizations to choose based on their existing infrastructure, technical requirements, and specific use cases.

Amazon Kendra

Amazon Kendra is an enterprise-grade intelligent search service that uses natural language processing and advanced machine learning algorithms to return specific answers to search questions from your data. Amazon Kendra simplifies the implementation of search functionality, making it an effective backend solution for generative AI applications.

Other key features of Amazon Kendra include the following:

- Native connections to over 40 data sources
- Built-in data preparation capabilities
- Quick setup that doesn't require deep technical expertise

Benefits of Amazon Kendra include the following

- Automated data processing (chunking, ingestion, retrieval)
- Powerful customization options:
  - Facet search
  - Search analytics
  - Tuning search relevance
- Simple programmatic access through the AWS SDK for Python (Boto3)

For more information, see Benefits of Amazon Kendra in the Amazon Kendra documentation.

Amazon OpenSearch Service

Amazon OpenSearch Service is a managed service that helps you deploy, operate, and scale OpenSearch Service clusters in the AWS Cloud.

Core capabilities of OpenSearch Service include the following:

- Open-source search and analytics engine
- Distributed architecture
- Real-time data processing

Some advantages of using OpenSearch Service include the following:

- Horizontal scalability
- RESTful API support
- Handles structured and unstructured data
- Real-time data analysis
- Suitable for various deployment sizes

For more information, see Features of Amazon OpenSearch Service in the OpenSearch Service documentation.

Amazon RDS for PostgreSQL with pgvector

Amazon RDS for PostgreSQL with pgvector combines the AWS managed relational database service with PostgreSQL's vector processing extension. This combination enables organizations to store and query high-dimensional vectors while maintaining Amazon RDS. The solution is particularly suitable for generative AI applications that require real-time vector operations without the overhead of managing database infrastructure.

Key benefits of Amazon RDS for PostgreSQL with pgvector include the following:

- High availability
- Automatic failover
- Cost-effective (pay-per-use)
- Built-in monitoring
- Real-time vector data integration

For more information, see Advantages of Amazon RDS in the Amazon RDS documentation.

Amazon DocumentDB

Amazon DocumentDB (with MongoDB compatibility) is a document database that offers native vector search capabilities in version 5.0 and later. It combines the flexibility of JSON-based document storage with vector search, supporting both hierarchical navigable small world (HNSW) and Inverted File Flat (IVFFlat) indexing methods.

Core capabilities of Amazon DocumentDB include the following:

- Store and index vectors up to 2,000 dimensions (up to 16,000 dimensions without indexing)
- Millisecond response times for vector similarity searches
- Support for euclidean, cosine, and dot product distance metrics
- Seamless integration with existing MongoDB-compatible applications

Use Amazon DocumentDB in the following situations:

- For applications that are already using MongoDB APIs and that need vector search capabilities
- For use cases that require flexible document data structures combined with semantic search

- For scenarios that need both traditional document queries and vector similarity searches
- For applications that provide product recommendations, personalization, chat assistants, and fraud detection

For more information, see Vector search for Amazon DocumentDB in the Amazon DocumentDB documentation.

Amazon MemoryDB

Amazon MemoryDB is a Redis-compatible, in-memory database that delivers the fastest vector search performance among popular vector databases on AWS. It provides sub-millisecond query latencies with multi-Availability Zone durability.

Core capabilities of MemoryDB include the following:

- Store application data and millions of vectors in a single database
- Single-digit millisecond query and update response times
- Highest recall rates at the fastest performance on AWS
- Support for up to 32,768 dimensions per vector
- Real-time semantic search and caching capabilities

Use MemoryDB in the following situations:

- For real-time applications that require ultra-low latency (sub-10ms)
- For high-throughput workloads with millions of requests per day
- For use cases such as real-time recommendation engines, semantic caching, and anomaly detection
- For applications that need both in-memory data store and vector search capabilities

For more information, see Vector search in the MemoryDB documentation.

Amazon Neptune Analytics

Amazon Neptune Analytics is a graph analytics engine that offers native vector search capabilities, making it ideal for Graph Retrieval Augmented Generation (GraphRAG) use cases. It combines vector similarity search with graph traversals and algorithms.

Core capabilities of Neptune Analytics include the following:

- Analyze tens of billions of relationships within seconds
- Combine vector search with graph algorithms (path finding, community detection, centrality)
- Support for GraphRAG applications with topological knowledge
- Up to 80 times faster than existing graph analytical solutions
- Integration with Amazon Bedrock for fully managed GraphRAG

Use Neptune Analytics in the following situations:

- For GraphRAG applications that require knowledge graphs with vector embeddings
- For use cases that require traversing complex relationships alongside vector similarity
- For applications that require explainable AI responses with relationship context
- For scenarios such as customer 360 views, fraud detection networks, and knowledge discovery

For more information, see the Amazon Neptune Analytics documentation.

Amazon S3 Vectors

Amazon S3 Vectors is the first cloud object store in AWS with native vector storage and query capabilities. It provides purpose-built, cost-optimized vector storage for AI applications that require massive scale.

Core capabilities of Amazon S3 Vectors include the following:

- Storage for up to 2 billion vectors per index with support for up to 10,000 indexes per vector bucket
- Sub-100 ms query latency that is optimized for long-term storage and infrequent access patterns
- Up to 90% cost reduction for vector operations compared to specialized vector databases
- Serverless architecture with automatic scaling and 99.999999999% (11 9s) durability

Use Amazon S3 Vectors in the following situations:

- For applications that require storage of billions of vectors at minimal cost

- For workloads that tolerate sub-second query latency (100 ms or more) rather than sub-10 ms
- For long-term vector retention and archival use cases
- For RAG applications with infrequent retrieval patterns
- For organizations that prioritize storage economics over ultra-low latency

Amazon S3 Vectors integrates natively with Amazon Bedrock Knowledge Bases and works well in tiered architectures with Amazon OpenSearch Service. You can use Amazon S3 Vectors for cold storage and use OpenSearch Service for hot queries.

For more information, see Working with S3 Vectors and vector buckets in the Amazon S3 documentation.

Managed service option

Amazon Bedrock Knowledge Bases represents the AWS fully managed approach to vector database implementation. The service's flexibility in storage options, combined with its automated management features, makes it particularly valuable for organizations seeking to implement RAG without managing complex infrastructure.

With Amazon Bedrock Knowledge Bases, you can create, maintain, and query knowledge bases that enhance your foundation models using RAG. This service simplifies the complex process of implementing RAG by managing the entire data ingestion, vectorization, and retrieval pipeline.

Key benefits of Amazon Bedrock Knowledge Bases include the following:

- Simplified data processing
  - Automatic data ingestion and chunking
  - Built-in text extraction from multiple file formats
  - Managed vector embeddings generation
  - Automatic metadata extraction and indexing
- Streamlined RAG implementation
  - Pre-configured retrieval strategies
  - Automatic context window optimization
  - Built-in relevancy tuning
  - Semantic search capabilities out of the box

- Security and governance
  - Integrated AWS Identity and Access Management (IAM) controls
  - Data encryption at rest and in transit
  - VPC support
  - Audit logging with AWS CloudTrail

Amazon Bedrock Knowledge Bases supports multiple vector store options, including:

- Amazon Aurora PostgreSQL with pgvector
- Amazon Neptune Analytics
- Amazon EMR Serverless
- Amazon S3 Vectors
- Pinecone
- Redis Enterprise Cloud

This managed service handles automated data ingestion, vectorization, and retrieval. This simplifies RAG implementations.

For detailed information about each supported vector store, see the Amazon Bedrock Knowledge Bases documentation.

Choosing the right vector database

Select your vector database based on these key decision factors:

- **If you need MongoDB-compatible document database with vector search** – Choose Amazon DocumentDB. This is ideal when your application uses MongoDB APIs and you want to add semantic search capabilities without managing separate vector infrastructure.
- **If you need ultra-low latency for real-time applications** – Choose Amazon MemoryDB. This provides the fastest vector search performance on AWS with sub-millisecond response times. It's ideal for real-time recommendation engines and high-throughput applications.
- **If you need graph-based knowledge representations with vector search** – Choose Amazon Neptune Analytics. This is best for GraphRAG applications that need to traverse complex relationships and perform graph-based queries alongside vector searches, providing explainable AI responses.

- **If you need to combine relational queries with vector search** – Choose Amazon Aurora PostgreSQL with pgvector. This option is ideal when your application requires both traditional SQL operations and vector similarity searches within the same database.
- **If you require high-throughput queries with sub-10 ms latency** – Choose Amazon OpenSearch Service. It excels at handling high-frequency queries and real-time applications and includes recent GPU acceleration improvements.
- **If you need to store billions of vectors cost-effectively** – Choose Amazon S3 Vectors. This option provides up to 90% cost savings and is ideal for applications with infrequent retrieval patterns (minutes to hours between queries) that can tolerate sub-100 ms latency.
- **If you need full-text search alongside vector search** – Choose Amazon OpenSearch Service. This option combines powerful full-text search capabilities with vector search in a single platform.

Vector database comparison

AWS provides multiple approaches to implementing vector search capabilities, ranging from individual vector databases to Amazon Bedrock Knowledge Bases, which is a fully managed service. When evaluating these options, organizations must consider various aspects including architecture, scalability, integration capabilities, performance characteristics, and security features.

Individual vector databases

The following table provides an overview of key features of several AWS individual vector database solutions, focusing on their architectures, scaling capabilities, data source integrations, and performance characteristics.

Feature	Amazon Kendra	Amazon OpenSearch Service	Amazon RDS for PostgreSQL with pgvector	Amazon DocumentDB	Amazon MemoryDB	Amazon Neptune Analytics	Amazon S3 Vectors
Primary use case	Enterprise search and RAG	Distributed search and analytics	Relational DB with vector support	DocumentDB with vector search	Real-time in-memory vector search	Graph analytics with vector search	Cost-optimized vector storage
Architecture	Fully managed	Distributed cluster	Relational database	Document-oriented	In-memory database	Graph analytics engine	Serverless object storage
Data model	Document-based	JSON documents	Relational tables	JSON documents	Key-value with JSON	Property graph	Object storage
Vector dimensions	Managed automatically	Up to 16,000	Configurable	Up to 2,000 (indexed)	Up to 32,768	Configurable	Up to 4,096

				; 16,000 (unindexed)			
Indexing methods	Automatic	HNSW, IVF	HNSW, IVFFlat	HNSW, IVFFlat	HNSW	Native graph and vector	Automatic
Distance metrics	Automatic	Cosine, Euclidean, dot product	Cosine, Euclidean, inner product	Cosine, Euclidean, dot product	Cosine, Euclidean, inner product	Cosine, Euclidean	Cosine, Euclidean
Query latency	Sub-second	Sub-10 ms (GPU-accelerated)	10-100 ms	Millisecond	Sub-millisecond	Sub-second	Sub-100 ms
Scaling model	Automatic	Horizontal (add nodes)	Vertical and read replicas	Horizontal (add instances)	Vertical and replicas	Automatic	Automatic (serverless)
Maximum vectors	Managed	Billions (cluster-dependent)	Millions (instance-dependent)	Millions per collection	Millions per database	Billions	2 billion per index; 10,000 indexes per bucket
Throughput	High	Very high (thousands of QPS)	Medium	High	Very high (millions of requests per day)	High	Medium (optimized for infrequent queries)

Data durability	99.999999999% (11 9s)	Configurable with replicas	99.99% (Multi-AZ)	99.99% (Multi-AZ)	99.99% (Multi-AZ)	99.99%	99.999999999% (11 9s)
Consistency model	Eventual	Eventual (configurable)	Strong (ACID)	Eventual	Strong	Strong	Strong
Additional capabilities	40 or more data connectors, NLP	Full-text search, analytics, dashboards	SQL queries, ACID transactions	MongoDB API compatibility	Redis API compatibility, caching	Graph algorithms, traversals	Amazon S3 integration, lifecycle policies
Pricing model	Pay per query and storage	Instance hours and storage	Instance hours and storage	Instance hours and storage	Instance hours and storage	Capacity units and storage	Storage, queries, and data transfer
Cost optimization	Usage-based	Reserved instances, auto-scaling	Reserved instances, Aurora Serverless	Reserved instances	Reserved instances	Auto-scaling	Up to 90% savings vs specialized DBs
Best for	Enterprise search with minimal setup	High-throughput, low-latency queries	Hybrid SQL and vector workloads	MongoDB-compatible apps needing vectors	Real-time, ultra-low latency apps	GraphRAG and knowledge graphs	Long-term, cost-effective storage

Ideal query pattern	Frequent enterprise searches	High-frequency real-time queries	Mixed SQL and vector queries	Document queries with semantic search	Millions of requests per day	Graph traversals with vector search	Infrequent queries (minutes to hours)
Setup complexity	Low (fully managed)	Medium (cluster configuration)	Medium (extension setup)	Medium (cluster configuration)	Medium (cluster configuration)	Low (fully managed)	Low (serverless)
Team expertise required	Minimal	OpenSearch or Elasticsearch	PostgreSQL, SQL	MongoDB	Redis	Graph databases	Amazon S3, basic vector concepts

Managed service – Amazon Bedrock Knowledge Bases

Amazon Bedrock Knowledge Bases provides a fully managed solution with multiple vector storage options. The following table compares these storage options.

Feature	**Aurora PostgreSQL with pgvector**	**Neptune Analytics**	**OpenSearch Service Serverless**	**Amazon S3 vectors**	**Pinecone**	**RedisEnterprise Cloud**
Primary use case	Relational DB with vector RAG	Graph-based vector search for GraphRAG	Knowledge management RAG	Cost-optimized vector RAG	High-performance vector search	In-memory vector search
Architecture	Fully managed relational	Fully managed graph analytics	Fully managed serverless	Serverless object storage	Fully managed hybrid cloud	Fully managed in-memory

Data model	Relational tables	Property graph	JSON documents	Object storage	Purpose-built vectors	Key-value with vectors
Vector storage	Through pgvector extension	Native graph vectors	Through OpenSearch engine	Native Amazon S3 vector storage	Native vector database	In-memory vectors
Amazon Bedrock integration	Native	Native	Native	Native	Native	Native
Automatic ingestion	Yes (via Amazon Bedrock)	Yes (via Amazon Bedrock)	Yes (via Amazon Bedrock)	Yes (via Amazon Bedrock)	Yes (via Amazon Bedrock)	Yes (via Amazon Bedrock)
Automatic vectorization	Yes (via Amazon Bedrock)	Yes (via Amazon Bedrock)	Yes (via Amazon Bedrock)	Yes (via Amazon Bedrock)	Yes (via Amazon Bedrock)	Yes (via Amazon Bedrock)
Scaling	Auto-scaling (Aurora Serverless)	Automatic graph scaling	Automatic serverless	Automatic (billions of vectors)	Auto-scaling pods	Auto-scaling clusters
Query performance	High for relational or vector	High for graph vectors	High	Medium (100 ms or more latency)	Very high	Very high
Maximum vectors	Millions (instance-dependent)	Billions	Billions	2 billion per index	Billions	Millions (memory-dependent)

Additional capabilities	SQL queries, ACID transactions	Graph algorithms, traversals	Full-text search, analytics	Amazon S3 lifecycle, tiering	Metadata filtering, namespaces	Redis data structures, caching
Cost optimization	Moderate (Aurora Serverless)	Moderate (capacity units)	High (serverless, pay-per-use)	Very high (up to 90% savings)	Moderate (pod-based pricing)	Low (in-memory premium)
Best for	Hybrid SQL/ vector workloads	Connected knowledge graphs	Full-text with vector search	Long-term, infrequent-access vectors	Real-time vector search at scale	Ultra-low latency needs
Ideal query pattern	Mixed SQL and vector queries	Graph traversals with vectors	Frequent searches with analytics	Infrequent retrieval (minutes to hours)	High-frequency real-time queries	Millions of requests per second
Setup with Amazon Bedrock	Simple (managed by Amazon Bedrock)	Simple (managed by Amazon Bedrock)	Simple (managed by Amazon Bedrock)	Simple (managed by Amazon Bedrock)	Simple (managed by Amazon Bedrock)	Simple (managed by Amazon Bedrock)
Data residency	AWS Regions	AWS Regions	AWS Regions	AWS Regions	Multi-cloud (AWS and others)	Multi-cloud (AWS and others)
Pricing model	Instance hours and storage	Capacity units and storage	Compute and storage (serverless)	Storage, queries, and transfer	Pod hours and storage	Node hours and storage

Choosing between individual and managed options

Consideration	Choose individual vector DB	Choose Amazon Bedrock Knowledge Bases (managed)
RAG implementation	You want full control over RAG pipeline	You want fully managed RAG with minimal setup
Customization	You need custom retrieval logic and preprocessing	Standard RAG patterns meet your needs
Existing infrastructure	You already have the database deployed	You're starting fresh or want simplified management
Team expertise	Your team has database administration expertise	You prefer to focus on application logic, not infrastru cture
Integration complexity	You need deep integration with existing systems	You want quick integration with Amazon Bedrock models
Operational overhead	You can manage database operations	You want AWS to handle operations
Cost structure	You prefer direct database pricing	You prefer unified Amazon Bedrock pricing
Time to market	You have time for custom implementation	You need rapid deployment

Cost comparisons and considerations

Understanding the cost structure of different vector database options is essential for making informed implementation decisions. The following table outlines some key cost considerations for various vector database solutions, including both individual databases and managed services. Each option has distinct pricing factors that can impact total cost of ownership (TCO), from pay-as-you-go models to infrastructure and operational costs.

Vector database	Cost model	Cost considerations
Amazon Kendra	Pay as you go, based on queries	Costs can vary based on the number of queries and the amount of data indexed. Additional charges can apply for data storage and data transfer. For more informati on, see Amazon Kendra pricing.
Amazon OpenSearch Service	Pay as you go, based on instance hours and storage	Costs include instance hours, storage (Amazon EBS volumes), data transfer, and optional UltraWarm storage. Reserved Instances can provide up to 30% savings. GPU-accelerated instances offer better price-performance for vector workloads. For more information, see Amazon OpenSearch Service pricing.
Open-source OpenSearch	Open-source, no direct cost (you don't have to pay to download or use the software, and there are no license costs)	Costs include infrastructure (such as servers and storage) and operational costs (such as maintenance and monitoring). Organizations need to budget

		for personnel to manage and maintain the infrastructure.
Amazon RDS for PostgreSQL with pgvector	Pay as you go, based on usage	Costs include database instance types, storage, data transfer, and backups. Additional charges can apply for data transfer, instance types, and storage beyond the AWS Free Tier. For more information, see Amazon RDS pricing.
Amazon DocumentDB	Pay as you go, based on instance hours and storage	Costs include instance hours, storage (GB-month), I/O requests, backup storage, and data transfer. Elastic clusters enable dynamic scaling. Reserved Instances available for cost optimizat ion. For more information, see Amazon DocumentDB pricing.
Amazon MemoryDB	Pay as you go, based on node hours and data storage	Costs include node hours (per node type), data storage (GB-hour), snapshot storage, and data transfer. Reserved Nodes can provide up to 55% savings. Optimized for high-throughput, low-latency workloads. For more information, see Amazon MemoryDB pricing.

Amazon Neptune Analytics	Pay as you go, based on capacity units	Costs include Neptune Capacity Units (NCUs), storage (GB-month), and data transfer. Auto-scaling based on workload with no upfront commitments. Minimum 128 NCUs required. For more information, see Amazon Neptune pricing.
Amazon S3 Vectors	Pay as you go, based on storage and requests	Costs include storage (GB-month), PUT and GET requests, vector index management, and data transfer. Provides up to 90% cost savings compared to specialized vector databases. Amazon S3 Intelligent-Tiering and lifecycle policies available for additional optimization. For more information, see Amazon S3 pricing.
Amazon Bedrock Knowledge Bases	Pay as you go, based on usage	Costs can vary based on the usage of the knowledge base and additional services such as Amazon OpenSearch Serverless. Additional charges can apply for data storage, data transfer, and additional features. For more informati on about pricing, see Amazon OpenSearch Service pricing.

Vector database use cases

The following examples highlight how different vector database options can be used effectively to enhance knowledge management, improve operational efficiency, and deliver better business outcomes. These use cases illustrate practical applications of the vector database solutions discussed earlier in this guide and provide insights into their real-world performance and benefits.

Knowledge management with Amazon Kendra

Customer problem – One of the largest general contractors in Japan was facing a decline in experienced personnel. The company needed a way to transfer the knowledge and skills of the experience personnel to the younger generation efficiently. They required a solution to capture and disseminate complex construction engineering knowledge and past experiences.

AWS solution – To address this problem, the customer turned to Amazon Kendra, an AI solution that could quickly and accurately handle their internal knowledge base and allow natural language queries. With Amazon Kendra, employees can now find the information they need much faster, improving productivity and facilitating knowledge transfer from experienced personnel to younger staff.

Impact – By implementing a generative AI chatbot powered by Amazon Kendra, the company created a unified knowledge platform. The chatbot allows employees to quickly access technical knowledge and past experiences on construction engineering. This solution has significantly improved the efficiency of knowledge transfer and decision-making processes within the organization, helping to ensure that valuable expertise is preserved and easily accessible to all employees. The cost of this solution may vary depending on your usage and configuration. For a detailed cost estimate, see the AWS Pricing Calculator. For vector database cost estimation, see the Cost comparison and considerationssection of this guide or see Amazon Kendra pricing.

For information about other customer use cases, see Amazon Kendra customers.

Real-time analytics with OpenSearch Serverless

Customer problem – A leading financial services provider faced the challenge of managing an enormous data ecosystem. It processed 300 million authorizations and 90 billion transactions annually, accumulating to approximately 1.1 petabytes (PB) of data. The existing system, serving

300,000 users who required access to over 6,000 reports, needed modernization to provide global consistency and enable real-time decision-making.

AWS solution – The solution architecture used foundation models available through Amazon Bedrock (including Anthropic, Sonnet 3, Sonnet 3.5, and Haiku) for natural language processing. The customer chose OpenSearch Serverless as the vector database for its superior scalability and ability to handle the massive data volume efficiently. This architecture enabled seamless processing of complex queries and dynamic report generation.

Impact – The implementation achieved a 50 percent increase in productivity by eliminating the need for manual generation of over 100 business intelligence dashboards. Users can now generate reports through natural language queries with response times of between 20-40 seconds. The cost of this solution may vary depending on your usage and configuration. For a detailed cost estimate, see the AWS Pricing Calculator. For vector database cost estimation, see the Cost comparison and considerationssection of this guide or see Amazon OpenSearch Service pricing. For information about other customer use cases, see Amazon OpenSearch Serverless.

Next steps and resources

After reviewing this guide, consider the following actions to move from understanding to implementation:

1. Evaluate your current needs:
   - Assess your existing database infrastructure and expertise.
   - Document your specific vector search requirements.
   - Define your performance, scaling, and cost targets.
2. Choose one of the following options to test vector database options:
   - **Option 1:** Set up a proof of concept using your preferred vector database solution.
   - **Option 2:** Experiment with sample datasets in Amazon Bedrock Knowledge Bases. Try the quick-create experience for an Amazon Bedrock Knowledge Base. For an example, see Quick create an Aurora PostgreSQL Knowledge Base for Amazon Bedrock in the Aurora documentation.
3. Review additional resources.
4. Get expert help:
   - Contact your AWS account team or AWS Solutions Architects for implementation guidance.
   - Engage with AWS Partners that specialize in vector databases.
5. Plan your production deployment:
   - Create a migration strategy if moving from existing databases.
   - Develop a scaling plan for your chosen solution.
   - Design your monitoring and maintenance procedures.

Resources

The following resources can help you in choosing a vector database.

AWS blog posts

- Accelerate your generative AI application development with Amazon Bedrock Knowledge Bases Quick Create and Amazon Aurora Serverless
- Amazon OpenSearch Service's vector database capabilities explained

- Dive deep into vector data stores using Amazon Bedrock Knowledge Bases
- Leverage pgvector and Amazon Aurora PostgreSQL for Natural Language Processing, Chatbots and Sentiment Analysis

AWS service documentation

- Choosing an AWS database service
- How Amazon Bedrock knowledge bases work
- Neptune Analytics documentation
- Overview of Amazon Web Services: Databases
- Using Aurora PostgreSQL as a Knowledge Base for Amazon Bedrock
- Working with Amazon Aurora PostgreSQL
- Amazon DocumentDB
- Amazon MemoryDB
- Amazon S3 Vectors

Other AWS resources

- Amazon Bedrock Knowledge Bases
- Vector Databases & Embeddings
- Vector Databases for generative AI applications
- What are Embeddings in Machine Learning?

Other resources

- About PostgreSQL
- pgvector documentation
- Pinecone as a Knowledge Base for Amazon Bedrock
- Redis Enterprise Cloud on AWS

Document history

The following table describes significant changes to this guide.

Change	Description	Date
Added AWS services	We added information about using Amazon DocumentDB, Amazon MemoryDB, Amazon S3 Vectors, and Amazon Neptune Analytics.	March 30, 2026
Initial publication	—	March 6, 2025