

warning "Backend update (October 2026)" GitHub Models inference was retired on July 30, 2026. Any references below are retained only for historical context while this lab is migrated. Use Lab 013 ? Free LLM backends to configure Foundry Local or Ollama before continuing.

Lab 039: Vector Database Comparison

Level: L300 Path: ? RAG Time: ~50 min ? Cost: Free ? All options have free tiers or local mode

What You'll Learn

Understand the key differences between major vector databases Compare pgvector, Chroma, Qdrant, and Azure AI Search on the same task Evaluate each on: setup complexity, query speed, filtering, cloud integration Choose the right vector database for your agent's requirements

Introduction

Choosing a vector database is one of the most consequential architecture decisions for a RAG-based agent. The "best" choice depends on your team's existing stack, scale requirements, filtering needs, and cloud strategy.

The candidates:

Database | Type | Free option | Best for | pgvector | PostgreSQL extension | Azure free tier / local | Teams already using PostgreSQL | Chroma | Embedded / server | Fully open-source | Local development, small projects | Qdrant | Dedicated vector DB | Qdrant Cloud free tier | High-scale production, advanced filtering | Azure AI Search | Azure service | Free tier (1 index) | Azure-native, hybrid search, enterprise |

Prerequisites

[diagram or code sample]

No API keys needed for Chroma (local) and Qdrant (local mode). GitHub Token required for embeddings: [diagram or code sample]

Setup: Shared Embedding Function

All four tests use the same OutdoorGear product data and the same embedding model:

[diagram or code sample]

Option A: Chroma (Local, No Setup)

Chroma is the easiest way to get started ? pure Python, runs in-memory or on-disk:

[diagram or code sample]

Option B: Qdrant (Local Server Mode)

Qdrant offers advanced filtering and scales to hundreds of millions of vectors:

[diagram or code sample]

[diagram or code sample]

Option C: pgvector (Azure PostgreSQL or Local)

See Lab 031 for the full pgvector setup. Quick comparison:

[diagram or code sample]

Option D: Azure AI Search (Hybrid Search)

Azure AI Search uniquely supports hybrid search: vector + BM25 keyword search combined with semantic reranking:

[diagram or code sample]

Decision Framework

[diagram or code sample]

Comparison Table

| pgvector | Chroma | Qdrant | Azure AI Search | Setup | Medium (DB needed) | Minimal | Easy (Docker) | Medium (Azure) | Local dev | ? (Docker) | ? (in-memory) | ? (Docker) | ?

(Azure only) | Hybrid search | ? (manual) | ? | ? | ? (built-in) | Filtering | SQL WHERE | Basic metadata | Advanced | Full OData | Scale | Moderate (100M) | High (enterprise) | Azure integration | ? (managed PG) | ? | Partial | ? (native) | Cost (free tier) | Free tier PG | Free | Qdrant Cloud free | 1 index free |

? Knowledge Check

question "1. What is hybrid search, and why is it often better than pure vector search?"

Hybrid search combines vector (semantic) search with keyword (BM25) search and ranks results using both signals. Vector search excels at semantic understanding ("warm shelter" ? sleeping bag) but can miss exact matches (a specific product ID). BM25 is excellent for exact keyword matches but misses synonyms. Combining them outperforms either alone, especially for product names, SKUs, and specialized terminology.

question "2. Why would you choose pgvector over a dedicated vector database like Qdrant?"

If you already have PostgreSQL as your primary database, pgvector adds vector search without adding another service to operate, maintain, and pay for. The data lives alongside your relational data ? you can JOIN product records with their embeddings in a single query. For most applications under 1M vectors, pgvector's performance is excellent. Choose Qdrant when you need > 100M vectors or very advanced filtering.

question "3. What is the IVFFlat index in pgvector, and when should you use HNSW instead?"

IVFFlat (Inverted File Index with Flat quantization): fast to build, uses less memory, good for datasets that don't change frequently. Uses approximate search ? lists parameter controls recall/speed trade-off. HNSW (Hierarchical Navigable Small World): better recall, faster queries, but higher memory usage and slower to build. Use IVFFlat for datasets < 1M that don't change much; use HNSW for frequently updated datasets or when recall is critical. Both require pgvector ? 0.5.0.

Summary

For the OutdoorGear learning hub scenario (< 10K products, Azure infrastructure, team knows SQL):

Recommended: pgvector on Azure Database for PostgreSQL Flexible Server.

No new service to learn SQL + vector in one query Free tier available Production-ready with Lab 031 migrations

Next Steps

Build the pgvector setup: ? Lab 031 ? pgvector Semantic Search Full RAG application: ? Lab

022 ? RAG with GitHub Models + pgvector Enterprise RAG with evaluation: ? Lab 042 ?

Enterprise RAG *(coming soon)*

Supporting kit

The site build generates stable PDF and ZIP downloads. The ZIP contains the lab guide and every supporting file declared for this lab.

Download generated PDF Download generated ZIP

Supporting file: CHEATSHEET.md

Supporting files: Lab 039: Vector Database Comparison

Start with the documented local or free path. Record assumptions and validation evidence.

Keep hosted services optional unless the lab marks them required.

Supporting file: README.md

Lab 039: Vector Database Comparison

Use these files with the lab guide and generated kit.