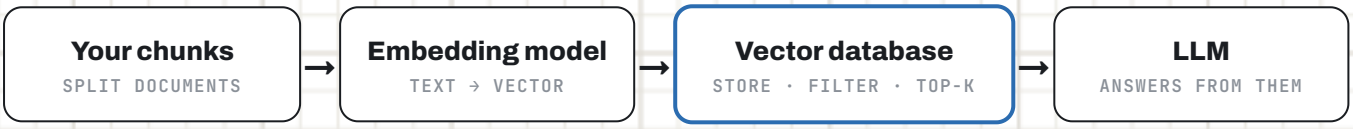
Which Vector Database for RAG?

9 OPTIONS, COMPARED

Pinecone, pgvector, Qdrant, Weaviate, Milvus, Elasticsearch, Redis, Chroma and FAISS: what each one is, and when to pick it.



MANAGED SERVICE

Someone else runs it. You call an API.

01 **Pinecone**

INSIDE WHAT YOU ALREADY RUN

Add vectors to a database or search engine you have.

02 **pgvector (Postgres)**
06 **Elasticsearch**
07 **Redis**

PURPOSE-BUILT, OPEN SOURCE

Vector databases first. Self-host, or use their cloud.

03 **Qdrant**
04 **Weaviate**
05 **Milvus**
08 **Chroma**

LIBRARY, NOT A DATABASE

Fast search code you build a service around.

09 **FAISS**

SUMIT KUMAR DASH
SKDNEURON.COM · @SKDNEURON

SAVE FOR YOUR NEXT INTERVIEW

FIG. 01 · ALL 9 ON ONE PAGE

The cheat sheet.

PICK BY YOUR SITUATION

#	NAME	KIND	RUN IT	PICK IT WHEN
01	**Pinecone**	MANAGED	THEIR CLOUD ONLY	You want zero infrastructure: no servers, no index tuning, scaling handled for you.
02	**pgvector**	EXTENSION	IN POSTGRES	You're already on Postgres and want vectors next to your tables, joins and transactions.
03	**Qdrant**	OPEN SOURCE	SELF-HOST · CLOUD	Every query filters on metadata: user, tenant, date, permissions.
04	**Weaviate**	OPEN SOURCE	SELF-HOST · CLOUD	You need keyword and vector search together (hybrid) without building it yourself.
05	**Milvus**	OPEN SOURCE	SELF-HOST · ZILLIZ	You're heading to hundreds of millions or billions of vectors and need to scale out.
06	**Elasticsearch**	SEARCH ENGINE	SELF-HOST · ELASTIC	Your keyword search already lives there. Add vectors instead of a new system.
07	**Redis**	IN-MEMORY	SELF-HOST · CLOUD	Latency matters most, or you want a semantic cache in front of the LLM.
08	**Chroma**	OPEN SOURCE	IN-PROCESS · SERVER	Your first prototype: pip install, add text, query, in minutes.
09	**FAISS**	LIBRARY	INSIDE YOUR CODE	Research, offline batch jobs, or building your own search service. Not a database.

The one idea to keep: at small scale, most of these return almost the same top results. What differs is **who runs it**, **how well it filters**, whether it does **hybrid search**, and **how it scales**. Pick on those, not on a benchmark chart.

How to use this guide: read the cheat sheet, then the card for anything you couldn't defend in a design review. Each card ends with a one-line interview answer and a link to the official docs.

01 Pinecone

MANAGED

WHAT IT IS

A fully managed vector database. You create an index, upsert vectors with metadata through an API, and query for the top-k. Serverless indexes scale storage and compute for you; namespaces split data per customer.

TRADE-OFFS

Nothing to run or tune, so a small team ships fast. But it's closed source and cloud only: you can't self-host, your vectors leave your infrastructure, and the bill grows with data and queries.

No servers · no index tuning

> SAY IT **"Fully managed: we call an API and they run the index, so we pay in money instead of ops time."**

Docs · Pinecone documentation · docs.pinecone.io

02 pgvector Postgres extension

EXTENSION

WHAT IT IS

Adds a `vector` column type to Postgres, distance operators, and two approximate indexes: HNSW and IVFFlat. Without an index it does exact search.

TRADE-OFFS

Vectors live next to your rows: one database, one backup, SQL joins and transactions, filters in the WHERE clause. At very large scale or very high query rates, a dedicated vector database is easier to tune.

```
ORDER BY embedding <=> $1 LIMIT 5
```

> SAY IT **"If we're already on Postgres, pgvector keeps vectors next to our data: one system, real SQL filters."**

Docs · pgvector on GitHub · github.com/pgvector/pgvector

03 Qdrant

OPEN SOURCE

WHAT IT IS

An open-source vector database written in Rust. Each vector carries a JSON payload, and filters on that payload are applied during the HNSW search itself, with payload indexes to keep them fast.

TRADE-OFFS

Filtering after the search can return fewer than k results, or none; filtering inside the search doesn't. Strong for multi-tenant RAG and permission-aware retrieval. Self-host, or use Qdrant Cloud.

Filter inside the search, not after it

> SAY IT **"When every query filters by tenant or permission, filter during the vector search, not after it."**

Docs · Qdrant documentation · qdrant.tech/documentation

04 Weaviate

OPEN SOURCE

WHAT IT IS

An open-source vector database with hybrid search built in: one query runs BM25 keyword search and vector search, then fuses the two scores. Optional modules call an embedding model for you.

TRADE-OFFS

Keyword search catches exact terms that embeddings miss: product codes, names, error messages. One parameter sets the balance. Self-host, or use Weaviate Cloud.

alpha = 0 keyword only · 1 vector only

> SAY IT **"Hybrid search: BM25 for exact terms, vectors for meaning, fused in one query."**

Docs · Weaviate documentation · weaviate.io/developers/weaviate

05 Milvus

OPEN SOURCE

WHAT IT IS

An open-source, distributed vector database (LF AI & Data Foundation). Storage and compute are separate, so query and indexing nodes scale out on their own. Supports HNSW, IVF, DiskANN and GPU indexes.

TRADE-OFFS

Built for hundreds of millions to billions of vectors. The cost is operations: distributed mode runs several services. Milvus Lite runs in-process for prototypes; Zilliz Cloud is the managed version.

Scale out · many index types

> SAY IT **"For billions of vectors I'd pick a distributed database like Milvus that scales storage and compute separately."**

Docs · Milvus documentation · milvus.io/docs

06 Elasticsearch

SEARCH ENGINE

WHAT IT IS

The Lucene-based search engine many teams already run for keyword search. A `dense_vector` field adds approximate kNN search (HNSW), and one request can combine a BM25 query with a kNN query.

TRADE-OFFS

No new system to learn, secure or monitor, and your keyword relevance work carries over. Vector indexes need plenty of memory. OpenSearch, its open-source fork, offers similar k-NN search.

BM25 + kNN in one request

> SAY IT **"If search already runs on Elasticsearch, add a vector field before adding a new database."**

Docs · Elasticsearch kNN search · elastic.co/docs/solutions/search/vector/knn

07 Redis

IN-MEMORY

WHAT IT IS

The in-memory data store, with vector search through its query engine: FLAT (exact) and HNSW indexes, plus filters on tags, numbers and text. Often already in the stack as a cache.

TRADE-OFFS

Data in RAM means very low latency, and it makes a good semantic cache: reuse an LLM answer when a new question is close enough to an old one. RAM is also the limit: large collections get expensive.

Vectors in memory · semantic cache

> SAY IT **"When latency matters most, keep the vectors in memory; Redis also doubles as a semantic cache."**

Docs · Redis vector search · redis.io/docs/latest/develop/ai

08 Chroma

OPEN SOURCE

WHAT IT IS

An open-source embedding database built for developers. It runs inside your Python process or as a server, and if you add raw text it embeds it with a default model. Plugs into LangChain and LlamaIndex.

TRADE-OFFS

The fastest way from zero to a working RAG demo. Before production, check it against your scale, filtering and uptime needs, or move the same data to a server or cloud deployment.

`pip install chromadb`

> SAY IT **"For a prototype I'd use Chroma: it runs in-process, so there's nothing to deploy."**

Docs · Chroma documentation · docs.trychroma.com

09 FAISS Facebook AI Similarity Search

LIBRARY

WHAT IT IS

Meta's C++ library, with Python bindings, for similarity search over dense vectors: exact (Flat), clustered (IVF), compressed (PQ), graph (HNSW), and GPU versions of many of them.

TRADE-OFFS

Very fast, but it's a library: no server, no API, no built-in metadata filtering, replication or access control. You build those around it. Many vector databases borrowed its ideas.

`IndexFlatL2` · `IndexIVFPQ` · `IndexHNSWFlat`

> SAY IT **"FAISS is a library, not a database: great for search itself, but we'd own storage, filters and serving."**

Douze et al., 2024 · The Faiss library · arxiv.org/abs/2401.08281

Measure it: approximate indexes trade a little accuracy for speed. Take 100 real questions, compare the approximate top 10 with an exact (brute-force) search, and check the overlap (recall@10). Tune index settings on that number before blaming the database.

FIG. 02 · CHOOSING ONE

Which one should you use?

START FROM YOUR SITUATION

No one to run infrastructure, and you want to ship this week?	**Pinecone** 01
Your app **already runs on Postgres**?	**pgvector** 02
Every query **filters by user, tenant or permission**?	**Qdrant** 03
Users search **exact terms**: codes, names, error messages?	**Weaviate hybrid** 04
Heading to **billions of vectors**?	**Milvus** 05
Your **keyword search already lives** in Elasticsearch?	**Elasticsearch** 06
Latency is the tightest budget, or you want a **semantic cache**?	**Redis** 07
A **notebook, demo or first prototype**?	**Chroma** 08
Research or offline batch search, inside your own code?	**FAISS** 09

Try it: a working vector search with Chroma

Add text, and Chroma embeds it with its default model. Query with text, get the closest chunk back.

```
import chromadb

client = chromadb.Client()                        # in-memory, nothing to deploy
docs = client.create_collection("docs")
docs.add(ids=["1", "2"], documents=["Reset your password from Settings",
                                    "Refunds take 5 to 7 days"])
res = docs.query(query_texts=["how do I change my password"], n_results=1)
print(res["documents"][0][0])                     # Reset your password from Settings
```

Found this useful?

Follow SKD Neuron for more explainers on RAG, AI agents and AI system design. Save this for your next interview.

@SKDNEURON
SKDNEURON.COM