QDRANT CLUSTER SCALING

This example simulates **Production Scaling with Qdrant Cluster** (shards + replication). You can run it on your **local Qdrant cluster** (multiple nodes) or adapt it later to Kubernetes.

**"""**

**Qdrant Cluster Example**

**----------------------**

**This script demonstrates:**

**1. Creating a distributed collection with shards & replication**

**2. Inserting documents into Qdrant**

**3. Running similarity search**

**4. Using filtering + scaling concepts**

**To run this, you need:**

**- A Qdrant cluster (at least 3 nodes) running on Docker/Kubernetes**

**- Python packages: qdrant-client, fastembed**

**"""**

from qdrant\_client import QdrantClient

from qdrant\_client.models import (

VectorParams, Distance, PointStruct,

Filter, FieldCondition, MatchValue

)

from fastembed import TextEmbedding

# ==============================

# 1. Connect to Cluster

# ==============================

# Connect to the coordinator node (usually node1:6333 in Docker)

client = QdrantClient(url="http://localhost:6333")

# Local embedder (no OpenAI API needed)

embedder = TextEmbedding()

print("✅ Connected to Qdrant cluster")

# ==============================

# 2. Create Distributed Collection

# ==============================

client.recreate\_collection(

collection\_name="news\_articles",

vectors\_config=VectorParams(size=384, distance=Distance.COSINE),

shard\_number=6, # Split into 6 shards across nodes

replication\_factor=2 # Each shard has 2 replicas

)

print("✅ Cluster collection 'news\_articles' created with sharding & replication")

# ==============================

# 3. Insert Example Documents

# ==============================

docs = [

{"id": 1, "text": "Breaking news: elections in Europe", "category": "politics"},

{"id": 2, "text": "Python is widely used in AI research", "category": "tech"},

{"id": 3, "text": "Berlin is the capital of Germany", "category": "travel"},

{"id": 4, "text": "AI is transforming healthcare rapidly", "category": "health"},

{"id": 5, "text": "NASA discovers new exoplanets", "category": "science"},

{"id": 6, "text": "Tourism in France is growing", "category": "travel"},

]

# Convert text → vectors

vectors = list(embedder.embed([d["text"] for d in docs]))

# Create points (vector + payload)

points = [

PointStruct(id=doc["id"], vector=vec, payload=doc)

for doc, vec in zip(docs, vectors)

]

# Upsert into Qdrant

client.upsert(collection\_name="news\_articles", points=points)

print("✅ Inserted documents into 'news\_articles' collection")

# ==============================

# 4. Similarity Search

# ==============================

query = "latest updates on AI research"

query\_vec = list(embedder.embed([query]))[0]

results = client.query\_points(

collection\_name="news\_articles",

query=query\_vec,

limit=3,

with\_payload=True

).points

print("\n🔍 Similarity Search Results:")

for r in results:

print(f"- {r.payload['text']} | Score: {round(r.score, 3)}")

# ==============================

# 5. Similarity Search with Filtering

# ==============================

# Only return documents where category = 'travel'

filter\_condition = Filter(

must=[FieldCondition(key="category", match=MatchValue(value="travel"))]

)

results\_filtered = client.query\_points(

collection\_name="news\_articles",

query=query\_vec,

limit=3,

with\_payload=True,

query\_filter=filter\_condition

).points

print("\n🔍 Filtered Search Results (category=travel):")

for r in results\_filtered:

print(f"- {r.payload['text']} | Score: {round(r.score, 3)}")

# ==============================

# 6. Update & Delete Example

# ==============================

# Update article with id=2

client.upsert(

collection\_name="news\_articles",

points=[PointStruct(

id=2,

vector=vectors[1],

payload={"text": "Python programming for AI", "category": "tech"}

)]

)

print("\n✅ Updated article with ID=2")

# Delete article with id=5

client.delete(collection\_name="news\_articles", points\_selector=[5])

print("✅ Deleted article with ID=5")

**🔹 Explanation of the Code**

1. **Connect to cluster**
   * QdrantClient(url="http://localhost:6333") → connects to your cluster’s coordinator.
2. **Create collection with shards & replication**
   * shard\_number=6 → splits collection into 6 parts.
   * replication\_factor=2 → each shard stored twice for safety.
3. **Insert documents**
   * Uses fastembed to generate embeddings (no API key needed).
   * Stores text + category as payload.
4. **Similarity search**
   * Finds top 3 closest documents to "latest updates on AI research".
5. **Filtering**
   * Restrict results to a category (like "travel").
6. **Update & delete**
   * Update payload of a document (ID=2).
   * Delete a document (ID=5).

**🔹 How to Run**

1. Make sure Qdrant cluster is running (Docker/Kubernetes).  
   Example for single-node testing:
2. docker run -p 6333:6333 qdrant/qdrant

For production, run multiple nodes with replication enabled.

1. Install dependencies:
2. pip install qdrant-client fastembed
3. Save code as qdrant\_cluster\_example.py and run:

python qdrant\_cluster\_example.py